

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAISO - CHILE



**“ESTRATEGIAS DE PARTICIÓN EN ASAP PARA
STRINGS CON RUNS Y ALFABETOS GRANDES”**

CARLOS EUGENIO ROJAS MORALES

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA**

Profesor Guía: Diego Arroyuelo
Profesor Correferente: Elizabeth Montero

Junio - 2024

DEDICATORIA

Dedicado a todos los que confiaron en mí y me apoyaron tanto durante mi formación como durante la realización de mi memoria, dedicado a mi familia y a Victoria Mezzano.

AGRADECIMIENTOS

Agradecimientos cordiales a mi profesor guía, Diego Arroyuelo, ya que sin el tiempo que este invirtió y sus conocimientos no habría sido posible llevar a cabo esta investigación junto a los recursos que me facilitó para realizar la experimentación necesaria para mi memoria.

RESUMEN

En la actualidad, la compresión de datos y su recuperación son aspectos vitales, especialmente en el contexto de textos largos y repetitivos, que es el enfoque principal de este documento. Se propone la implementación y evaluación de distintas estrategias de particionamiento aplicadas a una variedad de combinaciones de estructuras de datos comprimidas sobre la estructura ASAP. Entre estas estrategias, destaca el rendimiento de la estrategia A4, la cual utiliza "*dense partitioning*" junto con el uso de las cabeceras de cada run del texto para la estructura que maneja el *mapping* $m(\alpha)$. Esta combinación logró resultados superiores al *baseline* actual, mejorando tanto el manejo del espacio como la velocidad de consulta para las estructuras comprimidas utilizadas. En particular, la estructura ASAP RLMN(INT) RLE demostró el mejor rendimiento.

Palabras Clave— Alphabet Partitioning; Estrategia de partición; Wavelet Tree; ASAP; Compresión de datos

ABSTRACT

Currently, data compression and retrieval are vital aspects, especially in the context of long and repetitive texts, which is the main focus of this document. We propose the implementation and evaluation of different partitioning strategies applied to a variety of combinations of compressed data structures based on the ASAP structure. Among these strategies, the performance of the A4 strategy stands out, which uses "*dense partitioning*" along with the use of headers for each run of the text for the structure that handles the mapping $m(\alpha)$. This combination achieved results superior to the current baseline, improving both space management and query speed for the compressed structures used. In particular, the ASAP RLMN(INT) RLE structure demonstrated the best performance.

Keywords— Alphabet Partitioning; Partitioning Strategy; Wavelet Tree; ASAP; Data Compression

GLOSARIO

Baseline: Es un instrumento de métrica que se utiliza con fines de comparación, normalmente establecidos a partir de resultados pasados

Trade-off: Se refiere a una situación en la cual para acercarse a alcanzar un objetivo se debe sacrificar otro

AP: Alphabet Partitioning.

ASAP: Agile and Succinct Alphabet Partitioning

BWT: Burrow Wheeler Transformation.

RLE: Run Length Encoding.

RLMN: Run-Length FM-index

Run: Se refiere a una secuencia continua de caracteres idénticos en una cadena de texto.

INDICE DE CONTENIDOS

RESUMEN.....	4
ABSTRACT	4
INDICE DE FIGURAS.....	8
INDICE DE TABLAS.....	9
INTRODUCCIÓN	11
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	12
1.1 CONTEXTO Y SITUACIÓN ACTUAL	12
1.2 OBJETIVOS	12
CAPÍTULO 2: MARCO CONCEPTUAL	13
2.1 OPERACIONES ELEMENTALES DE BUSQUEDA.....	13
2.1.1 RANK	13
2.1.2 SELECT	13
2.1.3 ACCESS	13
2.2 WAVELET TREE	13
2.2.1 CREACIÓN DE WAVELET TREE	13
2.2.2 OPERACIONES ELEMENTALES DE BUSQUEDA – WAVELET TREE	14
2.2.2.1 RANK	14
2.2.2.2 SELECT	15
2.2.2.3 ACCESS	16
2.3 ENTROPÍA H_0	17
2.4 ALPHABET PARTITIONING	17
2.4.1 RANK	18
2.4.2 SELECT	18
2.4.3 ACCESS	19
2.5 ASAP (Agile and Succinct Alphabet Partitioning)	19
2.5.1 RANK	20
2.5.2 SELECT	20
2.5.3 ACCESS	20
2.6 TRANSFORMACIÓN BURROW-WHEELER	20
2.7 FM-COUNT.....	22
2.8 RUN LENGTH ENCODING (RLE).....	23
2.9 RLFM.....	23
2.10 VECTOR S18.....	23
CAPÍTULO 3: PROPUESTA DE SOLUCION	24
3.1 Estrategia A1	24
3.2 Estrategia A2	24
3.3 Estrategia A3	25
3.4 Estrategia A4	25
3.5 MAPPING M.....	25
3.6 RANK-ASAP MODIFICADO	25
3.7 ESTRUCTURA MAPPING “M” MODIFICADA	27

3.8 ALTERNATIVAS A PROBAR	27
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	28
4.1 ESPECIFICACIONES DEL SISTEMA	28
4.2 TEXTOS/ALFABETOS	28
4.2.1 COREUTILS	28
2.5.1 EINSTEIN.EN	29
2.5.1 EINSTEIN.DE	29
2.5.1 ENGLISH	29
2.5.1 WORLD LEADERS	30
4.3 RESULTADOS ALTERNATIVA A1	30
4.3.1 COMPARACIÓN A1 VS BASELINE	32
4.4 RESULTADOS ALTERNATIVA A2	33
4.4.1 COMPARACIÓN A2 VS BASELINE	35
4.5 RESULTADOS ALTERNATIVA A3	36
4.5.1 COMPARACIÓN A3 VS BASELINE	37
4.6 RESULTADOS ALTERNATIVA A4	39
4.6.1 COMPARACIÓN A4 VS BASELINE	40
CAPÍTULO 5: CONCLUSIONES	42
5.1 Principales Contribuciones y Aplicaciones	44
5.2 Trabajo Futuro	44
REFERENCIAS BIBLIOGRÁFICAS.....	47
ANEXOS.....	47
ANEXO A: RESULTADOS COMPLETOS A1	49
ANEXO B: RESULTADOS COMPLETOS A2	59
ANEXO C: RESULTADOS COMPLETOS A3	69
ANEXO D: RESULTADOS COMPLETOS A4	79

INDICE DE FIGURAS

Figura 1: Wavelet tree del string S = “alabar_a_la_alabarda”	14
Figura 2: Operación Rank sobre Wavelet tree de string S = “alabar_a_la_alabarda”	15
Figura 3: Operación Select sobre Wavelet Tree de string S = “alabar_a_la_alabarda”	16
Figura 4: Operación Access sobre Wavelet tree de string S = “alabar_a_la_alabarda”	16
Figura 5: Estructuras de datos Alphabet Partitioning para el string s=alabar_a_la_alabarda	18
Figura 6: Estructuras de datos ASAP para el string s=alabar_a_la_alabarda	19
Figura 7: n rotaciones cíclicas sobre string T.	20
Figura 8: Ordenamiento léxico sobre la estructura generada del string T.	21
Figura 9: Gráficos Alternativa A1	31
Figura 10: Gráficos Alternativa A2	34
Figura 11: Gráficos Alternativa A3	37
Figura 12: Gráficos Alternativa A4	40

INDICE DE TABLAS

Tabla 1: Información Coreutils	29
Tabla 2: Información Einstein.en	29
Tabla 3: Información Einstein.de	29
Tabla 4: Información English.001.2	30
Tabla 5: Información World Leaders	30
Tabla 6: Comparación A1 vs Baseline, Coreutils	32
Tabla 7: Comparación A1 vs Baseline, Einstein.de	32
Tabla 8: Comparación A1 vs Baseline, English.....	32
Tabla 9: Comparación A2 vs Baseline, Coreutils	35
Tabla 10: Comparación A2 vs Baseline, Einstein.de	35
Tabla 11: Comparación A2 vs Baseline, Coreutils	35
Tabla 12: Comparación A3 vs Baseline, Coreutils	37
Tabla 13: Comparación A3 vs Baseline, Einstein.de	38
Tabla 14: Comparación A3 vs Baseline, English	38
Tabla 15: Comparación A4 vs Baseline, Coreutils	40
Tabla 16: Comparación A4 vs Baseline, Einstein.de	40
Tabla 17: Comparación A4 vs Baseline, English	41
Tabla 18 : Resultados A1 – Coreutils m=4,m=8.....	49
Tabla 19: Resultados A1 – Coreutils m=16	50
Tabla 20: Resultados A1 – Einstein.de m=4,m=8	51
Tabla 21: Resultados A1 – Einstein.de m=16	52
Tabla 22: Resultados A1 – Einstein.en m=4,m=8	53
Tabla 23: Resultados A1 – Einstein.en m=16	54
Tabla 24: Resultados A1 – English m=4,m=8.....	55
Tabla 25: Resultados A1 – English m=16	56
Tabla 26: Resultados A1 – World Leaders m=4,m=8.....	57
Tabla 27: Resultados A1 – World Leaders m=16.....	58
Tabla 28: Resultados A2 – Coreutils m=4,m=8.....	59
Tabla 29: Resultados A2 – Coreutils m=16	60
Tabla 30: Resultados A2 – Einstein.de m=4, m=8	61
Tabla 31: Resultados A2 – Einstein.de m=16	62
Tabla 32: Resultados A2 – Einstein.en m=4, m=8	63
Tabla 33: Resultados A2 – Einstein.en m=16	64
Tabla 34: Resultados A2 – English m=4, m=8	65
Tabla 35: Resultados A2 – English m=16	66
Tabla 36: Resultados A2 – World Leaders m=4, m=8.....	67
Tabla 37: Resultados A2 – World Leaders m=16.....	68
Tabla 38: Resultados A3 – Coreutils m=4, m=8.....	69
Tabla 39: Resultados A3 – Coreutils m=16	70
Tabla 40: Resultados A3 – Einstein.de m=4, m=8	71
Tabla 41: Resultados A3 – Einstein.de m=16	72

Tabla 42: Resultados A3 – Einstein.en m=4, m=8	73
Tabla 43: Resultados A3 – Einstein.en m=16	74
Tabla 44: Resultados A3 – English m=4, m=8	75
Tabla 45: Resultados A3 – English m=16	76
Tabla 46: Resultados A3 – World Leaders m=4, m=8	77
Tabla 47: Resultados A3 – World Leaders m=16	78
Tabla 48: Resultados A4 – Coreutils m=4, m=8	79
Tabla 49: Resultados A4 – Coreutils m=16	80
Tabla 50: Resultados A4 – Einstein.de m=4, m=8	81
Tabla 51: Resultados A4 – Einstein.de m=16	82
Tabla 52: Resultados A4 – Einstein.en m=4, m=8	83
Tabla 53: Resultados A4 – Einstein.en m=16	84
Tabla 54: Resultados A4 – English m=4, m=8	85
Tabla 55: Resultados A4 – English m=16	86
Tabla 56: Resultados A4 – World Leaders m=4, m=8	87
Tabla 57: Resultados A4 – World Leaders m=16	88

INTRODUCCIÓN

Las estructuras de datos comprimidas y las estrategias de particionamiento de alfabetos han sido usadas como una herramienta esencial para mejorar tanto la eficiencia en la búsqueda como en la compresión de textos. Este documento se centra en el desarrollo y evaluación de nuevas estrategias de particionamiento de alfabetos para el enfoque ASAP, especialmente dirigidas a *strings* con repetición de caracteres (*runs*) y alfabetos grandes.

El objetivo principal de esta investigación es proponer y validar teórica y experimentalmente nuevas estrategias de particionamiento que sean competitivas en términos de rendimiento y eficiencia. Para lograr esto, se plantean tres objetivos específicos: desarrollar nuevas estrategias de particionamiento, demostrar su competitividad a través de experimentos e, implementar estas estrategias en código fuente abierto, facilitando su uso y evaluación por la comunidad científica.

La investigación se orienta específicamente a textos con alta repetición de caracteres, aprovechando técnicas como la BWT y la RLE las cuales son especialmente efectivas en estos contextos. La eficiencia en memoria y la velocidad de consulta se evaluará rigurosamente para garantizar que las nuevas estrategias de particionamiento sean adecuadas para textos con largos y repetitivos en conjunto con estructuras de datos comprimidas.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1 CONTEXTO Y SITUACIÓN ACTUAL

En el mundo de la información, los datos son la parte fundamental de esta área. Año tras año, se generan más y más datos, y cada vez a una escala mayor. Esto plantea un problema principal: el almacenamiento de dichos datos. Dado que la cantidad de datos es tan grande, la única manera de almacenarlos de manera razonable es mediante la compresión. Este campo ha sido ampliamente explorado por la informática debido a su importancia, pero al comprimir los datos surge la necesidad de recuperar estos posteriormente, esto nos lleva a un problema en el que cuanto más se comprime una estructura, a menudo se requieren más pasos y tiempo para la recuperación de los datos. Así, se convierte en un problema de *trade-off* entre la velocidad de consulta de los datos y el espacio en memoria utilizado.

En este documento, nos centraremos en los datos correspondientes a textos muy largos y repetitivos para aplicar estrategias sobre estructuras de datos comprimidas, con el objetivo de conseguir el mejor *trade-off* posible entre velocidad y espacio mediante el uso de estrategias de particionamiento sobre los textos. Además, utilizaremos estructuras que soporten las operaciones fundamentales de Rank, Select y Access, las cuales son de gran importancia a la hora de realizar consultas sobre un texto. Si no realizamos esta investigación, estrategias y privar a la industria de resultados que pueden ayudar a resolver este problema fundamental.

1.2 OBJETIVOS

Objetivo General: Nuevas estrategias de particionamiento para el enfoque “*Alphabet Partitioning*”, para *strings* con *runs* y alfabetos grandes, que sean competitivas tanto teóricamente como experimentalmente.

Objetivos Específicos:

- Obtener nuevas estrategias de particionamiento para el enfoque “*Alphabet Partitioning*”, para *strings* con *runs* y alfabetos grandes.
- Mostrar que las estrategias de particionamiento definidas en el punto anterior son competitivas experimentalmente.
- Producir código fuente *Open Source* con la implementación de las propuestas de estrategias de particionamiento.

CAPÍTULO 2: MARCO CONCEPTUAL

Dado que se desea verificar el rendimiento de diversos tipos de partición es imperativo primero definir el estado del arte actual en el sector para lo cual es indispensable primero realizar algunas definiciones acordes al tema descrito.

2.1 OPERACIONES ELEMENTALES DE BUSQUEDA

Para definir las operaciones elementales de búsqueda que se pueden realizar sobre *strings* se referenciará a Arroyuelo et al (Arroyuelo et al., 2024) , siendo un *string* $s[1..n]$ de largo n que posee un alfabeto $\Sigma = \{0, \dots, \sigma - 1\}$.

2.1.1 RANK

La operación elemental $s.Rank_c(i)$ retornará la cantidad de veces que se repite un carácter dentro del *string* s hasta la posición i .

2.1.2 SELECT

La operación elemental $s.Select_c(i)$ retornará la posición en la cual se encuentra la i -ésima repetición del símbolo c .

2.1.3 ACCESS

La operación elemental $s.Access(i)$ retornará el carácter que se encuentra en la posición i del *string* s .

2.2 WAVELET TREE

Un Wavelet tree es una estructura de datos avanzada la cual permite representar almacenar cadenas de texto de manera eficiente, además de permitir realizar las operaciones Rank, Select y Access sobre ella.

La construcción de un Wavelet tree se basa en la partición recursiva del alfabeto de la secuencia. En cada nivel del árbol, se divide el conjunto de caracteres en dos subconjuntos, y se utiliza un vector de bits para indicar a qué subconjunto pertenece cada carácter de la secuencia. Los nodos hoja del árbol representan caracteres individuales del alfabeto.

2.2.1 CREACIÓN DE WAVELET TREE

Para entender el funcionamiento del Wavelet tree se puede definir el siguiente ejemplo, se tiene un *string* $S = \text{"alabar_a_la_alabarda"}$ con un alfabeto $\Sigma = \{_, a, b, d, l, r\}$, el proceso para la creación del Wavelet tree sería el siguiente:

- **Partición del Alfabeto:** Se parte el alfabeto Σ en dos subconjuntos aproximadamente del mismo tamaño. Por ejemplo, si $\Sigma = \{a, b, d, l, r\}$, podríamos tener $\Sigma_1 = \{-, a, b\}$ y $\Sigma_2 = \{d, l, r\}$
- **Bit Vector:** Se crea un vector de bits en el nodo raíz del árbol, donde cada bit representa un carácter de la secuencia. Un bit es “0” si el carácter pertenece al primer subconjunto Σ_1 , y “1” si pertenece al segundo subconjunto Σ_2 .
- **Construcción Recursiva:** Este proceso se repite de manera recursiva para cada subconjunto, creando nuevos nodos y vectores de bits, hasta que se llega a los nodos hoja que representan caracteres individuales.

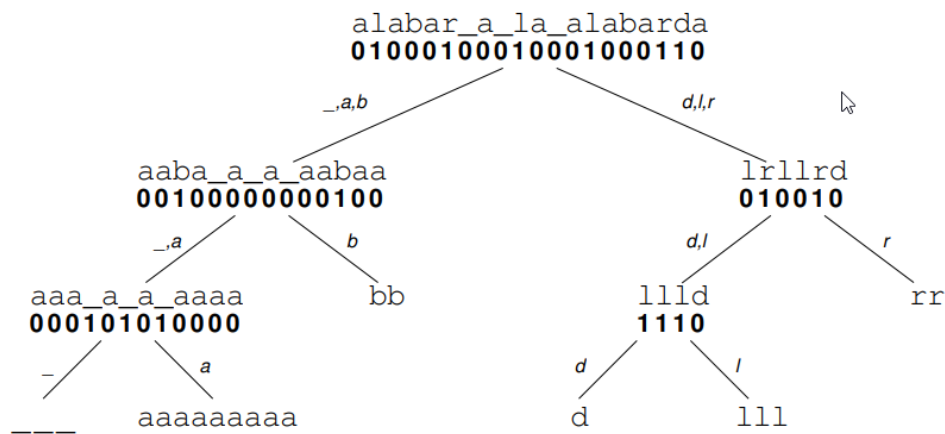


Figura 1: Wavelet tree del string $S = \text{"alabar_a_la_alabarda"}$

Fuente: Gonzalo Navarro (Navarro, 2014)

2.2.2 OPERACIONES ELEMENTALES DE BUSQUEDA – WAVELET TREE

Una ventaja crucial que tiene el uso de Wavelet Tree es que este soporta las operaciones elementales de búsqueda, a continuación, se detallara el algoritmo que sigue cada una de estas operaciones en esta estructura según lo detalla G. Navarro (Navarro, 2014).

2.2.2.1 RANK

Para realizar la operación Rank en un Wavelet Tree se debe comenzar desde el nodo raíz buscando si el carácter buscado es un 0 o un 1, luego se debe contar la cantidad del dígito respectivo en el bit vector raíz y avanzar hacia la rama a la que corresponde el carácter, luego repetir el proceso cambiando la posición en la que terminara dependiendo de la cantidad encontrada de 0s o 1s según corresponda en el nivel anterior.

Para el ejemplo usado se toma $s.Rank_l(11)$ se toman los 11 primeros dígitos del bit vector, al revisar este notamos que el carácter “l” corresponde el dígito 1, notando que realizando un rank en este bit vector hasta la posición 11, buscando las repeticiones de “1” se tiene que este sería igual a 3. Se continua por la rama correspondiente a la partición que contiene “l” cambiando el índice de termino a 3 repitiendo nuevamente el proceso anterior se tiene que ahora la cantidad de 0s presentes es de dos, por lo que al pasar al siguiente nodo tendremos que hasta la posición 2 se encuentran dos veces el dígito 1, llegando directamente a la hoja del árbol donde se obtiene que $s.Rank_l(11) = 2$.

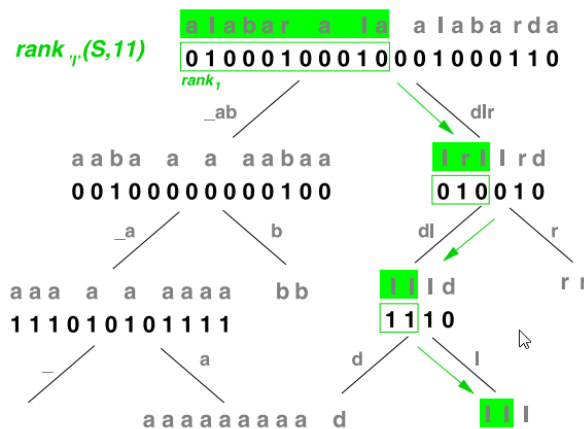


Figura 2: Operación Rank sobre Wavelet tree de $string S = \text{"alabar_a_la_alabarda"}$
Fuente: Gonzalo Navarro (Navarro, 2014)

2.2.2.2 SELECT

Al realizar la operación Select se inicializa desde la hoja que contenga el carácter y se deberá subir hasta la raíz del árbol, denotando que cada vez que se suba un nodo padre se deberá modificar la posición de termino para contener los caracteres englobados por el Select del hijo del que se subió hacia el padre.

Para el ejemplo usado se toma $s.Select_b(2)$, por lo que ubicándonos inicialmente en la hoja correspondiente al carácter “b” subimos a su padre, notando que realizando un select en el bit vector buscando la posición donde la cantidad de repeticiones de “1” sea igual a 2, obtenemos que esto se cumple en la posición 12, por lo nuevamente realizamos un Select en el padre de igual manera, obteniendo que $s.Select_b(2)=16$.

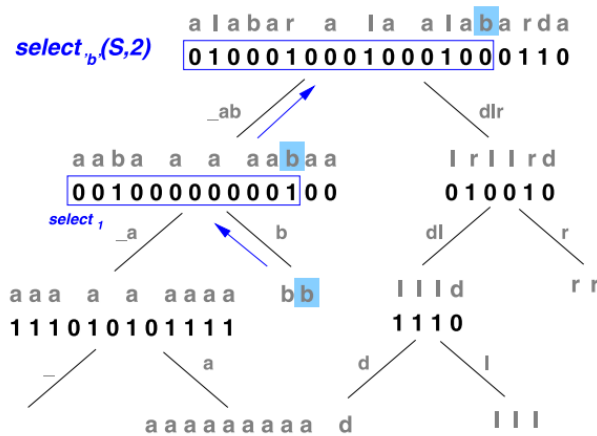


Figura 3: Operación Select sobre Wavelet Tree de *string* S = “alabar_a_la_alabarda”
Fuente: Gonzalo Navarro (Navarro, 2014)

2.2.2.3 ACCESS

La operación Access se aplicará directamente en la raíz del árbol, luego se debe obtener a que hijo se llega desde la posición correspondiente al Access (0 o 1) y realizar la operación Rank sobre el bit vector en el que nos encontramos para obtener el número de repeticiones del dígito encontrado, luego se aplicará recursivamente Rank sobre los nodos hijo hasta encontrar la hoja con el carácter que corresponde al resultado de la operación Access.

Para el ejemplo se toma $s.Access(11)$, se obtiene que el carácter en la posición 11 del bit vector es un 0, por lo que se realiza un $s.Rank_0(11)$ obteniendo 8, por lo que ahora iremos a su nodo hijo, posición 8 notando que el dígito en el bit vector para esta posición es 0, por lo que se volverá a realizar el proceso de la operación rank para este y luego pasaremos a su hijo finalmente obteniendo que $s.Access(11) = a$.

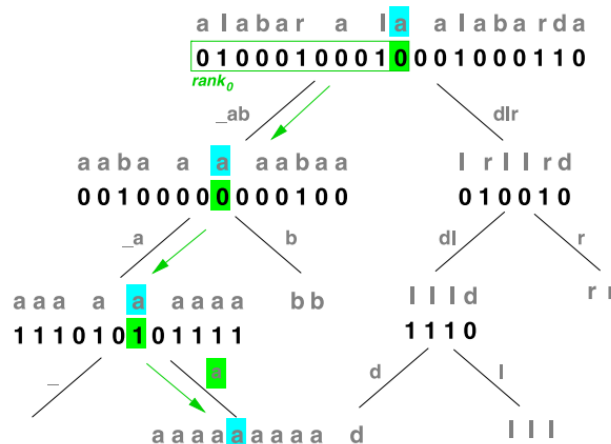


Figura 4: Operación Access sobre Wavelet tree de *string* S = “alabar_a_la_alabarda”
Fuente: Gonzalo Navarro (Navarro, 2014)

2.3 ENTROPÍA H_0

La entropía de orden 0 se refiere al mínimo de compresión posible para un *string* s teniendo que con un H_0 pequeño el *string* s será más compresible que para un H_0 grande. La fórmula asociada al cálculo de H_0 es la siguiente:

$$H_0 = \sum_{c \in \Sigma} \frac{n_c}{n} \log \frac{n}{n_c}$$

Siendo n el largo del *string* s y n_c la cantidad de repeticiones del carácter $c \in \Sigma_s$, el resultado H_0 indicará cuántos bits en promedio se usarán por carácter al comprimir el *string* s , tomando su valor máximo $\lg \sigma$ cuando se tiene la misma frecuencia para todos los caracteres del alfabeto Σ .

2.4 ALPHABET PARTITIONING

En el año 2010 se publica por Barbay et al (Barbay et al., 2010) la estrategia Alphabet Partitioning la cual consiste en la partición del alfabeto en subalfabetos, los cuales deben poseer una intersección nula entre sí (no repetir caracteres entre subalfabetos) y al realizar la unión entre estos obtener Σ , siendo Σ el alfabeto de un *string* $s[1..n]$, permitiendo además el uso de las operaciones elementales de búsqueda Rank, Select y Access en sus estructuras, por lo que apegándonos a la demostración de Barbay et al (Barbay et al., 2010) respecto al espacio utilizado por las estructuras será de $nH_0(s) + o(nH_0(s))$.

Apegándose a las definiciones presentadas por Barbay et al (Barbay et al., 2014) se tienen las siguientes estructuras:

- Mapeo de caracteres $m[1 \dots \sigma]$: Esta es una secuencia que asigna a cada carácter α en el rango de $[1 \dots \sigma]$ un subalfabeto correspondiente $m[\alpha]$, en este caso el método usado será *sparse*, el cual se define como

$$m[a] = \lceil \log(n/|s|_a) \log n \rceil$$

Donde $|s|_a$ denota el número de ocurrencias del carácter α en el *string* s , notando que $(m[a] \leq \lceil \log_2 n \rceil)$ para cualquier a en $[1 \dots \sigma]$.

- Secuencia $t[1 \dots n]$ de subalfabetos: Esta secuencia se obtiene del *string* s reemplazando cada ocurrencia de un carácter α por $m[\alpha]$, es decir, $(t[i] = m[s[i]])$.
- Subsecuencias $s_\alpha[1 \dots \sigma_\alpha]$ de caracteres de cada subalfabeto: Para $0 \leq \alpha \leq$

$\lceil \log_2 n \rceil$, definimos $\sigma_\alpha = |m|_\alpha$, que es el número de caracteres distintos de s reemplazados por α en t . Entonces $s_\alpha[1 \dots |t|_\alpha]$ es la secuencia sobre $[1 \dots (\sigma_\alpha)]$ definida por:

$$s_\alpha[t.rank_\alpha(i)] = m.rank_\alpha(s[i])$$

para todo $1 \leq i \leq n$ tal que $t[i] = \alpha$.

Cabe tener en cuenta que tanto el vector $t[1 \dots n]$ como $s_\alpha[1 \dots \sigma_\alpha]$ pueden ser representados como Wavelet tree de baja altura, por lo cual su rendimiento no debería presentar mayor problema.

Por ejemplo, se tiene al *string* $s = \text{alabar_a_la_alabarda}$ al cual se le aplicará alphabet partitioning obteniendo el siguiente resultado:

$$\begin{array}{c}
 s = \text{alabar_a_la_alabarda} \\
 \Sigma_1 = \{a\}; \quad \Sigma_2 = \{l, -\}; \quad \Sigma_3 = \{b, r\}; \quad \Sigma_4 = \{d\} \\
 t = 1 \ 2 \ 1 \ 3 \ 1 \ 3 \ 2 \ 1 \ 2 \ 2 \ 1 \ 2 \ 1 \ 2 \ 1 \ 3 \ 1 \ 3 \ 4 \ 1 \\
 \begin{array}{cccc}
 s_1 & s_2 & s_3 & s_4 \\
 \text{aaaaaaaa} & \text{l_l_l} & \text{brbr} & \text{d}
 \end{array}
 \end{array}$$

Figura 5: Estructuras de datos *Alphabet Partitioning* para el *string* $s = \text{alabar_a_la_alabarda}$
Fuente: Diego Arroyuelo et al (Arroyuelo et al., 2024)

Respecto a las operaciones fundamentales Rank, Select y Access, según Barbay et al (Barbay et al., 2010) n, con $\alpha \in \Sigma$ se definen como:

2.4.1 RANK

La operación Rank tiene un tiempo de ejecución promedio de $O(\log \log \sigma)$ y se define como:

$$s.rank_\alpha(i) = s_l.rank_c(t.rank_l(i)), \text{ donde } l = m.access(\alpha) \text{ y } c = m.rank_l(\alpha)$$

2.4.2 SELECT

La operación Select tiene un tiempo de ejecución promedio de $O(1)$ y se define como:

$$s.select_\alpha(i) = t.select_l(s_l.select_c(i)), \text{ donde } l = m.access(\alpha) \text{ y } c = m.rank_l(\alpha)$$

2.4.3 ACCESS

La operación Access tiene un tiempo de ejecución promedio de $O(\log \log \sigma)$ y se define como:

$$s.access(i) = m.select_l(s_l.access(t.rank_l(i))), \text{ donde } l = t.access(i)$$

2.5 ASAP (Agile and Succinct Alphabet Partitioning)

Es una estructura de datos comprimida nacida de una intervención en AP por D. Arroyuelo y E. Sepúlveda (Arroyuelo y Sepúlveda, 2019), esta maneja secuencias de caracteres de manera eficiente en términos de espacio y tiempo, este utiliza una partición adaptativa del alfabeto en caracteres de frecuencia similar, permitiendo codificar subsecuencias con representaciones rápidas no comprimidas sin afectar las tasas generales de compresión.

La principal diferencia con su contraparte AP es que en este caso se remplaza el vector t por una matriz de p vectores de bits B_i , siendo i la respectiva partición que representa ese Bit vector, cada vector marcara con 1 la posición donde se encuentren valores de la partición i y con 0 donde se encuentren valores pertenecientes a otra partición. Gracias a este cambio se puede cambiar el Wavelet Tree usado por t por una estructura más ligera como lo es un arreglo de vectores de bits así mejorando los tiempos de ejecución de las operaciones fundamentales sobre la estructura en su implementación.

Se tiene que, para el ejemplo, *string* $s = \text{alabar_a_la_alabarda}$ la estructura será:

$$\begin{aligned}
 s &= \text{alabar_a_la_alabarda} \\
 \Sigma_1 &= \{a\}; \quad \Sigma_2 = \{l, _\}; \quad \Sigma_3 = \{b, r\}; \quad \Sigma_4 = \{d\} \\
 B_1 &: 10101001001010101001 \\
 B_2 &: 01000010110101000000 \\
 B_3 &: 00010100000000010100 \\
 B_4 &: 00000000000000000010 \\
 s_1 & \quad s_2 \quad s_3 \quad s_4 \\
 \text{aaaaaaaa} & \quad 1_1_1 \quad \text{brbr} \quad \text{d}
 \end{aligned}$$

Figura 6: Estructuras de datos ASAP para el *string* $s = \text{alabar_a_la_alabarda}$

Fuente: Diego Arroyuelo et al (Arroyuelo et al., 2024)

Respecto a las operaciones fundamentales, a continuación, se definen a partir de lo explicado por Arroyuelo et al (Arroyuelo et al., 2024) con $\alpha \in \Sigma$.

2.5.1 RANK

La operación Rank tiene un tiempo de ejecución promedio de $O(\log n)$ y se define como:

$$s.Rank_{\alpha}(i) = s_l.Rank_c(B_l.Rank_1(i)), \text{ donde } l = m.access(\alpha), c = m.Rank_l(\alpha)$$

2.5.2 SELECT

La operación Select tiene un tiempo de ejecución promedio de $O(1)$ y se define como:

$$s.Select_{\alpha}(i) = B_l.Select_1(s_l.Select_c(i)), \text{ donde } l = m.Access(\alpha), c = m.Rank_l(\alpha)$$

2.5.3 ACCESS

La operación Access tiene un tiempo de ejecución promedio de $O(\log^2 n)$, siendo esta una limitación de la estructura ASAP y se define como:

$$s.Access(i) = m.Select_l(s_l.Access(B_l.Rank_l(i))), \text{ donde } l = m.Access(\alpha), c = m.Rank_l(\alpha)$$

2.6 TRANSFORMACIÓN BURROW-WHEELER

La transformada de Burrows-Wheeler (BWT) publicado por Michael Burrows y David Wheeler en 1996 es un algoritmo de permutación reversible que transforma una cadena S de n caracteres formando las n rotaciones (desplazamientos cíclicos) de S , ordenándolas lexicográficamente y extrayendo el último carácter de cada una de las rotaciones formando una cadena L a partir de estos caracteres. (Burrows y Wheeler, 1994)

Dado un *string* $T = \text{BANANA\$}$ se tiene que al aplicar la transformación Burrow-Wheeler (BWT(T)), primero se aplican n rotaciones cíclicas sobre T :

0	B	A	N	A	N	A	\$
1	\$	B	A	N	A	N	A
2	A	\$	B	A	N	A	N
3	N	A	\$	B	A	N	A
4	A	N	A	\$	B	A	N
5	N	A	N	A	\$	B	A
6	A	N	A	N	A	\$	B

Figura 7: n rotaciones cíclicas sobre *string* T .

Fuente: Elaboración Propia

Luego, aplicando un ordenamiento léxico sobre las filas, en este caso considerando el símbolo “\$” como inicial.

SA	F						L	C
1	\$	B	A	N	A	N	A	0
6	A	N	A	N	A	\$	B	1
4	A	N	A	\$	B	A	N	
2	A	\$	B	A	N	A	N	
0	B	A	N	A	N	A	\$	4
5	N	A	N	A	\$	B	A	5
3	N	A	\$	B	A	N	A	

Figura 8: Ordenamiento léxico sobre la estructura generada del *string* T.

Fuente: Elaboración Propia

Por lo tanto, se tiene que la columna *L* correspondería al $BWT(T)=ABNN$AA$, respecto a las demás columnas se tiene $SA = [1, 6, 4, 2, 0, 5, 3]$, la cual contiene la posición de cada fila luego de realizado el ordenamiento léxico, además se tiene la columna $C = [0, 1, 4, 5]$ la cual almacena la información de las filas en las que se encuentra la primera ocurrencia de cada carácter según la columna *F*.

Respecto a *SA*, esta estructura nos permite definir la siguiente relación:

$$L[i] = T[SA[i] - 1]$$

Lo cual nos permite generar $BWT(T)$ en tiempo $O(n)$.

La operación de ordenamiento reúne las rotaciones con los mismos caracteres iniciales. Dado que los caracteres iniciales de las rotaciones están adyacentes a los caracteres finales, los caracteres consecutivos en *L* están adyacentes a cadenas similares en *S* por lo que tienden a generarse repeticiones o “*runs*” en la columna *L* o $BWT(T)$ (Burrows y Wheeler, 1994).

2.7 FM-COUNT

FM-count es un algoritmo de búsqueda que usa el concepto postulado por FM-Index, publicado por Ferragina et al (Ferragina et al., 2004), el cual es básicamente el resultado de $BWT(T)$, el cual usa 2 estructuras auxiliares, la estructura C la cual fue definida en el punto anterior junto a $Occ(BWT(T), c, pos) = |\{BWT(T)[j] : BWT(T)[j] = c \text{ y } 1 \leq j \leq pos\}|$, la cual permite contar las ocurrencias de un patrón P , encontrando los rangos del array de sufijos (SA) para T que comienzan con sufijos sucesivos de P en las iteraciones sucesivas del bucle (Grabowski et al., 2017), esta última estructura es lograda utilizando el algoritmo FM-count, el cual utiliza en conjunto la operación fundamental Rank sobre L junto a la estructura C .

Este algoritmo toma el patrón P y busca desde el ultimo carácter del patrón hasta el primer carácter utilizando las variables b y e como los extremos de la búsqueda tomando el valor inicial $b = 0$ y $e = n$, siendo n el tamaño del *string* T , luego estos tomaran los valores $b = C[P[i]] + rank_L(P[i], b)$ y $e = C[P[i]] + rank_L(P[i], e)$, en cada iteración, repitiendo esta asignación para b y e hasta recorrer por completo el patrón P , en cada iteración se agregara un carácter del patrón siendo siempre $e-b$ igual al número de ocurrencias del subpatron s que se busca en cada iteración del algoritmo hasta completar P encontrando $FM-Count(P)=e-b$ al final del recorrido. Debido a que para cada símbolo se debe realizar una búsqueda en BWT la operación FM-count posee un orden de $O(m \log \sigma)$ en tiempo de ejecución

Por ejemplo, dado el *string* $T=^BANANA\$, L=BWT(T)= ABNN\$AA$ y $C = [0, 1, 4, 5, 7]$, se desea encontrar el patrón $P=AN$, por lo que se inicia con $c = A$, siendo $c = P[i]$ para cada iteración, por lo que se tiene:

$$b = C[N] + rank_L(N, 0) = 5 + 0 = 5$$

$$e = C[N] + rank_L(N, 7) = 5 + 2 = 7$$

$$e - b = 7 - 5 = 2$$

Por lo tanto, se tiene que hay 2 ocurrencias de N en el *string* T , ahora pasamos los nuevos e y b a la siguiente iteración:

$$b = C[A] + rank_L(A, 5) = 1 + 1 = 2$$

$$e = C[A] + rank_L(A, 7) = 1 + 3 = 4$$

$$e - b = 4 - 2 = 2$$

Por lo que se tienen 2 ocurrencias del patrón $P=AN$.

2.8 RUN LENGTH ENCODING (RLE)

Es un método simple de compresión de datos que especifica la cantidad de veces que se repite un carácter, seguido por el valor del carácter. El objetivo es reducir el número de bits utilizados para representar un conjunto de datos. (AQA, 2020)

Por ejemplo, se tiene el *string* $T = \text{AABBAAADDDDAADDD}$, al usar RLE sobre este *string* quedaría como $\text{RLE}(T) = 2\text{A}2\text{B}3\text{A}4\text{D}3\text{A}3\text{D}$, el algoritmo para la conversión sería primero leer el *string* a convertir, luego por cada carácter contar sus repeticiones y agregar a una estructura el número de repeticiones seguido por el carácter y repetir este proceso hasta terminar la secuencia del *string*.

2.9 RLFM

RLFM (Run-Length FM-index) es una estructura de datos publicada en 2005 por Makinen V. y G. Navarro (Mäkinen y Navarro, 2005) desarrollada para textos, fue diseñada para ocupar poco espacio (idealmente cercano al del texto comprimido), además este permite búsquedas eficientes de patrones y es capaz de reproducir cualquier subcadena del texto, reemplazando así al texto original.

Este usa un espacio de $nH_k \log \sigma + O(n)$, aprovechando las regularidades en los arreglos de sufijos y en la transformación de Burrows-Wheeler del texto y su implementación práctica es competitiva con las mejores implementaciones existentes, mostrando ser competitivo tanto en espacio usado como de tiempo de ejecución.

2.10 VECTOR S18

S18 vector es una variante híbrida de S9 vector, el cual mantiene los casos presentados en S9 añadiendo casos para así lograr el manejo eficiente de *runs* de caracteres cortas, este hace uso de la estructura GAP para elementos que no generaran demasiados *runs* mientras que para otros casos usa la estructura antes definida RLE para textos repetitivos, así se logra optimizar el uso de ambas estructuras para cada caso. (Arroyuelo et al., 2018)

CAPÍTULO 3: PROPUESTA DE SOLUCION

Se proponen 4 estrategias de partición a probar sobre la estructura AR-RLFM, versión modificada de la estructura ASAP para la cual se deberán probar tanto distintos tipos de estructuras para sus vectores s y *bit vectors*, creados a partir del BWT de cada alfabeto con, esto debido a la alta repetición y el tamaño de los alfabetos a comparar, por lo que para cada alfabeto se utilizara su estructura BWT en vez del texto original, luego al momento de crear cada vector se intervendrá el proceso para particionar los textos de la manera que se necesita, a continuación se definirá cada estrategia de partición.

3.1 Estrategia A1

Esta estrategia de partición consiste en partir el alfabeto en p particiones, siendo $p = \lceil \log \frac{n}{2} \rceil$. La cantidad de símbolos por partición será de $\lceil \sigma/p \rceil$, y el mapeo usado para cada carácter del alfabeto será:

$$m(\alpha) = \lfloor \frac{\alpha}{\lceil \sigma/p \rceil} \rfloor$$

Con este tipo de mapeo directo no es necesario almacenar el mapeo en una estructura adicional, además dentro del subalfabeto su posición estará determinada por $\alpha \bmod (\lceil \sigma/p \rceil)$, por lo que el símbolo en cuestión puede ser encontrado rápidamente realizando esas operaciones matemáticas.

3.2 Estrategia A2

Al igual que con A1 esta estrategia de partición consiste en realizar un mapeo algebraico directo al dividir el alfabeto de tamaño σ en p particiones, en este caso el mapeo utilizado será:

$$m(\alpha) = \alpha \bmod (p)$$

Su posición dentro del subalfabeto se encuentra reenumerada a ser $\lfloor \alpha/p \rfloor$ con lo cual al igual que en la estrategia anterior tanto para encontrar su subalfabeto como su posición en este mismo será solo resolver estas operaciones matemáticas.

Cabe destacar que tanto para A1 como para A2 se reduce el espacio utilizado por la estructura ya que al utilizar este mapeo algebraico directo se deja de necesitar la estructura adicional para la búsqueda de símbolos en las subparticiones, además se espera una mejora en velocidad de consulta gracias a que su mapeo rápidamente calculable, lo cual además permite saltarse operaciones Rank en consultas.

3.3 Estrategia A3

Esta estrategia de partición consiste en utilizar “*dense partitioning*” (Barbay et al., 2014) el cual reparte el alfabeto en subalfabetos de acuerdo con la frecuencia del carácter en el texto, repartiendo los símbolos de modo que en la primera partición quedará el carácter con más repeticiones en el texto, luego en el siguiente subalfabeto los 2 símbolos con más repeticiones doblando el número de símbolos que contiene cada partición subsiguiente (Arroyuelo et al., 2024) ,así el mapeo se define:

$$m(\alpha) = \lfloor \log r(\alpha) \rfloor$$

siendo $r(\alpha)$ el ranking del carácter α de acuerdo con su frecuencia en el texto.

3.4 Estrategia A4

La estrategia de partición A4 consiste en una variación de A3, en este caso usando solo las cabeceras de los *runs* de *strings* presentes en el texto, usando el mismo tipo de mapeo que A3 “*dense partitioning*”.

3.5 MAPPING M

Para realizar los mapeos correspondientes de cada carácter, el almacenamiento de esta información, gestionar las operaciones a realizarse que involucren búsqueda, Rank, Select, Access, entre otras dentro de este *mapping*, por lo que se usara la estructura auxiliar que se provee desde la librería SDSL (Gog et al., 2014), de los cuales en específico se usara la estructura para gestionar el *mapping* “m”, esta provee tanto el ranking por frecuencia de los caracteres a la hora de dividirlos por “*dense partitioning*”, junto a lo que sería la opción de este mapeo ajustado por Simon Gog usando Wavelet Tree, así gestionando el mapeo para los casos de la estrategia A3 y A4 usando estructuras experimentalmente probadas y que proveen un manejo óptimo de esta estructura.

3.6 RANK-ASAP MODIFICADO

Para el caso de A1 y A2 a estos no contar con una estructura adicional para almacenar y gestionar su mapeo se elimina la necesidad de hacer uso de la estructura m junto con su *m.rank()* y *m.access()* por lo que la implementación de la operación Rank pasa de ser su versión usada por A3 junto a A4 a su versión modificada que usaran A1 y A2, usando su definición de l y n como corresponda para así evitar el uso de la estructura “m” junto al mapeo relacionado a esta estructura:

ASAP Rank (position, target):

```

1  l = m.map()
2  k = bitvectorsl.rank(position)
3  c = m.rank(m.get_pos_by_char(target), l)
4  return s(i,l).rank(k, c)

```

ASAP Rank Modificado (position, target) para A1:

```

1  l = ⌊ $\frac{target}{\sigma/p}$ ⌋;
2  n = target mod ([ $\sigma/p$ ])
3  k = (bitvectorsl.rank(position))
4  return s(i,l).rank(k, n)

```

ASAP Rank Modificado (position, target) para A2:

```

1  l = ⌊target mod p⌋
2  n = ⌊target/p⌋
3  k = (bitvectorsl.rank(position))
4  return s(i,l).rank(k, n)

```

Como se puede intuir, ya que el Rank modificado utilizado para A1 y A2 no utiliza *m.rank()* ni otras funciones relacionadas a esta estructura los tiempos de procesamiento, este debería ser potencialmente más rápido, pero cabe mencionar que esto puede ser opacado por la forma en la cual se particiona el alfabeto dando como resultado subalfabetos de tamaño igual en contraste con A3 Y A4 el cual organiza sus subalfabetos de manera para textos muy repetitivos.

3.7 ESTRUCTURA MAPPING “M” MODIFICADA

Para el caso de A4 al contar con solo las cabeceras de los *runs* se hace complicada la creación de los vectores de “*bit vectors*” y “*s vector*” por lo que para A4 se creara una estructura m modificada la cual cuenta con una combinación de ambos diccionarios de frecuencia, usando la frecuencia modificada con las cabeceras para la mayoría de la estructura y usando la frecuencia completa para los tamaños correspondientes a *m.sizes()* y así crear correctamente las estructuras “*s vector*” y “*bit vector*”.

3.8 ALTERNATIVAS A PROBAR

Las estructuras que se tendrán como alternativa a probar tendrá la siguiente estructura: ASAP (RLMN(AP) | RLMN(INT) | FKKP(AP) | FKKP(GMR)) (RLE | S18)

Siendo ASAP la estructura que se usara en general para todas las pruebas, luego para la estructura *s vector* se tienen las estructuras:

- RLMN(AP) and RLMN(INT): En este caso ambos usan lo postulado por Mäkinen y G. Navarro en (Mäkinen y Navarro, 2005) junto a AP como base y en el segundo caso se emplea un Wavelet tree directo.
- FKKP(AP) y FKKP(GMR): Se usa la estructura FKKP postulada e implementada por Fuentes-Sepúlveda et al en 2018 (Fuentes-Sepúlveda et al., 2018) , para luego usar como base AP y en el otro caso GMR, postulado por Golynski et al (Golynski et al., 2006).

Para el caso de los *bit vectors* se tienen las alternativas:

- RLE y S18: Estructura vector RLE y S18 los cuales según lo publicado por Arroyuelo et al (Arroyuelo et al., 2024) posee el mejor intercambio entre tamaño y velocidad para la operación fundamental Rank, la cual es fundamental para la consulta FM-Count que se probara a continuación.

Para la implementación de estas estructuras se usará la librería SDSL publicada en Github por Gog et al (Gog et al., 2014) en conjunto con la modificación provista por J. Fuentes-Sepúlveda agregando (Fuentes-Sepúlveda J. , 2022) estructuras adicionales a la librería SDSL y finalmente la librería para S18 provista por H. Larrañaga (Larrañaga, 2022) y finalmente se subirá el código fuente usado a github¹.

¹ Enlace a repositorio github del proyecto: [carlosrojasm/ASAP-RLFMindex \(github.com\)](https://github.com/carlosrojasm/ASAP-RLFMindex)

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

Para la validación de la solución se realizaron pruebas orientadas al rendimiento tanto en espacio como en velocidad de las estructuras generadas a través de las particiones A1, A2, A3 y A4, para esto se utilizaron los textos Coreutils, Einstein.en, Einstein.de, English y World_Leaders mediante los cuales se realiza una comparación contra el *baseline* actualmente vigente. Las pruebas consisten en la ejecución del algoritmo FM-Count para búsqueda de patrones usando las estructuras sobre los alfabetos previamente mencionados, así estos test tomarán en cuenta la memoria utilizada por las estructuras utilizadas en la búsqueda (vector *mapping* *m*, *bit vectors*, vector *s*) y en velocidad tomará en cuenta el tiempo que se demora en realizar la búsqueda de patrones mediante FM-Count, así tomando en cuenta el tiempo desde que comienza este proceso hasta haber devuelto el número de ocurrencias del patrón correspondiente.

Estos patrones son previamente creados a partir de los mismos alfabetos utilizados el cual toma patrones aleatorios de largo $m = \{4, 16, 32\}$ de estos hasta lograr un listado de 50000 patrones a encontrar.

Ya que estos patrones son extraídos directamente desde los alfabetos en los que se realiza la búsqueda, estos deben ser obligatoriamente encontrados y hacerlo de igual manera tanto para el *baseline* como para las estructuras creadas a través de las particiones A1, A2 o A3, solo variando la velocidad con la cual se realizan los Rank y las operaciones realizadas en caso de A1 y A2, con lo cual al cerciorarnos que todos los valores de los límites superiores e inferiores (*e* y *b*) sean iguales y los patrones sean encontrados se podrá decir que la implementación de estas particiones está correcta para pasar a las pruebas.

4.1 ESPECIFICACIONES DEL SISTEMA

Las pruebas fueron realizadas en un sistema con las siguientes especificaciones: HP Proliant Intel® Xeon® CPU E5-2630 2.30GHz, 6 núcleos, 15 MB de cache y 384GB de memoria RAM, mientras que las configuraciones del compilador son, g++ 7 con flags -std=c++17 -O3 -msse4.2 -DNDEBUG -lstdc++fs.

4.2 TEXTOS/ALFABETOS

Los alfabetos a utilizar serán extraídos desde Pizza&Chili corpus de P.Ferragina y G. Navarro (Ferragina y Navarro, 2005), de los cuales se extrae la siguiente información:

4.2.1 COREUTILS

Este texto contiene la concatenación de códigos fuentes, principalmente la recopilación de 9 versiones del paquete Coreutils5 sin considerar archivos binarios, junto con versiones 1.0.x y 1.1.x de Linux Kernel 6 ordenadas por fechas de creación.

Tabla 1: Información Coreutils

Fuente: Francisco R. (Riveros, 2022)

Información	Valor
Cantidad símbolos	93.319.681
Símbolos distintos	148.654
Cantidad runs BWT	2.540.091
Largo promedio runs BWT	36

2.5.1 EINSTEIN.EN

Este texto contiene la concatenación de las distintas versiones del artículo de Einstein, recopiladas desde Wikipedia en inglés

Tabla 2: Información Einstein.en

Fuente: Francisco R. (Riveros, 2022)

Información	Valor
Cantidad símbolos	159.119.879
Símbolos distintos	15.862
Cantidad runs BWT	96.953
Largo promedio runs BWT	1.640

2.5.1 EINSTEIN.DE

Contiene la concatenación de las distintas versiones del artículo de Einstein, recopiladas desde Wikipedia en alemán.

Tabla 3: Información Einstein.de

Fuente: Francisco R. (Riveros, 2022)

Información	Valor
Cantidad símbolos	29.970.946
Símbolos distintos	7.105
Cantidad runs BWT	40.349
Largo promedio runs BWT	742

2.5.1 ENGLISH

La biblioteca de Pizza & Chili Corpus contiene textos pseudo-reales que han sido sometidos a una mutación para agregar repetitividad de forma artificial a los textos originales de la biblioteca. En este caso, el texto "English" es una concatenación de una colección de textos mutados provenientes del Proyecto Gutenberg.

Tabla 4: Información English.001.2
Fuente: Francisco R. (Riveros, 2022)

Información	Valor
Cantidad símbolos	39.894.587
Símbolos distintos	89.753
Cantidad runs BWT	644.502
Largo promedio runs BWT	61

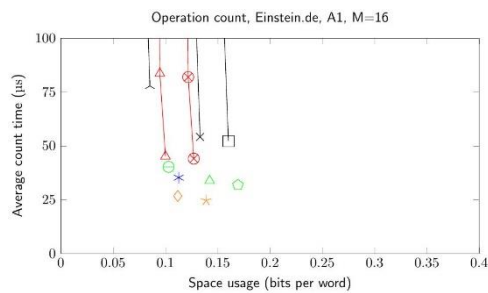
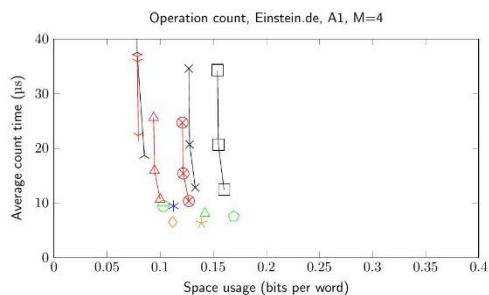
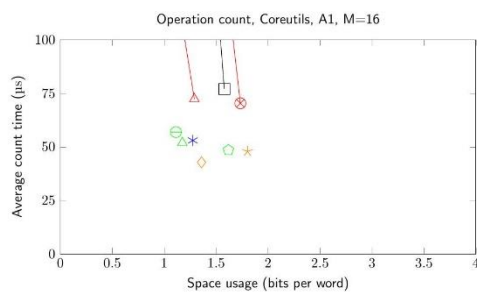
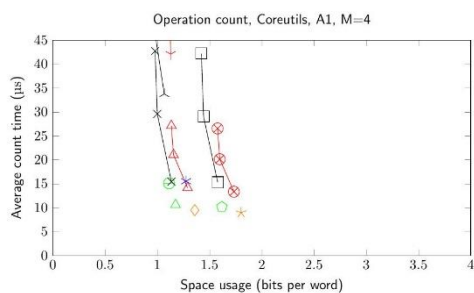
2.5.1 WORLD LEADERS

Este texto contiene la concatenación de todos los documentos PDF de la CIA desde el año 2006 hasta el 2009, los cuales fueron convertidos a texto utilizando el software pdftotext.

Tabla 5: Información World Leaders
Fuente: Francisco R. (Riveros, 2022)

Información	Valor
Cantidad símbolos	31.203.286
Símbolos distintos	29.176
Cantidad runs BWT	206.296
Largo promedio runs BWT	150

4.3 RESULTADOS ALTERNATIVA A1



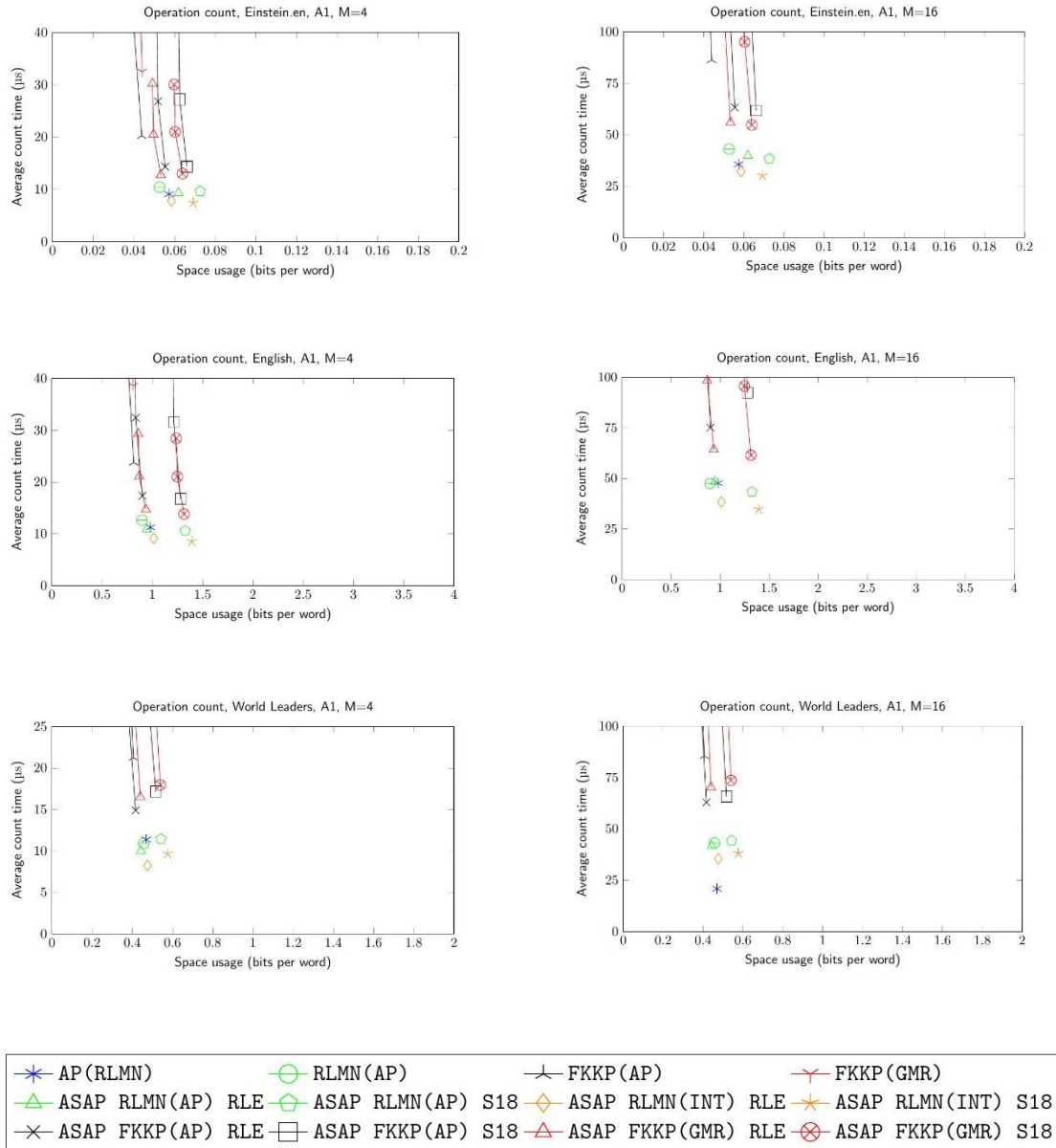


Figura 9: Gráficos Alternativa A1

Fuente: Elaboración Propia

Gráficamente se puede observar que tanto para $m=4$ como $m=16$, refiriéndonos con esto al tamaño de los patrones buscados mediante FM-Count, las alternativas que presentan mejor relación entre Tiempo y Memoria usada son ASAP RLMN (INT) RLE, ASAP RLMN (AP) RLE y ASAP FKKP (GMR) RLE, por lo que tomando estas 3 alternativas se realizara la comparación con AP (RLMN, RLMN) y RLMN (AP) los cuales funcionarían como el *baseline* para comprobar que tan optima es la estrategia de partición implementada, en este caso A1.

4.3.1 COMPARACIÓN A1 VS BASELINE

Coreutils

Tabla 6: Comparación A1 vs Baseline, Coreutils

Fuente: Elaboración Propia

Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	1.2754	15.5275	29.6793	53.0360
RLMN(AP)	1.1121	15.1003	27.3542	56.9387
ASAP RLMN(INT) RLE	1.3589	9.4838	21.7626	42.8100
ASAP RLMN(AP) RLE	1.7482	10.6090	28.7085	51.9567
ASAP FKPP(4,GMR) RLE	1.2900	14.1412	35.8576	72.3944
ASAP FKPP(16,GMR) RLE	1.1550	21.0650	55.5847	113.2447
ASAP FKPP(32,GMR) RLE	1.1342	27.1549	80.7738	169.0266

Einstein.de

Tabla 7: Comparación A1 vs Baseline, Einstein.de

Fuente: Elaboración Propia

Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	0.1127	9.4352	19.6475	35.2525
RLMN(AP)	0.1031	9.4557	20.2528	40.2516
ASAP RLMN(INT) RLE	0.1118	6.4876	12.7622	26.7132
ASAP FKPP(4,GMR) RLE	0.10004	10.6413	21.3308	45.1539
ASAP FKPP(16,GMR) RLE	0.0947	15.9172	37.7726	83.6508
ASAP FKPP(32,GMR) RLE	0.0939	25.6172	52.0576	116.6638

English

Tabla 8: Comparación A1 vs Baseline, English

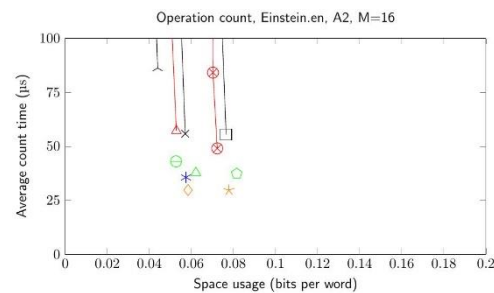
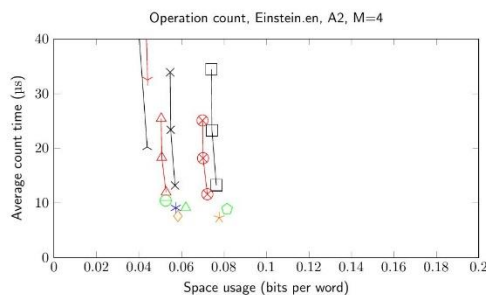
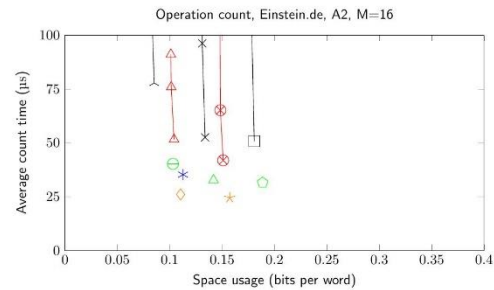
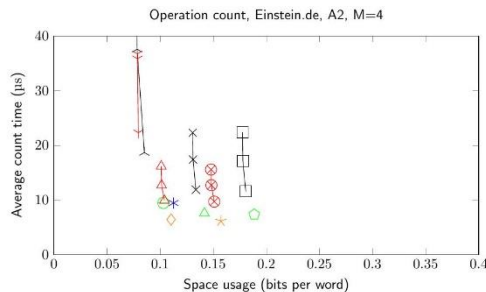
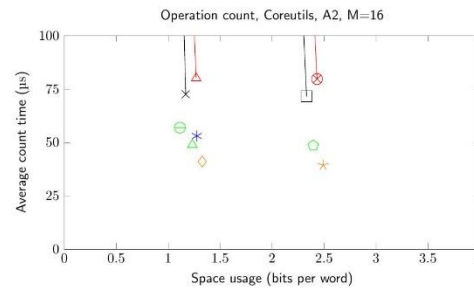
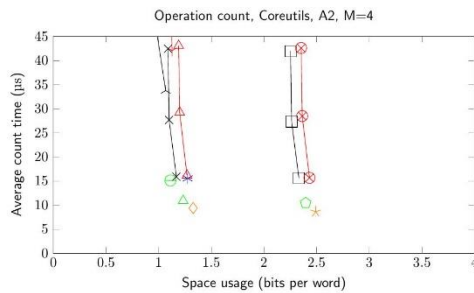
Fuente: Elaboración Propia.

Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	0.9807	11.2404	21.5004	47.5957
RLMN(AP)	0.8977	12.6341	25.5149	47.4585
ASAP RLMN(INT) RLE	1.0129	9.1204	18.7417	38.3089
ASAP RLMN(AP) RLE	0.9460	10.8931	22.7026	48.3697
ASAP FKPP(4,GMR) RLE	0.9353	14.7034	31.7021	64.3213
ASAP FKPP(16,GMR) RLE	0.8689	21.0651	47.6495	98.4053
ASAP FKPP(32,GMR) RLE	0.8588	29.3082	68.1847	142.6046

Para **Coreutils** se tiene que el espacio utilizado por ASAP RLMN(INT) RLE vs RLMN(AP) la cual es la que presenta menor cantidad de espacio usado para el *baseline*, se tiene que esta usa 1.22 veces la cantidad de espacio de RLMN(AP) pero en la velocidad de consulta este para $m=4$ es 1.59 más rápido que RLMN(AP) Y 1.33 veces más rápido con $m=16$, Para **Einstein.de**, con ASAP RLMN(INT) RLE, el consumo de espacio es 1.08 veces el de RLMN(AP), con tiempos de consulta 1.46 veces más rápidos para $m=4$ y 1.50 veces más rápidos para $m=16$. Con ASAP FKPP(4,GMR) RLE, el consumo es 1.09 veces el de RLMN(AP), y los tiempos de consulta son un 0.89 veces más lentos para $m=4$ y similarmente 0.89 veces más lento para $m=16$.

Para **English**, con ASAP RLMN(INT) RLE, el consumo de memoria es 1.12 veces el de RLMN(AP), con tiempos de consulta 1.39 veces más rápidos para $m=4$ y 1.24 veces más rápidos para $m=16$. Con ASAP RLMN(AP) RLE, el consumo de memoria es 1.05 veces el de RLMN(AP), y los tiempos de consulta son 1.16 veces más rápidos para $m=4$ y 0.98 veces más lento para $m=16$.

4.4 RESULTADOS ALTERNATIVA A2



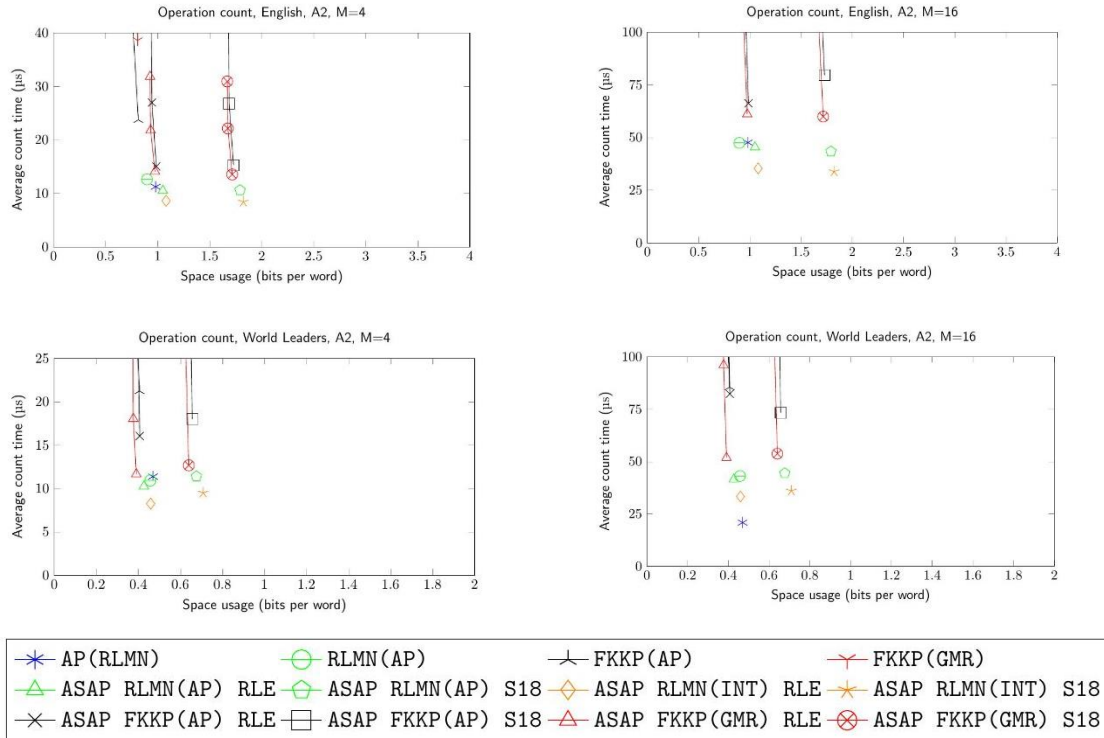


Figura 10: Gráficos Alternativa A2

Fuente: Elaboración Propia

Para este caso, al igual que para A1, se analizarán las alternativas ASAP RLMN(INT) RLE, ASAP RLMN(AP) RLE, ASAP FKPP(GMR) RLE ya que son las que analizando gráficamente poseen mejores resultados.

4.4.1 COMPARACIÓN A2 VS BASELINE

Coreutils

Tabla 9: Comparación A2 vs Baseline, Coreutils

Fuente: Elaboración Propia.

Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	1.2754	15.5275	29.6793	53.0360
RLMN(AP)	1.1121	15.1003	27.3542	56.9387
ASAP RLMN(INT) RLE	1.3278	9.3624	20.8092	41.0293
ASAP RLMN(AP) RLE	1.2326	10.9006	24.3997	48.9173
ASAP FKPP(4,GMR) RLE	1.2690	16.1785	38.3021	80.1493
ASAP FKPP(16,GMR) RLE	1.1997	29.2507	72.0616	159.4768
ASAP FKPP(32,GMR) RLE	1.1898	43.1040	115.0240	238.1239

Einstein.de

Tabla 10: Comparación A2 vs Baseline, Einstein.de

Fuente: Elaboración Propia.

Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	0.1127	9.4352	19.6475	35.2525
RLMN(AP)	0.1031	9.4557	20.2528	40.2516
ASAP RLMN(INT) RLE	0.1103	6.411367347	12.5041	26.0679
ASAP FKPP(4,GMR) RLE	0.1041	9.896591837	19.7167	51.6052
ASAP FKPP(16,GMR) RLE	0.1014	12.68834694	31.8888	75.9403
ASAP FKPP(32,GMR) RLE	0.1011	16.16593878	43.4627	91.0940

English

Tabla 11: Comparación A2 vs Baseline, Coreutils

Fuente: Elaboración Propia.

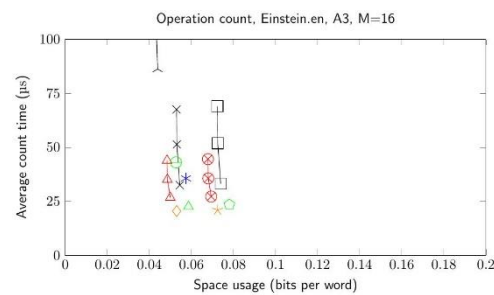
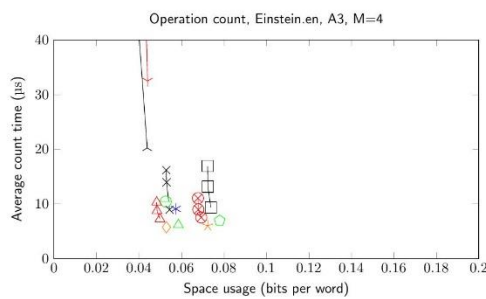
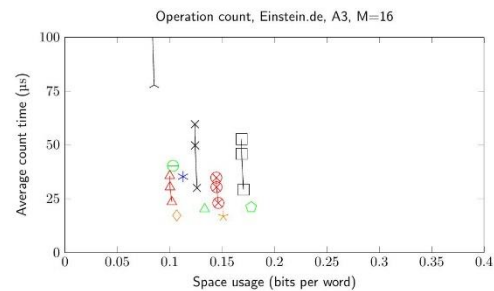
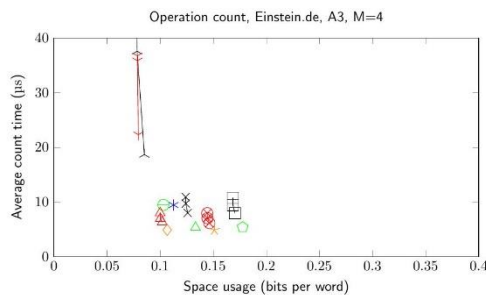
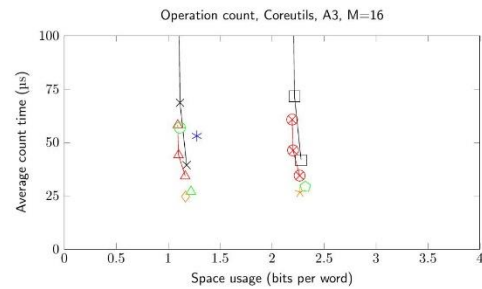
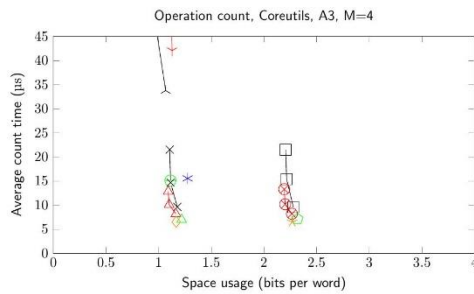
Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	0.9807	11.2404	21.5004	47.5957
RLMN(AP)	0.8977	12.6341	25.5149	47.4585
ASAP RLMN(INT) RLE	1.0814	8.6271	17.7506	35.3610
ASAP RLMN(AP) RLE	1.0500	10.5287	22.2563	45.4059
ASAP FKPP(4,GMR) RLE	0.9746	14.0812	30.1172	61.1206
ASAP FKPP(16,GMR) RLE	0.9329	21.8114	49.0385	111.1400
ASAP FKPP(32,GMR) RLE	0.9271	31.8409	71.7601	151.4332

Al comparar el espacio para **Coreutils** se tiene que el espacio utilizado por ASAP RLMN(INT) RLE vs RLMN(AP), la cual es la que presenta menor cantidad de espacio usado para el *baseline*, usa 1.19 veces la cantidad de espacio de RLMN(AP) pero en la velocidad de consulta este para $m=4$ es 1.62 más rápido que RLMN(AP) Y 1.39 veces más rápido con $m=16$,

Para **Einstein.de**, con ASAP RLMN(INT) RLE, el consumo de espacio es 1.07 veces el de RLMN(AP), con tiempos de consulta 1.47 veces más rápidos para $m=4$ y 1.54 veces más rápidos para $m=16$. Con ASAP FKKP(4,GMR) RLE, el consumo es 1.009 veces el de RLMN(AP), y los tiempos de consulta son un 5% más lentos para $m=4$ y 0.73 veces más lento para $m=16$.

Para **English**, con ASAP RLMN(INT) RLE, el consumo de memoria es 1.20 veces el de RLMN(AP), con tiempos de consulta 1.47 veces más rápidos para $m=4$ y 1.34 veces más rápidos para $m=16$. Con ASAP RLMN(AP) RLE, el consumo es 1.17 veces el de RLMN(AP), y los tiempos de consulta son 1.20 veces más rápidos para $m=4$ y 1.04 veces más rápidos para $m=16$.

4.5 RESULTADOS ALTERNATIVA A3



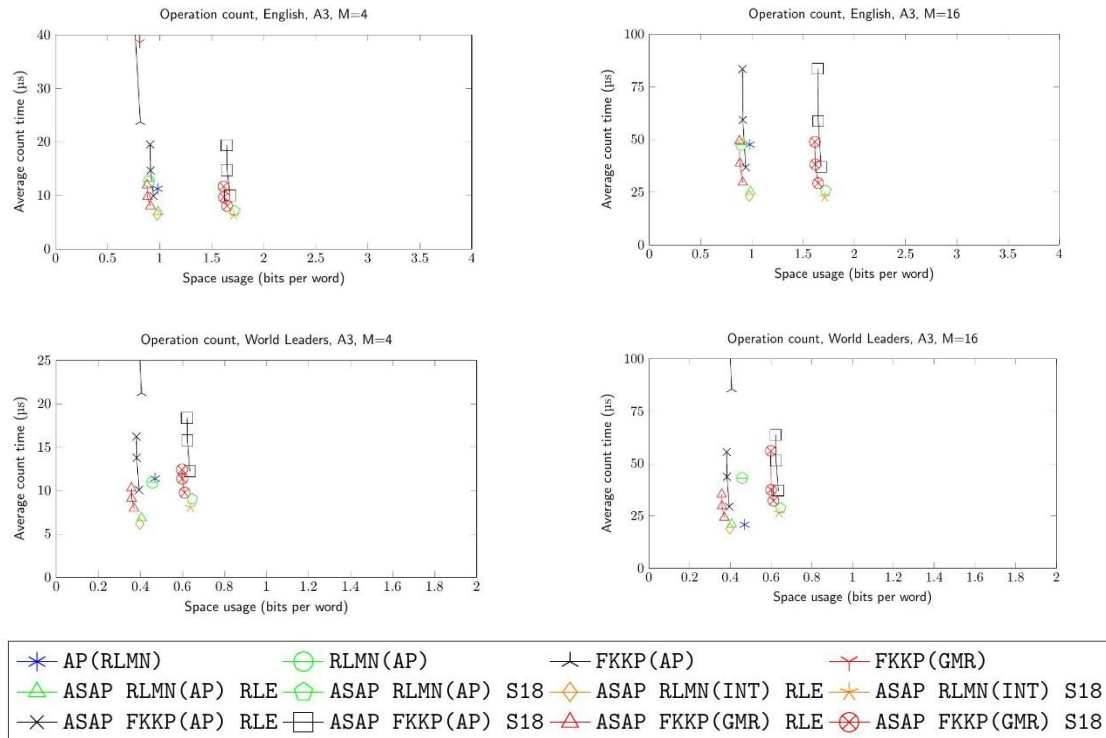


Figura 11: Gráficos Alternativa A3

Fuente: Elaboración Propia

4.5.1 COMPARACIÓN A3 VS BASELINE

Coreutils

Tabla 12: Comparación A3 vs Baseline, Coreutils

Fuente: Elaboración Propia

Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	1.2754	15.5275	29.6793	53.0360
RLMN(AP)	1.1121	15.1003	27.3542	56.9387
ASAP RLMN(INT) RLE	1.1556	6.8181	13.9393	26.3363
ASAP RLMN(AP) RLE	1.2083	6.8329	14.7717	27.4846
ASAP FKPP(4,GMR) RLE	1.1565	7.7753	16.5262	30.5971
ASAP FKPP(16,GMR) RLE	1.0937	9.8975	22.0919	42.5367
ASAP FKPP(32,GMR) RLE	1.0851	12.5237	29.2071	56.2592

Einstein.de**Tabla 13:** Comparación A3 vs Baseline, Einstein.de

Fuente: Elaboración Propia.

Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	0.1127	9.4352	19.6475	35.2525
RLMN(AP)	0.1031	9.4557	20.2528	40.2516
ASAP RLMN(INT) RLE	0.105501375	5.249591837	9.1507	17.5479
ASAP FKPP(4,GMR) RLE	0.101595992	5.979489796	11.1845	23.1535
ASAP FKPP(16,GMR) RLE	0.099815068	7.084897959	13.5963	30.8671
ASAP FKPP(32,GMR) RLE	0.996378292	7.86522449	16.8339	34.8621

English**Tabla 14:** Comparación A3 vs Baseline, English

Fuente: Elaboración Propia.

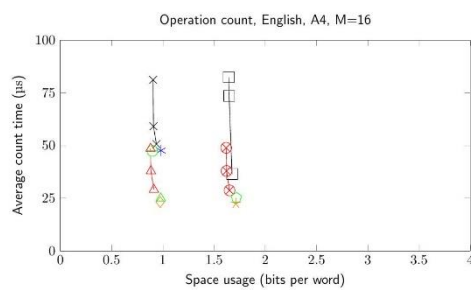
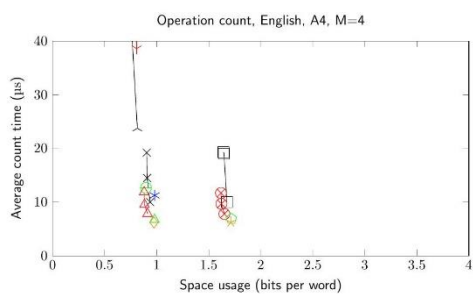
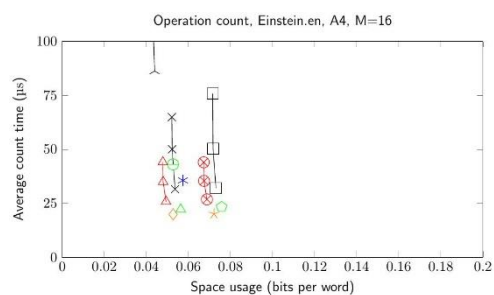
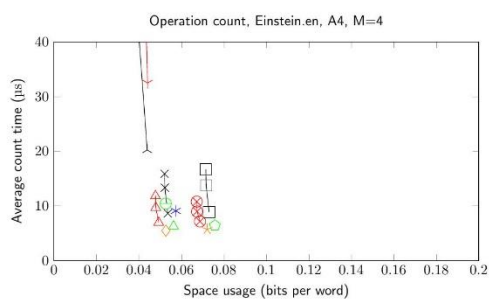
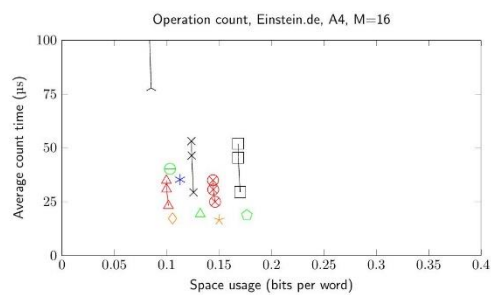
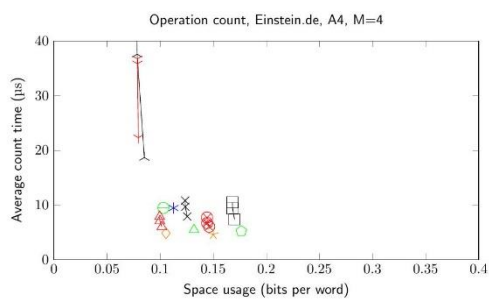
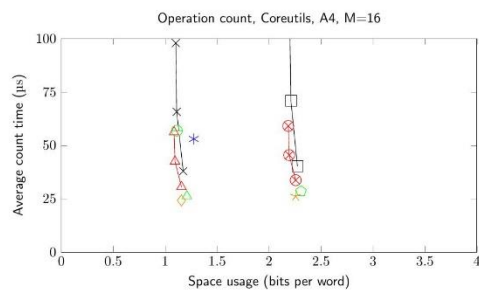
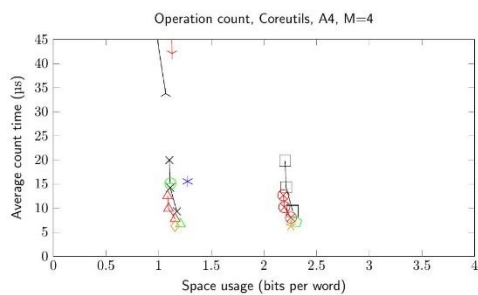
Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	0.9807	11.2404	21.5004	47.5957
RLMN(AP)	0.8977	12.6341	25.5149	47.4585
ASAP RLMN(INT) RLE	0.9746	6.4277	12.1723	23.3301
ASAP RLMN(AP) RLE	0.9794	7.1686	13.4987	26.0507
ASAP FKPP(GMR) RL	0.9127	7.8288	15.1883	29.0712
ASAP FKPP(16,GMR) RL	0.8838	9.5394	18.8936	37.8765
ASAP FKPP(32,GMR) RL	0.8801	11.8585	24.0159	48.5112

La estrategia de partición A3 en **Coreutils**, en el caso de ASAP RLMN(INT) RLE, el consumo es 1.04 veces el de RLMN(AP), con tiempos de consulta 2.22 veces más rápidos para m=4 y 2.16 veces más rápidos para m=16. Con ASAP RLMN(AP) RLE, el consumo es 1.09 veces el de RLMN(AP), y los tiempos de consulta son 2.19 veces más rápidos para m=4 y 2.07 veces más rápidos para m=16.

Para **Einstein.de**, con ASAP RLMN(INT) RLE, el consumo es 1.02 veces el de RLMN(AP), con tiempos de consulta 1.80 veces más rápidos para m=4 y 2.29 veces más rápidos para m=16. Con ASAP FKPP(4,GMR) RLE, el consumo es 0.98 veces el de RLMN(AP), y los tiempos de consulta son 1.58 veces más rápidos para m=4 y 1.73 veces más rápidos para m=16.

Para **English**, el consumo de espacio para ASAP RLMN(INT) RLE es 1.09 veces el de RLMN(AP), con tiempos de consulta 1.97 veces más rápidos para m=4 y 2.03 veces más rápidos para m=16. Con ASAP RLMN(AP) RLE, el consumo es 1.09 veces el de RLMN(AP), y los tiempos de consulta son 1.76 veces más rápidos para m=4 y 1.82 veces más rápidos para m=16.

4.6 RESULTADOS ALTERNATIVA A4



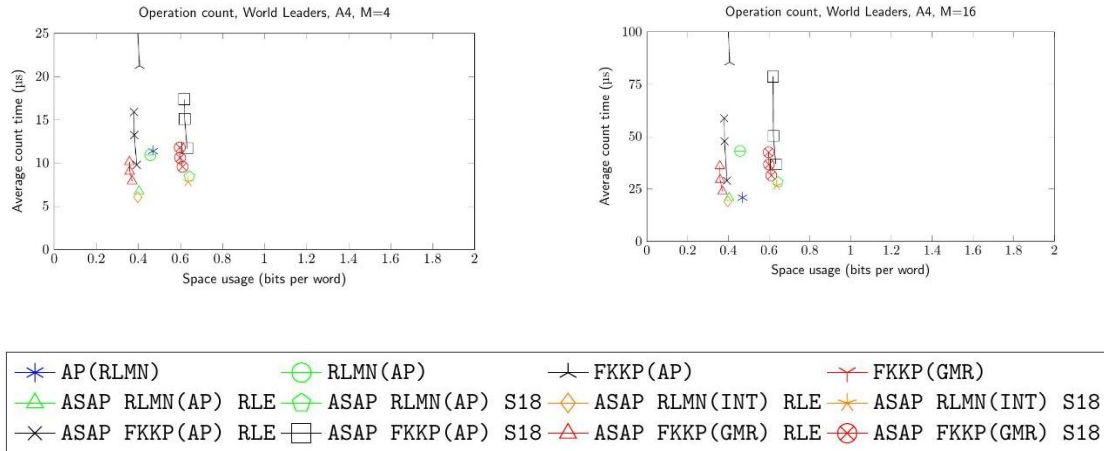


Figura 12: Gráficos Alternativa A4

Fuente: Elaboración Propia

4.6.1 COMPARACIÓN A4 VS BASELINE

Coreutils

Tabla 15: Comparación A4 vs Baseline, Coreutils

Fuente: Elaboración Propia.

Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	1.2754	15.5275	29.6793	53.0360
RLMN(AP)	1.1121	15.1003	27.3542	56.9387
ASAP RLMN(INT) RLE	1.1666	6.4786	13.1791	25.0918
ASAP RLMN(AP) RLE	1.2184	6.9129	14.5447	26.9566
ASAP FKKP(4,GMR) RLE	1.1636	8.0330	17.1502	32.9676
ASAP FKKP(16,GMR) RLE	1.0997	10.0892	23.1270	44.1274
ASAP FKKP(32,GMR) RLE	1.0912	12.8210	30.0218	58.0936

Einstein.de

Tabla 16: Comparación A4 vs Baseline, Einstein.de

Fuente: Elaboración Propia.

Alternativas	Bits/palabra	Tiempo [μs] m=4	Tiempo [μs] m=8	Tiempo [μs] m=16
AP(RLMN, RLMN)	0.1127	9.4352	19.6475	35.2525
RLMN(AP)	0.1031	9.4557	20.2528	40.2516
ASAP RLMN(INT) RLE	0.106700136	4.818918367	8.8954	17.5072
ASAP FKKP(4,GMR) RLE	0.101972357	6.167897959	11.7081	23.3469
ASAP FKKP(16,GMR) RLE	0.100204244	6.963326531	14.4583	30.4191
ASAP FKKP(32,GMR) RLE	0.100033412	7.995489796	16.7423	35.6318

English**Tabla 17:** Comparación A4 vs Baseline, English

Fuente: Elaboración Propia.

Alternativas	Bits/palabra	Tiempo [μ s] m=4	Tiempo [μ s] m=8	Tiempo [μ s] m=16
AP(RLMN, RLMN)	0.9807	11.2404	21.5004	47.5957
RLMN(AP)	0.8977	12.6341	25.5149	47.4585
ASAP RLMN(INT) RLE	0.9765	6.5767	12.0572	23.1932
ASAP RLMN(AP) RLE	0.9857	7.1187	13.4518	25.9298
ASAP FKPP(4,GMR) RLE	0.9119	8.1891	15.6098	29.9162
ASAP FKPP(16,GMR) RLE	0.8832	9.7231	19.4633	38.6355
ASAP FKPP(32,GMR) RLE	0.8795	11.9507	24.2169	49.3865

En el caso de la estrategia de partición A4, ASAP RLMN(INT) RLE generalmente proporciona la relación costo-beneficio más competitiva entre nuestros métodos. Consume solo 0.11 bits por palabra para el texto Einstein.de, 0.98 bits por palabra para textos en inglés, y 1.17 bits por palabra para Coreutils.

En cuanto al a las comparaciones con respecto a las demás estructuras se tiene que:

Coreutils: en este escenario, ASAP RLMN(INT) RLE consume 1.05 veces el espacio de RLMN(MN), mientras que realiza las consultas 2.33 veces más rápido (m = 4) y 2.27 veces más rápido (m = 16). ASAP FKPP(GMR) RLE consume 0.98 veces el espacio de RLMN(MN), mientras que realiza las consultas 1.50 veces más rápido (m = 4) y 1.30 veces más rápido (m = 16).

Einstein.de: ASAP RLMN(INT) RLE ocupa 1.03 veces el espacio de RLMN(MN) y realiza las consultas 1.96 veces más rápido (m = 4) y 2.30 veces más rápido (m = 16), mientras que para ASAP FKPP(GMR) RLE ocupa 0.99 veces el espacio de RLMN(MN) y realiza las consultas 1.53 veces más rápido (M = 4) y 1.72 veces más rápido (M = 16)

English: ASAP RLMN(INT) RLE requiere 1.09 veces el espacio de RLMN(MN) y realiza las consultas 1.92 veces más rápido (m = 4) y 2.05 veces más rápido (m = 16), mientras que para ASAP FKPP(GMR) RLE utiliza 0.98 veces el espacio de RLMN(MN), realizando las consultas 1.30 veces más rápido (M = 4) y 1.23 veces más rápido (M = 16).

CAPÍTULO 5: CONCLUSIONES

En este documento se ha logrado el diseño, implementación y evaluación de 4 estrategias de particionamiento de alfabeto para estructuras de datos comprimidas, las cuales permiten realizar las operaciones fundamentales de búsqueda Rank, Select y Access de manera eficiente para textos largos y repetitivos. Las estrategias y alternativas presentadas fueron evaluadas en termino de costo-beneficio utilizando como métrica el espacio en memoria utilizado por las estructuras y la velocidad de ejecución en las consultas (FM-COUNT).

Los resultados obtenidos a partir de los experimentos realizados demuestran que las soluciones propuestas, en particular la alternativa ASAP RLMN(INT) RLE junto con la estrategia de partición A4, ofrecen mejoras significativas tanto en la memoria usada como en la velocidad de sus consultas en comparación con las estructuras pertenecientes al *baseline*. Por ejemplo, en el caso del texto English, esta estrategia en conjunto con la alternativa antes mencionada consume solo 0.9765 [bits/palabra] en promedio, lo cual representa una eficiencia sobresaliente en términos de compresión y a la vez ofreciendo una velocidad de operaciones sumamente destacable. Las pruebas realizadas en textos de diferente tipo y tamaño, como Coreutils, Einstein.de, English, entro otros, han demostrado que, si bien A4 es la estrategia con mejor rendimiento, en general todas las estrategias propuestas proponen un intercambio eficiente en termino de memoria usada vs velocidad de ejecución de consultas usando las alternativas de estructuras comprimidas indicadas al problema. Esto es especialmente relevante para aplicaciones que usen textos repetitivos tales como el almacenamiento de URLs, control de versiones tales como GitHub, entre otros.

Los objetivos planteados al inicio de este proyecto han sido validados con éxito ya que se han propuesto las 4 estrategias de particionamiento los cuales han probado ser experimentalmente competitivos con el *baseline* propuesto tanto en memoria usada como en la velocidad validando así la eficiencia de estas soluciones, las cuales en conjunto con las estructuras de datos comprimidas probadas se han transformado en múltiples soluciones viables, con diferentes ventajas o desventajas en velocidad o espacio en memoria. Además de lo anterior mediante los experimentos realizados se ha logrado validar que las alternativas y estructuras usadas logran manejar de manera eficiente tanto el almacenado de los textos grandes y repetitivos como la recuperación de este. Finalmente, respecto al punto de los objetivos cumplidos, mediante la implementación de estas estrategias de partición mediante el uso del lenguaje C++ y librerías usadas como standard en la industria, lo cual significa que están completamente probadas y es uso de esta es confiable, se logró generar el código necesario y posteriormente publicarlo en la plataforma Github para que este quedara disponible para su futuro uso y mejora.

El enfoque metodológico usado para este proyecto ha girado en torno al proceso de diseño, implementación y evaluación de estrategias de particionamiento para alfabetos utilizando distintas estructuras de datos comprimidas en las cuales fuera posible utilizar las

operaciones fundamentales de búsqueda Rank, Select y Access, de estas se destacan las alternativas que para la mayoría de las estrategias de particionamiento fueron las mejores, ASAP RLMN(INT) RLE, ASAP RLMN(AP) RLE y ASAP FKPP(GMR) RLE, de las cuales si bien ASAP RLMN(INT) RLE fue la mejor en la mayoría de los casos las otras alternativas también se posicionan como estructuras competitivas frente al *baseline*, en el caso de la alternativa ASAP RLMN(AP) RLE, esta usa una mayor cantidad de espacio en memoria para su funcionamiento pero ofrece resultados en velocidad de consulta sumamente destacables, en el caso de competir en términos de espacio la alternativa ASAP FKPP(GMR) RLE se vuelve sumamente competitiva en términos tanto de espacio como de velocidad, este al tener 3 opciones (4,16,32) para el uso de la estructura FKPP, ofrece distintos *trade-off* entre la velocidad de la consulta y la memoria usada por lo que esta permite elegir una variación de la misma que supera al *baseline* en el apartado de espacio usado mientras que otra opción se vuelve competitiva con este en velocidad, así se concluye que estas 3 alternativas de estructuras de datos comprimidas usadas en conjunto con las estrategias de partición de alfabeto presentadas, en especial A4, son las mejores al enfrentarse al *baseline* propuesto.

Como se había postulado al definir las estrategias de particionamiento A1 y A2, estas al poseer una forma de realizar el *mapping* matemático rápido de sus caracteres a las estructuras de s_i y de bit vectors, se omite tanto la creación como el uso de la estructura de *mapping* "M", cabe recordar que se teorizaba que los resultados de estas alternativas podrían ser mucho más rápidos y usando un menor espacio al no tener que utilizar esta estructura adicional, pero según lo obtenido experimentalmente estas alternativas si bien el espacio usado en memoria se redujo, no lo hizo de sobremanera y comparado con la *baseline* este incluso subió su memoria usada por las estructuras, luego comparando la velocidad de consulta utilizando las alternativas ASAP RLMN(INT) RLE y ASAP RLMN(AP) RLE este se vuelve competitivo contra el *baseline* postulado ya que consigue que el *trade-off* entre la cantidad extra de espacio comparándolo con el *baseline*, pero comparando A1 y A2 con las estrategias A3 o A4 estas difieren totalmente con lo postulado inicialmente, estas aun usando su *mapping* matemático directo y rápido al realizar las consultas correspondientes a FM-COUNT este presenta tiempos de consulta peores respecto a A3 y A4, aun saltándose *mapping* "M", por lo que estarían haciendo una cantidad menor de operaciones rank al ejecutar la consulta FM-COUNT, poseen resultados peores a los esperados, esto resultó así ya que esta estrategia divide el alfabeto de cada texto en subalfabetos distintos a los usados en la estrategia A3 y A4, esta al no usar la estructura adicional "M" no cuenta con el ordenamiento por ranking de carácter en sus subalfabetos, sino que estos se presentan de igual tamaño ya que los caracteres son mapeados a cada uno de estos usando el valor de este carácter pasado por la operación matemática $m(\alpha)$ respectiva a A1 y A2 lo que provoca que el ordenamiento se realice de manera que la cantidad de caracteres por subalfabeto sea de un tamaño similar entre cada uno de ellos, lo cual repercute que al momento de realizar consultas sobre sus estructuras las búsquedas se tornen mucho más engorrosas respecto a lo usado por las estrategias A3 y A4, las cuales al realizar su ordenamiento mediante un ranking del cual luego se generaran subalfabetos de tamaños crecientes (primero de un carácter, luego de 2, luego de 4, etc), en los cuales por ejemplo el carácter con más repeticiones quedara en el alfabeto $\sum_1$ sin ningún otro

carácter por lo que realizar operaciones sobre este será mucho más rápido que el acceder a las estructuras que se generan tanto para A1 como para A2 sobre todo si se tratan de textos como los estudiados en este documento los cuales poseen gran cantidad de repetición, por lo cual respecto a este punto se puede concluir que tanto A1 como A2 en comparación con A3 y A4 poseen un rendimiento inferior, pero quizá si es probado en textos en la cual la frecuencia de los caracteres es similar pueda tener un rendimiento superior, cabe destacar que para esto igualmente se debe tener en cuenta que para un texto con estas características su entropía H_0 tendría un valor cercano a $\lg \sigma$, siendo este su valor máximo por lo que este sería un texto poco compresible por lo que difiere totalmente con el texto objetivo de este documento, dejándolo como un experimento posible pero no recomendado para el contexto al que se orienta este tema.

5.1 Principales Contribuciones y Aplicaciones

Las principales contribuciones de este trabajo es la propuesta, implementación y validación de las estrategias de particionamiento de alfabeto A1, A2, A3 y A4 las cuales mejoran la eficiencia de estructuras de datos comprimidas, como se ha dicho con anterioridad, optimizando el espacio usado y la velocidad de consulta. Las aplicaciones potenciales de este trabajo son amplias, desde el manejo de grandes volúmenes de datos en sistemas de almacenamiento y recuperación, hasta la mejora de rendimiento en aplicaciones que requieran acceso rápido a grandes textos comprimidos.

El impacto de este trabajo se refleja en la mejora significativa en la eficiencia de sistemas que manejen grandes volúmenes de datos repetitivos, las soluciones planteadas pueden contribuir tanto reduciendo los costos de almacenamiento al mejorar el espacio usado por las estructuras comprimidas, como en la mejora de los sistemas de recuperación de la información desde el almacenado, lo cual se traduce en reducción de costos operativos en procesamiento de datos, este impacto es especialmente valioso para organizaciones que manejen grandes volúmenes de datos con repetición, tales como pueden ser los sistemas de control de versiones o bibliotecas virtuales. Además, para desarrolladores y profesionales en el área de compresión de datos este trabajo proporciona nuevas herramientas y enfoques que pueden ser adaptados y mejoras para distintas aplicaciones futuras.

5.2 Trabajo Futuro

Si bien los resultados obtenidos son competitivos y han demostrado mantener velocidades de consulta altas para las estructuras propuestas, hay que recordar que estas fueron hechas en un entorno experimental, tomando 5 textos los cuales poseen condiciones idóneas para el funcionamiento de estos, ósea textos largos y con gran repetición de caracteres, por lo que una limitación de estos es que no han sido probados en un entorno practico con uso frecuente y concurrente, por lo cual sería interesante en futuras investigaciones abordar pruebas más intensas tanto en entornos que favorezcan a estos como en algunos en los que

se encuentre en desventaja para tener una comparativa de su rendimiento en estos entornos, siendo lo más probable que al compararla con un *baseline* óptimo para esas condiciones este se vea sobrepasado en rendimiento, debido principalmente al comportamiento de la entropía H_0 cercana a la máxima que adquiere en circunstancias con baja repetición, pero sería interesante ver el desempeño por ejemplo de las estrategias A1 y A2 las cuales al dividir el alfabeto en subalfabetos de igual número de caracteres podría mejorar su rendimiento en este tipo de entornos, en cualquier caso sería interesante obtener datos precisos de cuál sería su memoria usada y velocidad de consulta tanto en contextos prácticos e intensivos como en contextos desventajosos para estos. Además de los entornos antes nombrados igualmente sería bastante útil para diferentes industrias la realización de pruebas tanto de las estrategias de partición como de las estructuras de datos comprimidas en otros tipos de datos repetitivos más allá de textos, tales como imágenes o secuencias de ADN.

Algunas consideraciones a tener en cuenta son por ejemplo las estructuras comprimidas usadas inicialmente por la implementación en C++, en este caso se utiliza BWT y se almacena la estructura L resultante correspondiente a $L = BWT(T)$, siendo T el texto en cuestión (Coreutils, Einstein.de, etc), junto al vector C y otras estructuras auxiliares las cuales son clave para el funcionamiento de la implementación, estas deben generarse aparte antes de ejecutar la implementación de ASAP junto a las alternativas tanto de las estrategias de partición como las alternativas de estructuras de datos para s_i y los *bit vector*, por lo cual esto debe ser tomado en cuenta a la hora de utilizar la implementación, además este genera resultados directo en la carpeta que se crea luego de haber generado los archivos antes mencionados resultantes de BWT, generando así 2 carpetas para los resultados de la memoria usada por alternativa y otra para los resultados de la consulta FM-Count realizada, como se explicó en el punto de “Validación de la solución” para ejecutar esta consulta se debe haber generado desde el texto usado conjuntos de patrones de distinto tamaño con anterioridad, al obtener los resultados de tiempo y memoria usada en las consultas estas deben ser procesadas por un programa aparte para obtener resultados promedio como los usados en este documento, por lo que una de las posibles mejoras, si se desea seguir este proyecto, sería que el ciclo de pruebas para alternativas de estructuras y estrategias de partición incluyera la verificación de que estos archivos existiesen y en caso de que alguno faltase estuviese incluido en la ejecución la creación de estos, y luego a la vez de tener como salida los resultados de los experimentos dar la opción del post procesado de estos para tener datos legibles en vez de solo datos brutos e incluso se puede considerar el incluir directamente el graficado de estos datos para tener el ciclo completo de las pruebas y comparativa listo. Además, también es recomendable incluir la opción de cambio de estrategia de particionamiento en los *inputs* del programa para pruebas, actualmente se encuentran las implementaciones completas de esta en el código de este proyecto, pero para la elección entre una estrategia o otra se realiza directamente en los archivos del programa por lo cual para su fácil prueba sin necesidad de recompilación para cada estrategia se recomienda realizar esta mejora.

Para futuro trabajo orientado a la prueba de nuevas estructuras, un paso posible y recomendable sería la implementación de ASAP junto a las 4 estrategias de partición propuestas sobre las estructuras usadas como *baseline* comparativo en este proyecto, ósea AP(RLMN, RLMN) Y RLMN(AP), los cuales igualmente se encuentran incluidos en la librería SDSL pero por falta de alcance en el proyecto y como estos fueron usados como *baseline* se dejaron fuera por el momento, pero como trabajo futuro esta orientación parece la indicada para la continuación de este trabajo ya que al contar actualmente con un rendimiento tal que fue elegido como *baseline* para este trabajo, en conjunto con las nuevas estrategias postuladas estas tienen el potencial de tener resultados aun mejores a los ya logrados con las estructuras ASAP RLMN(INT) RLE, ASAP RLMN(AP) RLE en conjunto a la estrategia de particionamiento A4.

Adicionalmente a lo antes nombrado igualmente sería útil realizar como trabajo futuro la realización de herramientas o librerías que implementen directamente estas estrategias junto a ASAP con el fin de facilitar su uso y adopción en la industria para así proporcionar un impacto practico en la industria.

REFERENCIAS BIBLIOGRÁFICAS

- (AQA, 2020) AQA. (2020). *Teaching guide: Run-length encoding (RLE)*. Retrieved 25 de 05 de 2024, from GCSE Computer Science specification (8525): <https://filestore.aqa.org.uk/resources/computing/AQA-8525-TG-RLE.PDF?ref=aspires.cc>
- (Arroyuelo et al., 2024) Arroyuelo, D., Carmona, G., Larrañaga, H., Riveros, F., Rojas-Morales, C. E., y Sepúlveda, E. (2024). *Engineering Rank/Select Data Structures for Big-Alphabet Strings*. arXiv:2305.14461v2.
- (Arroyuelo et al., 2018) Arroyuelo, D., Oyarzún, M., González, S., y Sepúlveda, V. (2018). Hybrid compression of inverted lists for reordered document collections. *Information Processing & Management*, 54(6), 1308-1324.
- (Arroyuelo y Sepúlveda, 2019) Arroyuelo, D., y Sepúlveda, E. (2019). A practical alphabet-partitioning rank/select data structure. En *International Symposium on String Processing and Information Retrieval* (pp. 452-466). Cham: Springer International Publishing.
- (Barbay et al., 2014) Barbay, J., Claude, F., Gagie, T., Navarro, G., y Nekrich, Y. (2014). Efficient fully-compressed sequence representations. *Algorithmica*, 69(1), 232-268.
- (Barbay et al., 2010) Barbay, J., Gagie, T., Navarro, G., y Nekrich, Y. (2010). Alphabet Partitioning for Compressed Rank/Select and Applications. En *International Symposium on Algorithms and Computation* (pp. 315-326). Berlin, Heidelberg: Springer Berlin Heidelberg.
- (Burrows y Wheeler, 1994) Burrows, M., y Wheeler, D. (1994). A block-sorting lossless data compression algorithm. *SRC Research Report*.
- (Ferragina et al., 2004) Ferragina, P., Manzini, G., Mäkinen, V., y Navarro, G. (2004). An alphabet-friendly FM-index. En *International Symposium on String Processing and Information Retrieval* (pp. 150-160). Berlin, Heidelberg: Springer Berlin Heidelberg.
- (Ferragina y Navarro, 2005) Ferragina, P., y Navarro, G. (2005). *Pizza&Chili Corpus—Compressed indexes and their testbeds*. <https://pizzachili.dcc.uchile.cl/>
- (Fuentes-Sepúlveda J. , 2022) Fuentes-Sepúlveda, J. (2022). *Github*. RI runs library: <https://github.com/jfuentess/sdsl-lite>
- (Fuentes-Sepúlveda et al., 2018) Fuentes-Sepúlveda, J., Kärkkäinen, J., Kosolobov, D., y Puglisi, S. (2018). Run compressed rank/select for large alphabets. En *Data Compression Conference* (pp. 315-324). IEEE.
- (Gog et al., 2014) Gog, S., Beller, T., Moffat, A., y Petri, M. (2014). From Theory to Practice: Plug and Play with Succinct Data Structures. En *13th International Symposium, SEA 2014, Copenhagen, Denmark, June 29–July 1, 2014. Proceedings 13* (pp. 326-337). Springer International Publishing. Retrieved 01 de 06 de 2024, from <https://github.com/simongog/sdsl-lite>
- (Golynski et al., 2006) Golynski, A., Munro, J. I., y Rao, S. S. (2006). Rank/select operations on large alphabets: a tool for text indexin. En *SODA* (Vol. 6, págs. 368-373).

- (Grabowski et al., 2017) Grabowski, S., Raniszewski, M., y Deorowicz, S. (2017). FM-index for dummies. En *13th International Conference, BDAS 2017, Ustroń, Poland, May 30-June 2, 2017, Proceedings 13* (pp. 189-201). Springer International Publishing.
- (Larrañaga, 2022) Larrañaga, H. (2022). *Github*. S18 library:
<https://github.com/HectorxH/Modificacion-S18>
- (Mäkinen y Navarro, 2005) Mäkinen, V., y Navarro, G. (2005). Succinct suffix arrays based on run-length encoding. En *16th Annual Symposium, CPM 2005, Jeju Island, Korea, June 19-22, 2005. Proceedings 16* (pp. 45-56). Springer Berlin Heidelberg.
- (Navarro, 2014) Navarro, G. (2014). Wavelet trees for all. *Journal of Discrete Algorithms*, 25, 2-20.
- (Riveros, 2022) Riveros, F. (2022). ESTRUCTURA DE DATOS COMPRIMIDAS PARA BÚSQUEDAS EN TEXTOS MUY REPETITIVOS Y ALFABETOS GRANDES (Tesis de Pregado). UTFSM, Santiago.

ANEXOS

ANEXO A: RESULTADOS COMPLETOS A1

Tabla 18 : Resultados A1 – Coreutils m=4,m=8

Fuente: Elaboración propia.

m	estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	1.39846856	9.62E-06
4	ASAP AP(RLMN) S18	1.84133608	9.01E-06
4	ASAP FKKP(AP) RLE	0.97905495	4.27E-05
4	ASAP FKKP(AP) RLE	0.99987157	2.96E-05
4	ASAP FKKP(AP) RLE	1.13488429	1.54E-05
4	ASAP FKKP(AP) S18	1.42192248	4.22E-05
4	ASAP FKKP(AP) S18	1.44273909	2.90E-05
4	ASAP FKKP(AP) S18	1.57775181	1.53E-05
4	ASAP FKKP(GMR) RLE	1.1341749	2.72E-05
4	ASAP FKKP(GMR) RLE	1.15499152	2.11E-05
4	ASAP FKKP(GMR) RLE	1.29000424	1.41E-05
4	ASAP FKKP(GMR) S18	1.57704242	2.66E-05
4	ASAP FKKP(GMR) S18	1.59785904	2.01E-05
4	ASAP FKKP(GMR) S18	1.73287176	1.34E-05
4	ASAP RLMN(AP) RLE	1.17481919	1.06E-05
4	ASAP RLMN(AP) S18	1.61768671	1.02E-05
4	ASAP RLMN(INT) RLE	1.35891926	9.48E-06
4	ASAP RLMN(INT) S18	1.80178678	8.99E-06
8	ASAP AP(RLMN) RLE	1.39846856	2.05E-05
8	ASAP AP(RLMN) S18	1.84133608	1.97E-05
8	ASAP FKKP(AP) RLE	0.97905495	0.00013037
8	ASAP FKKP(AP) RLE	0.99987157	8.05E-05
8	ASAP FKKP(AP) RLE	1.13488429	3.89E-05
8	ASAP FKKP(AP) S18	1.42192248	0.0001293
8	ASAP FKKP(AP) S18	1.44273909	7.96E-05
8	ASAP FKKP(AP) S18	1.57775181	3.88E-05
8	ASAP FKKP(GMR) RLE	1.1341749	8.08E-05
8	ASAP FKKP(GMR) RLE	1.15499152	5.56E-05
8	ASAP FKKP(GMR) RLE	1.29000424	3.59E-05
8	ASAP FKKP(GMR) S18	1.57704242	7.97E-05
8	ASAP FKKP(GMR) S18	1.59785904	5.72E-05
8	ASAP FKKP(GMR) S18	1.73287176	3.52E-05
8	ASAP RLMN(AP) RLE	1.17481919	2.87E-05
8	ASAP RLMN(AP) S18	1.61768671	2.43E-05
8	ASAP RLMN(INT) RLE	1.35891926	2.18E-05
8	ASAP RLMN(INT) S18	1.80178678	2.14E-05

Tabla 19: Resultados A1 – Coreutils m=16

Fuente: Elaboración propia.

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	1.39846856	3.39E-05
16	ASAP AP(RLMN) S18	1.84133608	3.37E-05
16	ASAP FKKP(AP) RLE	0.97905495	0.00027367
16	ASAP FKKP(AP) RLE	0.99987157	0.00017469
16	ASAP FKKP(AP) RLE	1.13488429	0.00010557
16	ASAP FKKP(AP) S18	1.42192248	0.00027233
16	ASAP FKKP(AP) S18	1.44273909	0.00016386
16	ASAP FKKP(AP) S18	1.57775181	7.71E-05
16	ASAP FKKP(GMR) RLE	1.1341749	0.00016903
16	ASAP FKKP(GMR) RLE	1.15499152	0.00011324
16	ASAP FKKP(GMR) RLE	1.29000424	7.24E-05
16	ASAP FKKP(GMR) S18	1.57704242	0.00020705
16	ASAP FKKP(GMR) S18	1.59785904	0.00012797
16	ASAP FKKP(GMR) S18	1.73287176	7.04E-05
16	ASAP RLMN(AP) RLE	1.17481919	5.20E-05
16	ASAP RLMN(AP) S18	1.61768671	4.85E-05
16	ASAP RLMN(INT) RLE	1.35891926	4.28E-05
16	ASAP RLMN(INT) S18	1.80178678	4.79E-05

Tabla 20: Resultados A1 – Einstein.de m=4,m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	0.47614753	7.09E-06
4	ASAP AP(RLMN) S18	0.50323083	6.93E-06
4	ASAP FKPP(AP) RLE	0.12701795	3.46E-05
4	ASAP FKPP(AP) RLE	0.12779523	2.07E-05
4	ASAP FKPP(AP) RLE	0.13314228	1.28E-05
4	ASAP FKPP(AP) S18	0.15410124	3.43E-05
4	ASAP FKPP(AP) S18	0.15487853	2.07E-05
4	ASAP FKPP(AP) S18	0.16022557	1.24E-05
4	ASAP FKPP(GMR) RLE	0.09391656	2.56E-05
4	ASAP FKPP(GMR) RLE	0.09469384	1.59E-05
4	ASAP FKPP(GMR) RLE	0.10004089	1.06E-05
4	ASAP FKPP(GMR) S18	0.12099985	2.47E-05
4	ASAP FKPP(GMR) S18	0.12177714	1.54E-05
4	ASAP FKPP(GMR) S18	0.12712418	1.03E-05
4	ASAP RLMN(AP) RLE	0.14226872	8.06E-06
4	ASAP RLMN(AP) S18	0.16935201	7.52E-06
4	ASAP RLMN(INT) RLE	0.11183311	6.49E-06
4	ASAP RLMN(INT) S18	0.1389164	6.34E-06
8	ASAP AP(RLMN) RLE	0.47614753	1.38E-05
8	ASAP AP(RLMN) S18	0.50323083	1.37E-05
8	ASAP FKPP(AP) RLE	0.12701795	7.67E-05
8	ASAP FKPP(AP) RLE	0.12779523	5.13E-05
8	ASAP FKPP(AP) RLE	0.13314228	2.48E-05
8	ASAP FKPP(AP) S18	0.15410124	7.64E-05
8	ASAP FKPP(AP) S18	0.15487853	5.09E-05
8	ASAP FKPP(AP) S18	0.16022557	2.41E-05
8	ASAP FKPP(GMR) RLE	0.09391656	5.21E-05
8	ASAP FKPP(GMR) RLE	0.09469384	3.78E-05
8	ASAP FKPP(GMR) RLE	0.10004089	2.13E-05
8	ASAP FKPP(GMR) S18	0.12099985	5.52E-05
8	ASAP FKPP(GMR) S18	0.12177714	3.62E-05
8	ASAP FKPP(GMR) S18	0.12712418	2.05E-05
8	ASAP RLMN(AP) RLE	0.14226872	1.62E-05
8	ASAP RLMN(AP) S18	0.16935201	1.49E-05
8	ASAP RLMN(INT) RLE	0.11183311	1.28E-05
8	ASAP RLMN(INT) S18	0.1389164	1.21E-05

Tabla 21: Resultados A1 – Einstein.de m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	0.47614753	2.90E-05
16	ASAP AP(RLMN) S18	0.50323083	2.77E-05
16	ASAP FKKP(AP) RLE	0.12701795	0.00017302
16	ASAP FKKP(AP) RLE	0.12779523	0.00012905
16	ASAP FKKP(AP) RLE	0.13314228	5.42E-05
16	ASAP FKKP(AP) S18	0.15410124	0.00017475
16	ASAP FKKP(AP) S18	0.15487853	0.00012069
16	ASAP FKKP(AP) S18	0.16022557	5.22E-05
16	ASAP FKKP(GMR) RLE	0.09391656	0.00011666
16	ASAP FKKP(GMR) RLE	0.09469384	8.37E-05
16	ASAP FKKP(GMR) RLE	0.10004089	4.52E-05
16	ASAP FKKP(GMR) S18	0.12099985	0.00012008
16	ASAP FKKP(GMR) S18	0.12177714	8.19E-05
16	ASAP FKKP(GMR) S18	0.12712418	4.41E-05
16	ASAP RLMN(AP) RLE	0.14226872	3.38E-05
16	ASAP RLMN(AP) S18	0.16935201	3.18E-05
16	ASAP RLMN(INT) RLE	0.11183311	2.67E-05
16	ASAP RLMN(INT) S18	0.1389164	2.47E-05

Tabla 22: Resultados A1 – Einstein.en m=4,m=8

Fuente: Elaboración Propia

m	Estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	0.1458612	8.86E-06
4	ASAP AP(RLMN) S18	0.15655351	8.69E-06
4	ASAP FKKP(AP) RLE	0.05139225	4.25E-05
4	ASAP FKKP(AP) RLE	0.05189501	2.68E-05
4	ASAP FKKP(AP) RLE	0.05544856	1.43E-05
4	ASAP FKKP(AP) S18	0.06208456	4.25E-05
4	ASAP FKKP(AP) S18	0.06258733	2.72E-05
4	ASAP FKKP(AP) S18	0.06614087	1.44E-05
4	ASAP FKKP(GMR) RLE	0.04921115	3.02E-05
4	ASAP FKKP(GMR) RLE	0.04971391	2.05E-05
4	ASAP FKKP(GMR) RLE	0.05326746	1.27E-05
4	ASAP FKKP(GMR) S18	0.05990346	3.01E-05
4	ASAP FKKP(GMR) S18	0.06040623	2.10E-05
4	ASAP FKKP(GMR) S18	0.06395978	1.30E-05
4	ASAP RLMN(AP) RLE	0.06197607	9.31E-06
4	ASAP RLMN(AP) S18	0.07266838	9.64E-06
4	ASAP RLMN(INT) RLE	0.05857621	7.79E-06
4	ASAP RLMN(INT) S18	0.06926853	7.43E-06
8	ASAP AP(RLMN) RLE	0.1458612	1.78E-05
8	ASAP AP(RLMN) S18	0.15655351	1.71E-05
8	ASAP FKKP(AP) RLE	0.05139225	0.00010049
8	ASAP FKKP(AP) RLE	0.05189501	6.25E-05
8	ASAP FKKP(AP) RLE	0.05544856	3.12E-05
8	ASAP FKKP(AP) S18	0.06208456	0.00011459
8	ASAP FKKP(AP) S18	0.06258733	6.21E-05
8	ASAP FKKP(AP) S18	0.06614087	3.06E-05
8	ASAP FKKP(GMR) RLE	0.04921115	7.01E-05
8	ASAP FKKP(GMR) RLE	0.04971391	4.66E-05
8	ASAP FKKP(GMR) RLE	0.05326746	2.76E-05
8	ASAP FKKP(GMR) S18	0.05990346	6.94E-05
8	ASAP FKKP(GMR) S18	0.06040623	4.62E-05
8	ASAP FKKP(GMR) S18	0.06395978	2.76E-05
8	ASAP RLMN(AP) RLE	0.06197607	1.97E-05
8	ASAP RLMN(AP) S18	0.07266838	1.92E-05
8	ASAP RLMN(INT) RLE	0.05857621	1.61E-05
8	ASAP RLMN(INT) S18	0.06926853	1.52E-05

Tabla 23: Resultados A1 – Einstein.en m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	0.1458612	3.54E-05
16	ASAP AP(RLMN) S18	0.15655351	3.33E-05
16	ASAP FKKP(AP) RLE	0.05139225	0.00021922
16	ASAP FKKP(AP) RLE	0.05189501	0.00013044
16	ASAP FKKP(AP) RLE	0.05544856	6.33E-05
16	ASAP FKKP(AP) S18	0.06208456	0.0002338
16	ASAP FKKP(AP) S18	0.06258733	0.00012923
16	ASAP FKKP(AP) S18	0.06614087	6.18E-05
16	ASAP FKKP(GMR) RLE	0.04921115	0.00014677
16	ASAP FKKP(GMR) RLE	0.04971391	0.00011826
16	ASAP FKKP(GMR) RLE	0.05326746	5.60E-05
16	ASAP FKKP(GMR) S18	0.05990346	0.00014322
16	ASAP FKKP(GMR) S18	0.06040623	9.51E-05
16	ASAP FKKP(GMR) S18	0.06395978	5.48E-05
16	ASAP RLMN(AP) RLE	0.06197607	3.98E-05
16	ASAP RLMN(AP) S18	0.07266838	3.84E-05
16	ASAP RLMN(INT) RLE	0.05857621	3.22E-05
16	ASAP RLMN(INT) S18	0.06926853	3.03E-05

Tabla 24: Resultados A1 – English m=4,m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	1.3666265	9.80E-06
4	ASAP AP(RLMN) S18	1.74615709	9.08E-06
4	ASAP FKKP(AP) RLE	0.823656	4.94E-05
4	ASAP FKKP(AP) RLE	0.8336744	3.24E-05
4	ASAP FKKP(AP) RLE	0.90009785	1.73E-05
4	ASAP FKKP(AP) S18	1.20318659	4.88E-05
4	ASAP FKKP(AP) S18	1.21320499	3.16E-05
4	ASAP FKKP(AP) S18	1.27962844	1.68E-05
4	ASAP FKKP(GMR) RLE	0.85884313	2.93E-05
4	ASAP FKKP(GMR) RLE	0.86886153	2.11E-05
4	ASAP FKKP(GMR) RLE	0.93528498	1.47E-05
4	ASAP FKKP(GMR) S18	1.23837372	2.84E-05
4	ASAP FKKP(GMR) S18	1.24839212	2.10E-05
4	ASAP FKKP(GMR) S18	1.31481557	1.38E-05
4	ASAP RLMN(AP) RLE	0.94604654	1.09E-05
4	ASAP RLMN(AP) S18	1.32557713	1.06E-05
4	ASAP RLMN(INT) RLE	1.01292579	9.12E-06
4	ASAP RLMN(INT) S18	1.39245638	8.56E-06
8	ASAP AP(RLMN) RLE	1.3666265	1.78E-05
8	ASAP AP(RLMN) S18	1.74615709	1.64E-05
8	ASAP FKKP(AP) RLE	0.823656	0.00011687
8	ASAP FKKP(AP) RLE	0.8336744	7.33E-05
8	ASAP FKKP(AP) RLE	0.90009785	3.66E-05
8	ASAP FKKP(AP) S18	1.20318659	0.00011579
8	ASAP FKKP(AP) S18	1.21320499	7.18E-05
8	ASAP FKKP(AP) S18	1.27962844	3.56E-05
8	ASAP FKKP(GMR) RLE	0.85884313	6.82E-05
8	ASAP FKKP(GMR) RLE	0.86886153	4.76E-05
8	ASAP FKKP(GMR) RLE	0.93528498	3.17E-05
8	ASAP FKKP(GMR) S18	1.23837372	6.58E-05
8	ASAP FKKP(GMR) S18	1.24839212	4.66E-05
8	ASAP FKKP(GMR) S18	1.31481557	2.98E-05
8	ASAP RLMN(AP) RLE	0.94604654	2.27E-05
8	ASAP RLMN(AP) S18	1.32557713	2.17E-05
8	ASAP RLMN(INT) RLE	1.01292579	1.87E-05
8	ASAP RLMN(INT) S18	1.39245638	1.72E-05

Tabla 25: Resultados A1 – English m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	1.3666265	2.40E-05
16	ASAP AP(RLMN) S18	1.74615709	2.19E-05
16	ASAP FKKP(AP) RLE	0.823656	0.00024687
16	ASAP FKKP(AP) RLE	0.8336744	0.00015292
16	ASAP FKKP(AP) RLE	0.90009785	7.50E-05
16	ASAP FKKP(AP) S18	1.20318659	0.00024418
16	ASAP FKKP(AP) S18	1.21320499	0.00014973
16	ASAP FKKP(AP) S18	1.27962844	9.22E-05
16	ASAP FKKP(GMR) RLE	0.85884313	0.0001426
16	ASAP FKKP(GMR) RLE	0.86886153	9.84E-05
16	ASAP FKKP(GMR) RLE	0.93528498	6.43E-05
16	ASAP FKKP(GMR) S18	1.23837372	0.00013627
16	ASAP FKKP(GMR) S18	1.24839212	9.56E-05
16	ASAP FKKP(GMR) S18	1.31481557	6.14E-05
16	ASAP RLMN(AP) RLE	0.94604654	4.84E-05
16	ASAP RLMN(AP) S18	1.32557713	4.33E-05
16	ASAP RLMN(INT) RLE	1.01292579	3.83E-05
16	ASAP RLMN(INT) S18	1.39245638	3.47E-05

Tabla 26: Resultados A1 – World Leaders m=4,m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	0.90443885	9.12E-06
4	ASAP AP(RLMN) S18	1.00451626	1.07E-05
4	ASAP FKKP(AP) RLE	0.37417136	4.00E-05
4	ASAP FKKP(AP) RLE	0.3796454	2.69E-05
4	ASAP FKKP(AP) RLE	0.4168743	1.49E-05
4	ASAP FKKP(AP) S18	0.47424877	4.19E-05
4	ASAP FKKP(AP) S18	0.47972281	2.83E-05
4	ASAP FKKP(AP) S18	0.51695171	1.72E-05
4	ASAP FKKP(GMR) RLE	0.39721893	4.89E-05
4	ASAP FKKP(GMR) RLE	0.40269297	3.10E-05
4	ASAP FKKP(GMR) RLE	0.43992187	1.65E-05
4	ASAP FKKP(GMR) S18	0.49729634	5.11E-05
4	ASAP FKKP(GMR) S18	0.50277038	3.26E-05
4	ASAP FKKP(GMR) S18	0.53999928	1.80E-05
4	ASAP RLMN(AP) RLE	0.4426603	9.99E-06
4	ASAP RLMN(AP) S18	0.54273771	1.14E-05
4	ASAP RLMN(INT) RLE	0.47604191	8.23E-06
4	ASAP RLMN(INT) S18	0.57611932	9.68E-06
8	ASAP AP(RLMN) RLE	0.90443885	1.74E-05
8	ASAP AP(RLMN) S18	1.00451626	1.98E-05
8	ASAP FKKP(AP) RLE	0.37417136	0.00010812
8	ASAP FKKP(AP) RLE	0.3796454	6.16E-05
8	ASAP FKKP(AP) RLE	0.4168743	3.21E-05
8	ASAP FKKP(AP) S18	0.47424877	0.00011404
8	ASAP FKKP(AP) S18	0.47972281	6.37E-05
8	ASAP FKKP(AP) S18	0.51695171	3.47E-05
8	ASAP FKKP(GMR) RLE	0.39721893	0.00011353
8	ASAP FKKP(GMR) RLE	0.40269297	7.01E-05
8	ASAP FKKP(GMR) RLE	0.43992187	3.49E-05
8	ASAP FKKP(GMR) S18	0.49729634	0.00011711
8	ASAP FKKP(GMR) S18	0.50277038	7.33E-05
8	ASAP FKKP(GMR) S18	0.53999928	3.82E-05
8	ASAP RLMN(AP) RLE	0.4426603	2.06E-05
8	ASAP RLMN(AP) S18	0.54273771	2.32E-05
8	ASAP RLMN(INT) RLE	0.47604191	1.74E-05
8	ASAP RLMN(INT) S18	0.57611932	1.92E-05

Tabla 27: Resultados A1 – World Leaders m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	0.90443885	3.12E-05
16	ASAP AP(RLMN) S18	1.00451626	3.21E-05
16	ASAP FKKP(AP) RLE	0.37417136	0.00021062
16	ASAP FKKP(AP) RLE	0.3796454	0.00012761
16	ASAP FKKP(AP) RLE	0.4168743	6.29E-05
16	ASAP FKKP(AP) S18	0.47424877	0.00023799
16	ASAP FKKP(AP) S18	0.47972281	0.00013017
16	ASAP FKKP(AP) S18	0.51695171	6.57E-05
16	ASAP FKKP(GMR) RLE	0.39721893	0.00025646
16	ASAP FKKP(GMR) RLE	0.40269297	0.00015096
16	ASAP FKKP(GMR) RLE	0.43992187	7.02E-05
16	ASAP FKKP(GMR) S18	0.49729634	0.00025871
16	ASAP FKKP(GMR) S18	0.50277038	0.00015344
16	ASAP FKKP(GMR) S18	0.53999928	7.36E-05
16	ASAP RLMN(AP) RLE	0.4426603	4.17E-05
16	ASAP RLMN(AP) S18	0.54273771	4.43E-05
16	ASAP RLMN(INT) RLE	0.47604191	3.54E-05
16	ASAP RLMN(INT) S18	0.57611932	3.81E-05

ANEXO B: RESULTADOS COMPLETOS A2**Tabla 28: Resultados A2 – Coreutils m=4,m=8**

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	1.42116114	9.18E-06
4	ASAP AP(RLMN) S18	2.58460772	8.77E-06
4	ASAP FKKP(AP) RLE	1.08919832	4.25E-05
4	ASAP FKKP(AP) RLE	1.09911786	2.77E-05
4	ASAP FKKP(AP) RLE	1.16846915	1.59E-05
4	ASAP FKKP(AP) S18	2.25264491	4.20E-05
4	ASAP FKKP(AP) S18	2.26256444	2.73E-05
4	ASAP FKKP(AP) S18	2.33191573	1.56E-05
4	ASAP FKKP(GMR) RLE	1.18977338	4.31E-05
4	ASAP FKKP(GMR) RLE	1.19969291	2.93E-05
4	ASAP FKKP(GMR) RLE	1.2690442	1.62E-05
4	ASAP FKKP(GMR) S18	2.35321996	4.26E-05
4	ASAP FKKP(GMR) S18	2.3631395	2.85E-05
4	ASAP FKKP(GMR) S18	2.43249078	1.57E-05
4	ASAP RLMN(AP) RLE	1.23255775	1.09E-05
4	ASAP RLMN(AP) S18	2.39600433	1.04E-05
4	ASAP RLMN(INT) RLE	1.32782846	9.36E-06
4	ASAP RLMN(INT) S18	2.49127504	8.62E-06
8	ASAP AP(RLMN) RLE	1.42116114	1.91E-05
8	ASAP AP(RLMN) S18	2.58460772	1.91E-05
8	ASAP FKKP(AP) RLE	1.08919832	0.00011954
8	ASAP FKKP(AP) RLE	1.09911786	7.19E-05
8	ASAP FKKP(AP) RLE	1.16846915	3.62E-05
8	ASAP FKKP(AP) S18	2.25264491	0.00011166
8	ASAP FKKP(AP) S18	2.26256444	7.00E-05
8	ASAP FKKP(AP) S18	2.33191573	3.57E-05
8	ASAP FKKP(GMR) RLE	1.18977338	0.00011502
8	ASAP FKKP(GMR) RLE	1.19969291	7.21E-05
8	ASAP FKKP(GMR) RLE	1.2690442	3.83E-05
8	ASAP FKKP(GMR) S18	2.35321996	0.00011918
8	ASAP FKKP(GMR) S18	2.3631395	7.15E-05
8	ASAP FKKP(GMR) S18	2.43249078	3.86E-05
8	ASAP RLMN(AP) RLE	1.23255775	2.44E-05
8	ASAP RLMN(AP) S18	2.39600433	2.45E-05
8	ASAP RLMN(INT) RLE	1.32782846	2.08E-05
8	ASAP RLMN(INT) S18	2.49127504	1.98E-05

Tabla 29: Resultados A2 – Coreutils m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	1.42116114	3.44E-05
16	ASAP AP(RLMN) S18	2.58460772	3.36E-05
16	ASAP FKKP(AP) RLE	1.08919832	0.0002333
16	ASAP FKKP(AP) RLE	1.09911786	0.00020004
16	ASAP FKKP(AP) RLE	1.16846915	7.26E-05
16	ASAP FKKP(AP) S18	2.25264491	0.00023449
16	ASAP FKKP(AP) S18	2.26256444	0.00014451
16	ASAP FKKP(AP) S18	2.33191573	7.16E-05
16	ASAP FKKP(GMR) RLE	1.18977338	0.00023812
16	ASAP FKKP(GMR) RLE	1.19969291	0.00015948
16	ASAP FKKP(GMR) RLE	1.2690442	8.01E-05
16	ASAP FKKP(GMR) S18	2.35321996	0.00024044
16	ASAP FKKP(GMR) S18	2.3631395	0.00014812
16	ASAP FKKP(GMR) S18	2.43249078	7.97E-05
16	ASAP RLMN(AP) RLE	1.23255775	4.89E-05
16	ASAP RLMN(AP) S18	2.39600433	4.86E-05
16	ASAP RLMN(INT) RLE	1.32782846	4.10E-05
16	ASAP RLMN(INT) S18	2.49127504	3.94E-05

Tabla 30: Resultados A2 – Einstein.de m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	0.47205317	6.50E-06
4	ASAP AP(RLMN) S18	0.51896687	6.29E-06
4	ASAP FKKP(AP) RLE	0.13069324	2.23E-05
4	ASAP FKKP(AP) RLE	0.13102209	1.74E-05
4	ASAP FKKP(AP) RLE	0.13371697	1.19E-05
4	ASAP FKKP(AP) S18	0.17760694	2.24E-05
4	ASAP FKKP(AP) S18	0.17793579	1.71E-05
4	ASAP FKKP(AP) S18	0.18063067	1.16E-05
4	ASAP FKKP(GMR) RLE	0.10105814	1.62E-05
4	ASAP FKKP(GMR) RLE	0.10138699	1.27E-05
4	ASAP FKKP(GMR) RLE	0.10408187	9.90E-06
4	ASAP FKKP(GMR) S18	0.14797184	1.56E-05
4	ASAP FKKP(GMR) S18	0.14830069	1.27E-05
4	ASAP FKKP(GMR) S18	0.15099557	9.69E-06
4	ASAP RLMN(AP) RLE	0.14175195	7.57E-06
4	ASAP RLMN(AP) S18	0.18866565	7.33E-06
4	ASAP RLMN(INT) RLE	0.11033993	6.41E-06
4	ASAP RLMN(INT) S18	0.15725363	6.16E-06
8	ASAP AP(RLMN) RLE	0.47205317	1.28E-05
8	ASAP AP(RLMN) S18	0.51896687	1.21E-05
8	ASAP FKKP(AP) RLE	0.13069324	6.24E-05
8	ASAP FKKP(AP) RLE	0.13102209	4.49E-05
8	ASAP FKKP(AP) RLE	0.13371697	2.35E-05
8	ASAP FKKP(AP) S18	0.17760694	6.30E-05
8	ASAP FKKP(AP) S18	0.17793579	4.43E-05
8	ASAP FKKP(AP) S18	0.18063067	2.27E-05
8	ASAP FKKP(GMR) RLE	0.10105814	4.35E-05
8	ASAP FKKP(GMR) RLE	0.10138699	3.19E-05
8	ASAP FKKP(GMR) RLE	0.10408187	1.97E-05
8	ASAP FKKP(GMR) S18	0.14797184	4.27E-05
8	ASAP FKKP(GMR) S18	0.14830069	3.15E-05
8	ASAP FKKP(GMR) S18	0.15099557	1.92E-05
8	ASAP RLMN(AP) RLE	0.14175195	1.54E-05
8	ASAP RLMN(AP) S18	0.18866565	1.50E-05
8	ASAP RLMN(INT) RLE	0.11033993	1.25E-05
8	ASAP RLMN(INT) S18	0.15725363	1.20E-05

Tabla 31: Resultados A2 – Einstein.de m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	0.47205317	2.70E-05
16	ASAP AP(RLMN) S18	0.51896687	2.49E-05
16	ASAP FKKP(AP) RLE	0.13069324	0.00014207
16	ASAP FKKP(AP) RLE	0.13102209	9.63E-05
16	ASAP FKKP(AP) RLE	0.13371697	5.26E-05
16	ASAP FKKP(AP) S18	0.17760694	0.00014595
16	ASAP FKKP(AP) S18	0.17793579	0.00011176
16	ASAP FKKP(AP) S18	0.18063067	5.08E-05
16	ASAP FKKP(GMR) RLE	0.10105814	9.11E-05
16	ASAP FKKP(GMR) RLE	0.10138699	7.59E-05
16	ASAP FKKP(GMR) RLE	0.10408187	5.16E-05
16	ASAP FKKP(GMR) S18	0.14797184	0.00010695
16	ASAP FKKP(GMR) S18	0.14830069	6.52E-05
16	ASAP FKKP(GMR) S18	0.15099557	4.19E-05
16	ASAP RLMN(AP) RLE	0.14175195	3.27E-05
16	ASAP RLMN(AP) S18	0.18866565	3.16E-05
16	ASAP RLMN(INT) RLE	0.11033993	2.61E-05
16	ASAP RLMN(INT) S18	0.15725363	2.45E-05

Tabla 32: Resultados A2 – Einstein.en m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	0.14401228	7.78E-06
4	ASAP AP(RLMN) S18	0.1634509	7.52E-06
4	ASAP FKKP(AP) RLE	0.05473312	3.39E-05
4	ASAP FKKP(AP) RLE	0.05498853	2.34E-05
4	ASAP FKKP(AP) RLE	0.05702614	1.32E-05
4	ASAP FKKP(AP) S18	0.07417175	3.44E-05
4	ASAP FKKP(AP) S18	0.07442716	2.33E-05
4	ASAP FKKP(AP) S18	0.07646476	1.33E-05
4	ASAP FKKP(GMR) RLE	0.05052337	2.54E-05
4	ASAP FKKP(GMR) RLE	0.05077877	1.83E-05
4	ASAP FKKP(GMR) RLE	0.05281638	1.20E-05
4	ASAP FKKP(GMR) S18	0.06996199	2.51E-05
4	ASAP FKKP(GMR) S18	0.0702174	1.82E-05
4	ASAP FKKP(GMR) S18	0.07225501	1.16E-05
4	ASAP RLMN(AP) RLE	0.06208205	9.16E-06
4	ASAP RLMN(AP) S18	0.08152068	8.88E-06
4	ASAP RLMN(INT) RLE	0.05837928	7.59E-06
4	ASAP RLMN(INT) S18	0.07781791	7.32E-06
8	ASAP AP(RLMN) RLE	0.14401228	1.57E-05
8	ASAP AP(RLMN) S18	0.1634509	1.53E-05
8	ASAP FKKP(AP) RLE	0.05473312	7.84E-05
8	ASAP FKKP(AP) RLE	0.05498853	5.27E-05
8	ASAP FKKP(AP) RLE	0.05702614	2.77E-05
8	ASAP FKKP(AP) S18	0.07417175	7.88E-05
8	ASAP FKKP(AP) S18	0.07442716	5.14E-05
8	ASAP FKKP(AP) S18	0.07646476	2.71E-05
8	ASAP FKKP(GMR) RLE	0.05052337	5.82E-05
8	ASAP FKKP(GMR) RLE	0.05077877	4.09E-05
8	ASAP FKKP(GMR) RLE	0.05281638	2.52E-05
8	ASAP FKKP(GMR) S18	0.06996199	6.10E-05
8	ASAP FKKP(GMR) S18	0.0702174	4.07E-05
8	ASAP FKKP(GMR) S18	0.07225501	2.44E-05
8	ASAP RLMN(AP) RLE	0.06208205	1.87E-05
8	ASAP RLMN(AP) S18	0.08152068	1.86E-05
8	ASAP RLMN(INT) RLE	0.05837928	1.54E-05
8	ASAP RLMN(INT) S18	0.07781791	1.50E-05

Tabla 33: Resultados A2 – Einstein.en m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	0.14401228	3.07E-05
16	ASAP AP(RLMN) S18	0.1634509	2.98E-05
16	ASAP FKKP(AP) RLE	0.05473312	0.00019003
16	ASAP FKKP(AP) RLE	0.05498853	0.00010901
16	ASAP FKKP(AP) RLE	0.05702614	5.58E-05
16	ASAP FKKP(AP) S18	0.07417175	0.00016464
16	ASAP FKKP(AP) S18	0.07442716	0.00010725
16	ASAP FKKP(AP) S18	0.07646476	5.55E-05
16	ASAP FKKP(GMR) RLE	0.05052337	0.00012225
16	ASAP FKKP(GMR) RLE	0.05077877	0.00010207
16	ASAP FKKP(GMR) RLE	0.05281638	5.72E-05
16	ASAP FKKP(GMR) S18	0.06996199	0.00015068
16	ASAP FKKP(GMR) S18	0.0702174	8.42E-05
16	ASAP FKKP(GMR) S18	0.07225501	4.91E-05
16	ASAP RLMN(AP) RLE	0.06208205	3.78E-05
16	ASAP RLMN(AP) S18	0.08152068	3.74E-05
16	ASAP RLMN(INT) RLE	0.05837928	2.98E-05
16	ASAP RLMN(INT) S18	0.07781791	2.97E-05

Tabla 34: Resultados A2 – English m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	1.44005782	8.64E-06
4	ASAP AP(RLMN) S18	2.18165276	8.28E-06
4	ASAP FKKP(AP) RLE	0.93960938	4.07E-05
4	ASAP FKKP(AP) RLE	0.94543914	2.70E-05
4	ASAP FKKP(AP) RLE	0.98715808	1.50E-05
4	ASAP FKKP(AP) S18	1.68120432	4.03E-05
4	ASAP FKKP(AP) S18	1.68703408	2.68E-05
4	ASAP FKKP(AP) S18	1.72875303	1.53E-05
4	ASAP FKKP(GMR) RLE	0.92707534	3.18E-05
4	ASAP FKKP(GMR) RLE	0.93290511	2.18E-05
4	ASAP FKKP(GMR) RLE	0.97462405	1.41E-05
4	ASAP FKKP(GMR) S18	1.66867029	3.09E-05
4	ASAP FKKP(GMR) S18	1.67450005	2.22E-05
4	ASAP FKKP(GMR) S18	1.71621899	1.36E-05
4	ASAP RLMN(AP) RLE	1.05001533	1.05E-05
4	ASAP RLMN(AP) S18	1.79161028	1.06E-05
4	ASAP RLMN(INT) RLE	1.08138039	8.63E-06
4	ASAP RLMN(INT) S18	1.82297533	8.43E-06
8	ASAP AP(RLMN) RLE	1.44005782	1.65E-05
8	ASAP AP(RLMN) S18	2.18165276	1.55E-05
8	ASAP FKKP(AP) RLE	0.93960938	9.21E-05
8	ASAP FKKP(AP) RLE	0.94543914	6.10E-05
8	ASAP FKKP(AP) RLE	0.98715808	3.26E-05
8	ASAP FKKP(AP) S18	1.68120432	9.23E-05
8	ASAP FKKP(AP) S18	1.68703408	5.97E-05
8	ASAP FKKP(AP) S18	1.72875303	3.22E-05
8	ASAP FKKP(GMR) RLE	0.92707534	7.18E-05
8	ASAP FKKP(GMR) RLE	0.93290511	4.90E-05
8	ASAP FKKP(GMR) RLE	0.97462405	3.01E-05
8	ASAP FKKP(GMR) S18	1.66867029	7.20E-05
8	ASAP FKKP(GMR) S18	1.67450005	4.95E-05
8	ASAP FKKP(GMR) S18	1.71621899	2.90E-05
8	ASAP RLMN(AP) RLE	1.05001533	2.23E-05
8	ASAP RLMN(AP) S18	1.79161028	2.18E-05
8	ASAP RLMN(INT) RLE	1.08138039	1.78E-05
8	ASAP RLMN(INT) S18	1.82297533	1.68E-05

Tabla 35: Resultados A2 – English m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	1.44005782	2.64E-05
16	ASAP AP(RLMN) S18	2.18165276	2.53E-05
16	ASAP FKKP(AP) RLE	0.93960938	0.00020629
16	ASAP FKKP(AP) RLE	0.94543914	0.00012632
16	ASAP FKKP(AP) RLE	0.98715808	6.62E-05
16	ASAP FKKP(AP) S18	1.68120432	0.00024555
16	ASAP FKKP(AP) S18	1.68703408	0.0001243
16	ASAP FKKP(AP) S18	1.72875303	7.96E-05
16	ASAP FKKP(GMR) RLE	0.92707534	0.00015143
16	ASAP FKKP(GMR) RLE	0.93290511	0.00011114
16	ASAP FKKP(GMR) RLE	0.97462405	6.11E-05
16	ASAP FKKP(GMR) S18	1.66867029	0.00014985
16	ASAP FKKP(GMR) S18	1.67450005	0.00010211
16	ASAP FKKP(GMR) S18	1.71621899	5.99E-05
16	ASAP RLMN(AP) RLE	1.05001533	4.54E-05
16	ASAP RLMN(AP) S18	1.79161028	4.34E-05
16	ASAP RLMN(INT) RLE	1.08138039	3.54E-05
16	ASAP RLMN(INT) S18	1.82297533	3.39E-05

Tabla 36: Resultados A2 – World Leaders m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP AP(RLMN) RLE	0.87794574	9.38E-06
4	ASAP AP(RLMN) S18	1.12772866	1.03E-05
4	ASAP FKKP(AP) RLE	0.39080499	6.81E-05
4	ASAP FKKP(AP) RLE	0.3925402	3.62E-05
4	ASAP FKKP(AP) RLE	0.40673562	1.61E-05
4	ASAP FKKP(AP) S18	0.64058792	6.88E-05
4	ASAP FKKP(AP) S18	0.64232312	3.72E-05
4	ASAP FKKP(AP) S18	0.65651855	1.80E-05
4	ASAP FKKP(GMR) RLE	0.37461337	2.76E-05
4	ASAP FKKP(GMR) RLE	0.37634857	1.80E-05
4	ASAP FKKP(GMR) RLE	0.390544	1.17E-05
4	ASAP FKKP(GMR) S18	0.62439629	2.86E-05
4	ASAP FKKP(GMR) S18	0.62613149	2.53E-05
4	ASAP FKKP(GMR) S18	0.64032692	1.27E-05
4	ASAP RLMN(AP) RLE	0.4266061	1.02E-05
4	ASAP RLMN(AP) S18	0.67638902	1.14E-05
4	ASAP RLMN(INT) RLE	0.45899345	8.25E-06
4	ASAP RLMN(INT) S18	0.70877638	9.51E-06
8	ASAP AP(RLMN) RLE	0.87794574	1.74E-05
8	ASAP AP(RLMN) S18	1.12772866	1.93E-05
8	ASAP FKKP(AP) RLE	0.39080499	0.00017021
8	ASAP FKKP(AP) RLE	0.3925402	8.09E-05
8	ASAP FKKP(AP) RLE	0.40673562	3.52E-05
8	ASAP FKKP(AP) S18	0.64058792	0.00015411
8	ASAP FKKP(AP) S18	0.64232312	8.43E-05
8	ASAP FKKP(AP) S18	0.65651855	3.78E-05
8	ASAP FKKP(GMR) RLE	0.37461337	6.46E-05
8	ASAP FKKP(GMR) RLE	0.37634857	4.09E-05
8	ASAP FKKP(GMR) RLE	0.390544	2.54E-05
8	ASAP FKKP(GMR) S18	0.62439629	6.56E-05
8	ASAP FKKP(GMR) S18	0.62613149	5.33E-05
8	ASAP FKKP(GMR) S18	0.64032692	2.68E-05
8	ASAP RLMN(AP) RLE	0.4266061	2.11E-05
8	ASAP RLMN(AP) S18	0.67638902	2.32E-05
8	ASAP RLMN(INT) RLE	0.45899345	1.68E-05
8	ASAP RLMN(INT) S18	0.70877638	1.93E-05

Tabla 37: Resultados A2 – World Leaders m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP AP(RLMN) RLE	0.87794574	3.20E-05
16	ASAP AP(RLMN) S18	1.12772866	3.36E-05
16	ASAP FKKP(AP) RLE	0.39080499	0.00030407
16	ASAP FKKP(AP) RLE	0.3925402	0.00015612
16	ASAP FKKP(AP) RLE	0.40673562	8.23E-05
16	ASAP FKKP(AP) S18	0.64058792	0.00030595
16	ASAP FKKP(AP) S18	0.64232312	0.0001667
16	ASAP FKKP(AP) S18	0.65651855	7.33E-05
16	ASAP FKKP(GMR) RLE	0.37461337	0.0001592
16	ASAP FKKP(GMR) RLE	0.37634857	9.61E-05
16	ASAP FKKP(GMR) RLE	0.390544	5.19E-05
16	ASAP FKKP(GMR) S18	0.62439629	0.00014669
16	ASAP FKKP(GMR) S18	0.62613149	0.00010588
16	ASAP FKKP(GMR) S18	0.64032692	5.37E-05
16	ASAP RLMN(AP) RLE	0.4266061	4.16E-05
16	ASAP RLMN(AP) S18	0.67638902	4.44E-05
16	ASAP RLMN(INT) RLE	0.45899345	3.33E-05
16	ASAP RLMN(INT) S18	0.70877638	3.61E-05

ANEXO C: RESULTADOS COMPLETOS A3**Tabla 38: Resultados A3 – Coreutils m=4, m=8**

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	1.101832913	2.02E-05
4	ASAP FKKP(AP) RLE	1.110431828	1.45E-05
4	ASAP FKKP(AP) RLE	1.173175806	9.24E-06
4	ASAP FKKP(AP) S18	2.201308254	2.02E-05
4	ASAP FKKP(AP) S18	2.209907168	1.44E-05
4	ASAP FKKP(AP) S18	2.272651146	9.20E-06
4	ASAP FKKP(GMR) RLE	1.085143872	1.27E-05
4	ASAP FKKP(GMR) RLE	1.093742787	9.97E-06
4	ASAP FKKP(GMR) RLE	1.156486765	7.81E-06
4	ASAP FKKP(GMR) S18	2.184619212	1.26E-05
4	ASAP FKKP(GMR) S18	2.193218127	1.01E-05
4	ASAP FKKP(GMR) S18	2.255962105	7.92E-06
4	ASAP RLMN(AP) RLE	1.20826131	6.83E-06
4	ASAP RLMN(AP) S18	2.30773665	6.95E-06
4	ASAP RLMN(INT) RLE	1.155578575	6.37E-06
4	ASAP RLMN(INT) S18	2.255053915	6.53E-06
8	ASAP FKKP(AP) RLE	1.101832913	5.03E-05
8	ASAP FKKP(AP) RLE	1.110431828	3.43E-05
8	ASAP FKKP(AP) RLE	1.173175806	2.04E-05
8	ASAP FKKP(AP) S18	2.201308254	5.13E-05
8	ASAP FKKP(AP) S18	2.209907168	3.55E-05
8	ASAP FKKP(AP) S18	2.272651146	2.13E-05
8	ASAP FKKP(GMR) RLE	1.085143872	2.91E-05
8	ASAP FKKP(GMR) RLE	1.093742787	2.25E-05
8	ASAP FKKP(GMR) RLE	1.156486765	1.65E-05
8	ASAP FKKP(GMR) S18	2.184619212	3.04E-05
8	ASAP FKKP(GMR) S18	2.193218127	2.33E-05
8	ASAP FKKP(GMR) S18	2.255962105	1.73E-05
8	ASAP RLMN(AP) RLE	1.20826131	1.47E-05
8	ASAP RLMN(AP) S18	2.30773665	1.51E-05
8	ASAP RLMN(INT) RLE	1.155578575	1.35E-05
8	ASAP RLMN(INT) S18	2.255053915	1.40E-05

Tabla 39: Resultados A3 – Coreutils m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	1.101832913	9.85E-05
16	ASAP FKKP(AP) RLE	1.110431828	6.66E-05
16	ASAP FKKP(AP) RLE	1.173175806	3.81E-05
16	ASAP FKKP(AP) S18	2.201308254	0.000100661
16	ASAP FKKP(AP) S18	2.209907168	6.87E-05
16	ASAP FKKP(AP) S18	2.272651146	4.05E-05
16	ASAP FKKP(GMR) RLE	1.085143872	5.66E-05
16	ASAP FKKP(GMR) RLE	1.093742787	4.83E-05
16	ASAP FKKP(GMR) RLE	1.156486765	3.11E-05
16	ASAP FKKP(GMR) S18	2.184619212	5.93E-05
16	ASAP FKKP(GMR) S18	2.193218127	4.50E-05
16	ASAP FKKP(GMR) S18	2.255962105	3.35E-05
16	ASAP RLMN(AP) RLE	1.20826131	2.67E-05
16	ASAP RLMN(AP) S18	2.30773665	2.88E-05
16	ASAP RLMN(INT) RLE	1.155578575	2.43E-05
16	ASAP RLMN(INT) S18	2.255053915	2.70E-05

Tabla 40: Resultados A3 – Einstein.de m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	0.123571141	1.06E-05
4	ASAP FKKP(AP) RLE	0.12374838	9.59E-06
4	ASAP FKKP(AP) RLE	0.125529304	8.00E-06
4	ASAP FKKP(AP) S18	0.168111344	1.05E-05
4	ASAP FKKP(AP) S18	0.168288582	9.20E-06
4	ASAP FKKP(AP) S18	0.170069507	7.54E-06
4	ASAP FKKP(GMR) RLE	0.099637829	7.80E-06
4	ASAP FKKP(GMR) RLE	0.099815068	7.01E-06
4	ASAP FKKP(GMR) RLE	0.101595992	6.44E-06
4	ASAP FKKP(GMR) S18	0.144178032	7.38E-06
4	ASAP FKKP(GMR) S18	0.14435527	6.79E-06
4	ASAP FKKP(GMR) S18	0.146136195	5.87E-06
4	ASAP RLMN(AP) RLE	0.131941381	5.36E-06
4	ASAP RLMN(AP) S18	0.176481583	4.94E-06
4	ASAP RLMN(INT) RLE	0.105501375	4.77E-06
4	ASAP RLMN(INT) S18	0.150041577	4.39E-06
8	ASAP FKKP(AP) RLE	0.123571141	2.36E-05
8	ASAP FKKP(AP) RLE	0.12374838	1.89E-05
8	ASAP FKKP(AP) RLE	0.125529304	1.39E-05
8	ASAP FKKP(AP) S18	0.168111344	2.35E-05
8	ASAP FKKP(AP) S18	0.168288582	1.84E-05
8	ASAP FKKP(AP) S18	0.170069507	1.33E-05
8	ASAP FKKP(GMR) RLE	0.099637829	1.66E-05
8	ASAP FKKP(GMR) RLE	0.099815068	1.34E-05
8	ASAP FKKP(GMR) RLE	0.101595992	1.16E-05
8	ASAP FKKP(GMR) S18	0.144178032	1.64E-05
8	ASAP FKKP(GMR) S18	0.14435527	1.31E-05
8	ASAP FKKP(GMR) S18	0.146136195	1.11E-05
8	ASAP RLMN(AP) RLE	0.131941381	9.66E-06
8	ASAP RLMN(AP) S18	0.176481583	9.15E-06
8	ASAP RLMN(INT) RLE	0.105501375	8.77E-06
8	ASAP RLMN(INT) S18	0.150041577	8.14E-06

Tabla 41: Resultados A3 – Einstein.de m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	0.123571141	5.30E-05
16	ASAP FKKP(AP) RLE	0.12374838	5.29E-05
16	ASAP FKKP(AP) RLE	0.125529304	2.99E-05
16	ASAP FKKP(AP) S18	0.168111344	5.19E-05
16	ASAP FKKP(AP) S18	0.168288582	4.56E-05
16	ASAP FKKP(AP) S18	0.170069507	2.87E-05
16	ASAP FKKP(GMR) RLE	0.099637829	3.49E-05
16	ASAP FKKP(GMR) RLE	0.099815068	3.10E-05
16	ASAP FKKP(GMR) RLE	0.101595992	2.38E-05
16	ASAP FKKP(GMR) S18	0.144178032	3.49E-05
16	ASAP FKKP(GMR) S18	0.14435527	3.03E-05
16	ASAP FKKP(GMR) S18	0.146136195	2.30E-05
16	ASAP RLMN(AP) RLE	0.131941381	1.95E-05
16	ASAP RLMN(AP) S18	0.176481583	1.84E-05
16	ASAP RLMN(INT) RLE	0.105501375	1.71E-05
16	ASAP RLMN(INT) S18	0.150041577	1.65E-05

Tabla 42: Resultados A3 – Einstein.en m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	0.052097425	1.72E-05
4	ASAP FKKP(AP) RLE	0.052236993	1.27E-05
4	ASAP FKKP(AP) RLE	0.053621006	8.51E-06
4	ASAP FKKP(AP) S18	0.071537561	1.64E-05
4	ASAP FKKP(AP) S18	0.071677128	1.30E-05
4	ASAP FKKP(AP) S18	0.073061142	8.77E-06
4	ASAP FKKP(GMR) RLE	0.047819795	1.05E-05
4	ASAP FKKP(GMR) RLE	0.047959363	9.55E-06
4	ASAP FKKP(GMR) RLE	0.049343376	7.89E-06
4	ASAP FKKP(GMR) S18	0.06725993	1.09E-05
4	ASAP FKKP(GMR) S18	0.067399498	9.02E-06
4	ASAP FKKP(GMR) S18	0.068783511	7.29E-06
4	ASAP RLMN(AP) RLE	0.056328349	6.04E-06
4	ASAP RLMN(AP) S18	0.075768484	6.52E-06
4	ASAP RLMN(INT) RLE	0.052712458	6.11E-06
4	ASAP RLMN(INT) S18	0.072152594	5.69E-06
8	ASAP FKKP(AP) RLE	0.052097425	3.37E-05
8	ASAP FKKP(AP) RLE	0.052236993	2.48E-05
8	ASAP FKKP(AP) RLE	0.053621006	1.63E-05
8	ASAP FKKP(AP) S18	0.071537561	3.33E-05
8	ASAP FKKP(AP) S18	0.071677128	2.54E-05
8	ASAP FKKP(AP) S18	0.073061142	1.68E-05
8	ASAP FKKP(GMR) RLE	0.047819795	2.16E-05
8	ASAP FKKP(GMR) RLE	0.047959363	1.77E-05
8	ASAP FKKP(GMR) RLE	0.049343376	1.34E-05
8	ASAP FKKP(GMR) S18	0.06725993	2.19E-05
8	ASAP FKKP(GMR) S18	0.067399498	1.77E-05
8	ASAP FKKP(GMR) S18	0.068783511	1.38E-05
8	ASAP RLMN(AP) RLE	0.056328349	1.14E-05
8	ASAP RLMN(AP) S18	0.075768484	1.20E-05
8	ASAP RLMN(INT) RLE	0.052712458	1.06E-05
8	ASAP RLMN(INT) S18	0.072152594	1.06E-05

Tabla 43: Resultados A3 – Einstein.en m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	0.052097425	6.66E-05
16	ASAP FKKP(AP) RLE	0.052236993	4.95E-05
16	ASAP FKKP(AP) RLE	0.053621006	3.13E-05
16	ASAP FKKP(AP) S18	0.071537561	6.67E-05
16	ASAP FKKP(AP) S18	0.071677128	5.84E-05
16	ASAP FKKP(AP) S18	0.073061142	3.22E-05
16	ASAP FKKP(GMR) RLE	0.047819795	4.32E-05
16	ASAP FKKP(GMR) RLE	0.047959363	3.51E-05
16	ASAP FKKP(GMR) RLE	0.049343376	2.63E-05
16	ASAP FKKP(GMR) S18	0.06725993	4.48E-05
16	ASAP FKKP(GMR) S18	0.067399498	3.53E-05
16	ASAP FKKP(GMR) S18	0.068783511	2.67E-05
16	ASAP RLMN(AP) RLE	0.056328349	2.23E-05
16	ASAP RLMN(AP) S18	0.075768484	2.25E-05
16	ASAP RLMN(INT) RLE	0.052712458	2.00E-05
16	ASAP RLMN(INT) S18	0.072152594	2.05E-05

Tabla 44: Resultados A3 – English m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	0.904915547	1.92E-05
4	ASAP FKKP(AP) RLE	0.908685482	1.46E-05
4	ASAP FKKP(AP) RLE	0.937583036	9.63E-06
4	ASAP FKKP(AP) S18	1.64303072	1.93E-05
4	ASAP FKKP(AP) S18	1.646800655	1.45E-05
4	ASAP FKKP(AP) S18	1.675698209	9.75E-06
4	ASAP FKKP(GMR) RLE	0.880079295	1.16E-05
4	ASAP FKKP(GMR) RLE	0.88384923	9.84E-06
4	ASAP FKKP(GMR) RLE	0.912746784	7.97E-06
4	ASAP FKKP(GMR) S18	1.618194468	1.19E-05
4	ASAP FKKP(GMR) S18	1.621964403	9.74E-06
4	ASAP FKKP(GMR) S18	1.650861958	7.85E-06
4	ASAP RLMN(AP) RLE	0.979433325	6.92E-06
4	ASAP RLMN(AP) S18	1.717548498	7.05E-06
4	ASAP RLMN(INT) RLE	0.974550658	6.16E-06
4	ASAP RLMN(INT) S18	1.712665831	6.36E-06
8	ASAP FKKP(AP) RLE	0.904915547	4.02E-05
8	ASAP FKKP(AP) RLE	0.908685482	2.93E-05
8	ASAP FKKP(AP) RLE	0.937583036	1.85E-05
8	ASAP FKKP(AP) S18	1.64303072	4.01E-05
8	ASAP FKKP(AP) S18	1.646800655	2.93E-05
8	ASAP FKKP(AP) S18	1.675698209	1.85E-05
8	ASAP FKKP(GMR) RLE	0.880079295	2.41E-05
8	ASAP FKKP(GMR) RLE	0.88384923	1.94E-05
8	ASAP FKKP(GMR) RLE	0.912746784	1.54E-05
8	ASAP FKKP(GMR) S18	1.618194468	2.44E-05
8	ASAP FKKP(GMR) S18	1.621964403	1.95E-05
8	ASAP FKKP(GMR) S18	1.650861958	1.48E-05
8	ASAP RLMN(AP) RLE	0.979433325	1.31E-05
8	ASAP RLMN(AP) S18	1.717548498	1.31E-05
8	ASAP RLMN(INT) RLE	0.974550658	1.17E-05
8	ASAP RLMN(INT) S18	1.712665831	1.18E-05

Tabla 45: Resultados A3 – English m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	0.904915547	8.20E-05
16	ASAP FKKP(AP) RLE	0.908685482	6.71E-05
16	ASAP FKKP(AP) RLE	0.937583036	3.64E-05
16	ASAP FKKP(AP) S18	1.64303072	8.16E-05
16	ASAP FKKP(AP) S18	1.646800655	5.87E-05
16	ASAP FKKP(AP) S18	1.675698209	3.62E-05
16	ASAP FKKP(GMR) RLE	0.880079295	4.85E-05
16	ASAP FKKP(GMR) RLE	0.88384923	3.85E-05
16	ASAP FKKP(GMR) RLE	0.912746784	2.93E-05
16	ASAP FKKP(GMR) S18	1.618194468	4.89E-05
16	ASAP FKKP(GMR) S18	1.621964403	3.82E-05
16	ASAP FKKP(GMR) S18	1.650861958	2.87E-05
16	ASAP RLMN(AP) RLE	0.979433325	2.53E-05
16	ASAP RLMN(AP) S18	1.717548498	2.47E-05
16	ASAP RLMN(INT) RLE	0.974550658	2.28E-05
16	ASAP RLMN(INT) S18	1.712665831	2.28E-05

Tabla 46: Resultados A3 – World Leaders m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	0.37957951	1.57E-05
4	ASAP FKKP(AP) RLE	0.380879886	1.34E-05
4	ASAP FKKP(AP) RLE	0.392700692	9.86E-06
4	ASAP FKKP(AP) S18	0.619308107	1.76E-05
4	ASAP FKKP(AP) S18	0.620608483	1.54E-05
4	ASAP FKKP(AP) S18	0.63242929	1.14E-05
4	ASAP FKKP(GMR) RLE	0.357817186	1.01E-05
4	ASAP FKKP(GMR) RLE	0.359117562	9.15E-06
4	ASAP FKKP(GMR) RLE	0.370938368	7.98E-06
4	ASAP FKKP(GMR) S18	0.597545784	1.20E-05
4	ASAP FKKP(GMR) S18	0.59884616	1.06E-05
4	ASAP FKKP(GMR) S18	0.610666966	9.73E-06
4	ASAP RLMN(AP) RLE	0.403555702	6.73E-06
4	ASAP RLMN(AP) S18	0.6432843	8.78E-06
4	ASAP RLMN(INT) RLE	0.397418913	6.00E-06
4	ASAP RLMN(INT) S18	0.637147511	7.90E-06
8	ASAP FKKP(AP) RLE	0.37957951	3.16E-05
8	ASAP FKKP(AP) RLE	0.380879886	2.57E-05
8	ASAP FKKP(AP) RLE	0.392700692	1.76E-05
8	ASAP FKKP(AP) S18	0.619308107	3.55E-05
8	ASAP FKKP(AP) S18	0.620608483	2.96E-05
8	ASAP FKKP(AP) S18	0.63242929	2.10E-05
8	ASAP FKKP(GMR) RLE	0.357817186	1.98E-05
8	ASAP FKKP(GMR) RLE	0.359117562	1.75E-05
8	ASAP FKKP(GMR) RLE	0.370938368	1.42E-05
8	ASAP FKKP(GMR) S18	0.597545784	2.79E-05
8	ASAP FKKP(GMR) S18	0.59884616	2.12E-05
8	ASAP FKKP(GMR) S18	0.610666966	1.83E-05
8	ASAP RLMN(AP) RLE	0.403555702	1.25E-05
8	ASAP RLMN(AP) S18	0.6432843	1.62E-05
8	ASAP RLMN(INT) RLE	0.397418913	1.08E-05
8	ASAP RLMN(INT) S18	0.637147511	1.50E-05

Tabla 47: Resultados A3 – World Leaders m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	0.37957951	5.41E-05
16	ASAP FKKP(AP) RLE	0.380879886	4.32E-05
16	ASAP FKKP(AP) RLE	0.392700692	2.93E-05
16	ASAP FKKP(AP) S18	0.619308107	6.18E-05
16	ASAP FKKP(AP) S18	0.620608483	5.08E-05
16	ASAP FKKP(AP) S18	0.63242929	3.64E-05
16	ASAP FKKP(GMR) RLE	0.357817186	3.45E-05
16	ASAP FKKP(GMR) RLE	0.359117562	2.96E-05
16	ASAP FKKP(GMR) RLE	0.370938368	2.38E-05
16	ASAP FKKP(GMR) S18	0.597545784	5.45E-05
16	ASAP FKKP(GMR) S18	0.59884616	3.69E-05
16	ASAP FKKP(GMR) S18	0.610666966	3.15E-05
16	ASAP RLMN(AP) RLE	0.403555702	2.15E-05
16	ASAP RLMN(AP) S18	0.6432843	2.91E-05
16	ASAP RLMN(INT) RLE	0.397418913	1.87E-05
16	ASAP RLMN(INT) S18	0.637147511	2.65E-05

ANEXO D: RESULTADOS COMPLETOS A4**Tabla 48: Resultados A4 – Coreutils m=4, m=8**

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	1.10492028	2.15E-05
4	ASAP FKKP(AP) RLE	1.11342995	1.47E-05
4	ASAP FKKP(AP) RLE	1.17737822	9.61E-06
4	ASAP FKKP(AP) S18	2.20667415	2.15E-05
4	ASAP FKKP(AP) S18	2.21518383	1.54E-05
4	ASAP FKKP(AP) S18	2.27913209	9.55E-06
4	ASAP FKKP(GMR) RLE	1.09116944	1.28E-05
4	ASAP FKKP(GMR) RLE	1.09967911	1.01E-05
4	ASAP FKKP(GMR) RLE	1.16362738	8.15E-06
4	ASAP FKKP(GMR) S18	2.19292331	1.33E-05
4	ASAP FKKP(GMR) S18	2.20143299	1.02E-05
4	ASAP FKKP(GMR) S18	2.26538125	8.18E-06
4	ASAP RLMN(AP) RLE	1.21840099	6.96E-06
4	ASAP RLMN(AP) S18	2.32015487	7.08E-06
4	ASAP RLMN(INT) RLE	1.16656232	6.51E-06
4	ASAP RLMN(INT) S18	2.2683162	6.53E-06
8	ASAP FKKP(AP) RLE	1.10492028	5.30E-05
8	ASAP FKKP(AP) RLE	1.11342995	3.48E-05
8	ASAP FKKP(AP) RLE	1.17737822	2.10E-05
8	ASAP FKKP(AP) S18	2.20667415	5.36E-05
8	ASAP FKKP(AP) S18	2.21518383	3.65E-05
8	ASAP FKKP(AP) S18	2.27913209	2.22E-05
8	ASAP FKKP(GMR) RLE	1.09116944	3.00E-05
8	ASAP FKKP(GMR) RLE	1.09967911	2.31E-05
8	ASAP FKKP(GMR) RLE	1.16362738	1.72E-05
8	ASAP FKKP(GMR) S18	2.19292331	3.10E-05
8	ASAP FKKP(GMR) S18	2.20143299	2.37E-05
8	ASAP FKKP(GMR) S18	2.26538125	1.76E-05
8	ASAP RLMN(AP) RLE	1.21840099	1.43E-05
8	ASAP RLMN(AP) S18	2.32015487	1.52E-05
8	ASAP RLMN(INT) RLE	1.16656232	1.32E-05
8	ASAP RLMN(INT) S18	2.2683162	1.40E-05

Tabla 49: Resultados A4 – Coreutils m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	1.10492028	0.0001041
16	ASAP FKKP(AP) RLE	1.11342995	6.86E-05
16	ASAP FKKP(AP) RLE	1.17737822	3.94E-05
16	ASAP FKKP(AP) S18	2.20667415	0.00010583
16	ASAP FKKP(AP) S18	2.21518383	7.17E-05
16	ASAP FKKP(AP) S18	2.27913209	4.17E-05
16	ASAP FKKP(GMR) RLE	1.09116944	5.81E-05
16	ASAP FKKP(GMR) RLE	1.09967911	4.41E-05
16	ASAP FKKP(GMR) RLE	1.16362738	3.42E-05
16	ASAP FKKP(GMR) S18	2.19292331	6.07E-05
16	ASAP FKKP(GMR) S18	2.20143299	4.63E-05
16	ASAP FKKP(GMR) S18	2.26538125	3.44E-05
16	ASAP RLMN(AP) RLE	1.21840099	2.70E-05
16	ASAP RLMN(AP) S18	2.32015487	2.93E-05
16	ASAP RLMN(INT) RLE	1.16656232	2.46E-05
16	ASAP RLMN(INT) S18	2.2683162	2.67E-05

Tabla 50: Resultados A4 – Einstein.de m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	0.12403159	1.09E-05
4	ASAP FKKP(AP) RLE	0.12420242	9.70E-06
4	ASAP FKKP(AP) RLE	0.12597053	7.99E-06
4	ASAP FKKP(AP) S18	0.16843085	1.08E-05
4	ASAP FKKP(AP) S18	0.16860169	9.41E-06
4	ASAP FKKP(AP) S18	0.1703698	7.96E-06
4	ASAP FKKP(GMR) RLE	0.10003341	8.00E-06
4	ASAP FKKP(GMR) RLE	0.10020424	6.96E-06
4	ASAP FKKP(GMR) RLE	0.10197236	6.29E-06
4	ASAP FKKP(GMR) S18	0.14443268	7.95E-06
4	ASAP FKKP(GMR) S18	0.14460351	7.03E-06
4	ASAP FKKP(GMR) S18	0.14637162	6.14E-06
4	ASAP RLMN(AP) RLE	0.13331418	5.36E-06
4	ASAP RLMN(AP) S18	0.17771344	5.38E-06
4	ASAP RLMN(INT) RLE	0.10670014	4.88E-06
4	ASAP RLMN(INT) S18	0.1510994	4.86E-06
8	ASAP FKKP(AP) RLE	0.12403159	2.31E-05
8	ASAP FKKP(AP) RLE	0.12420242	1.96E-05
8	ASAP FKKP(AP) RLE	0.12597053	1.41E-05
8	ASAP FKKP(AP) S18	0.16843085	2.27E-05
8	ASAP FKKP(AP) S18	0.16860169	1.97E-05
8	ASAP FKKP(AP) S18	0.1703698	1.39E-05
8	ASAP FKKP(GMR) RLE	0.10003341	1.67E-05
8	ASAP FKKP(GMR) RLE	0.10020424	1.45E-05
8	ASAP FKKP(GMR) RLE	0.10197236	1.19E-05
8	ASAP FKKP(GMR) S18	0.14443268	1.63E-05
8	ASAP FKKP(GMR) S18	0.14460351	1.43E-05
8	ASAP FKKP(GMR) S18	0.14637162	1.14E-05
8	ASAP RLMN(AP) RLE	0.13331418	1.01E-05
8	ASAP RLMN(AP) S18	0.17771344	1.08E-05
8	ASAP RLMN(INT) RLE	0.10670014	8.81E-06
8	ASAP RLMN(INT) S18	0.1510994	8.61E-06

Tabla 51: Resultados A4 – Einstein.de m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	0.12403159	5.96E-05
16	ASAP FKKP(AP) RLE	0.12420242	4.98E-05
16	ASAP FKKP(AP) RLE	0.12597053	3.00E-05
16	ASAP FKKP(AP) S18	0.16843085	5.27E-05
16	ASAP FKKP(AP) S18	0.16860169	4.59E-05
16	ASAP FKKP(AP) S18	0.1703698	2.92E-05
16	ASAP FKKP(GMR) RLE	0.10003341	3.56E-05
16	ASAP FKKP(GMR) RLE	0.10020424	3.04E-05
16	ASAP FKKP(GMR) RLE	0.10197236	2.35E-05
16	ASAP FKKP(GMR) S18	0.14443268	3.48E-05
16	ASAP FKKP(GMR) S18	0.14460351	3.04E-05
16	ASAP FKKP(GMR) S18	0.14637162	2.30E-05
16	ASAP RLMN(AP) RLE	0.13331418	2.02E-05
16	ASAP RLMN(AP) S18	0.17771344	2.11E-05
16	ASAP RLMN(INT) RLE	0.10670014	1.72E-05
16	ASAP RLMN(INT) S18	0.1510994	1.69E-05

Tabla 52: Resultados A4 – Einstein.en m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	0.05290492	1.62E-05
4	ASAP FKKP(AP) RLE	0.0530473	1.40E-05
4	ASAP FKKP(AP) RLE	0.05444418	8.94E-06
4	ASAP FKKP(AP) S18	0.0723535	1.69E-05
4	ASAP FKKP(AP) S18	0.07249588	1.32E-05
4	ASAP FKKP(AP) S18	0.07389277	9.36E-06
4	ASAP FKKP(GMR) RLE	0.04839441	1.03E-05
4	ASAP FKKP(GMR) RLE	0.04853679	8.76E-06
4	ASAP FKKP(GMR) RLE	0.04993367	7.25E-06
4	ASAP FKKP(GMR) S18	0.06784299	1.10E-05
4	ASAP FKKP(GMR) S18	0.06798537	8.98E-06
4	ASAP FKKP(GMR) S18	0.06938225	7.52E-06
4	ASAP RLMN(AP) RLE	0.05857873	6.18E-06
4	ASAP RLMN(AP) S18	0.07802731	6.94E-06
4	ASAP RLMN(INT) RLE	0.05297093	5.73E-06
4	ASAP RLMN(INT) S18	0.07241951	6.02E-06
8	ASAP FKKP(AP) RLE	0.05290492	3.33E-05
8	ASAP FKKP(AP) RLE	0.0530473	2.58E-05
8	ASAP FKKP(AP) RLE	0.05444418	1.68E-05
8	ASAP FKKP(AP) S18	0.0723535	3.37E-05
8	ASAP FKKP(AP) S18	0.07249588	2.59E-05
8	ASAP FKKP(AP) S18	0.07389277	1.71E-05
8	ASAP FKKP(GMR) RLE	0.04839441	2.16E-05
8	ASAP FKKP(GMR) RLE	0.04853679	1.76E-05
8	ASAP FKKP(GMR) RLE	0.04993367	1.37E-05
8	ASAP FKKP(GMR) S18	0.06784299	2.25E-05
8	ASAP FKKP(GMR) S18	0.06798537	1.78E-05
8	ASAP FKKP(GMR) S18	0.06938225	1.41E-05
8	ASAP RLMN(AP) RLE	0.05857873	1.18E-05
8	ASAP RLMN(AP) S18	0.07802731	1.23E-05
8	ASAP RLMN(INT) RLE	0.05297093	1.07E-05
8	ASAP RLMN(INT) S18	0.07241951	1.13E-05

Tabla 53: Resultados A4 – Einstein.en m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	0.05290492	6.75E-05
16	ASAP FKKP(AP) RLE	0.0530473	5.14E-05
16	ASAP FKKP(AP) RLE	0.05444418	3.25E-05
16	ASAP FKKP(AP) S18	0.0723535	6.90E-05
16	ASAP FKKP(AP) S18	0.07249588	5.19E-05
16	ASAP FKKP(AP) S18	0.07389277	3.32E-05
16	ASAP FKKP(GMR) RLE	0.04839441	4.39E-05
16	ASAP FKKP(GMR) RLE	0.04853679	3.51E-05
16	ASAP FKKP(GMR) RLE	0.04993367	2.67E-05
16	ASAP FKKP(GMR) S18	0.06784299	4.46E-05
16	ASAP FKKP(GMR) S18	0.06798537	3.57E-05
16	ASAP FKKP(GMR) S18	0.06938225	2.72E-05
16	ASAP RLMN(AP) RLE	0.05857873	2.25E-05
16	ASAP RLMN(AP) S18	0.07802731	2.35E-05
16	ASAP RLMN(INT) RLE	0.05297093	2.05E-05
16	ASAP RLMN(INT) S18	0.07241951	2.10E-05

Tabla 54: Resultados A4 – English m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	0.90870794	1.95E-05
4	ASAP FKKP(AP) RLE	0.91244098	1.47E-05
4	ASAP FKKP(AP) RLE	0.94113279	9.92E-06
4	ASAP FKKP(AP) S18	1.64454531	1.94E-05
4	ASAP FKKP(AP) S18	1.64827835	1.47E-05
4	ASAP FKKP(AP) S18	1.67697016	1.00E-05
4	ASAP FKKP(GMR) RLE	0.87949596	1.20E-05
4	ASAP FKKP(GMR) RLE	0.883229	9.72E-06
4	ASAP FKKP(GMR) RLE	0.91192081	8.00E-06
4	ASAP FKKP(GMR) S18	1.61533333	1.17E-05
4	ASAP FKKP(GMR) S18	1.61906637	9.67E-06
4	ASAP FKKP(GMR) S18	1.64775818	8.05E-06
4	ASAP RLMN(AP) RLE	0.98573473	6.93E-06
4	ASAP RLMN(AP) S18	1.7215721	7.14E-06
4	ASAP RLMN(INT) RLE	0.97649057	6.42E-06
4	ASAP RLMN(INT) S18	1.71232794	6.39E-06
8	ASAP FKKP(AP) RLE	0.90870794	4.10E-05
8	ASAP FKKP(AP) RLE	0.91244098	2.96E-05
8	ASAP FKKP(AP) RLE	0.94113279	1.90E-05
8	ASAP FKKP(AP) S18	1.64454531	4.07E-05
8	ASAP FKKP(AP) S18	1.64827835	2.97E-05
8	ASAP FKKP(AP) S18	1.67697016	1.92E-05
8	ASAP FKKP(GMR) RLE	0.87949596	2.42E-05
8	ASAP FKKP(GMR) RLE	0.883229	1.95E-05
8	ASAP FKKP(GMR) RLE	0.91192081	1.51E-05
8	ASAP FKKP(GMR) S18	1.61533333	2.39E-05
8	ASAP FKKP(GMR) S18	1.61906637	1.90E-05
8	ASAP FKKP(GMR) S18	1.64775818	1.51E-05
8	ASAP RLMN(AP) RLE	0.98573473	1.31E-05
8	ASAP RLMN(AP) S18	1.7215721	1.30E-05
8	ASAP RLMN(INT) RLE	0.97649057	1.20E-05
8	ASAP RLMN(INT) S18	1.71232794	1.17E-05

Tabla 55: Resultados A4 – English m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	0.90870794	8.34E-05
16	ASAP FKKP(AP) RLE	0.91244098	5.93E-05
16	ASAP FKKP(AP) RLE	0.94113279	3.68E-05
16	ASAP FKKP(AP) S18	1.64454531	8.37E-05
16	ASAP FKKP(AP) S18	1.64827835	5.88E-05
16	ASAP FKKP(AP) S18	1.67697016	3.69E-05
16	ASAP FKKP(GMR) RLE	0.87949596	4.94E-05
16	ASAP FKKP(GMR) RLE	0.883229	3.86E-05
16	ASAP FKKP(GMR) RLE	0.91192081	2.96E-05
16	ASAP FKKP(GMR) S18	1.61533333	4.89E-05
16	ASAP FKKP(GMR) S18	1.61906637	3.82E-05
16	ASAP FKKP(GMR) S18	1.64775818	2.93E-05
16	ASAP RLMN(AP) RLE	0.98573473	2.54E-05
16	ASAP RLMN(AP) S18	1.7215721	2.59E-05
16	ASAP RLMN(INT) RLE	0.97649057	2.32E-05
16	ASAP RLMN(INT) S18	1.71232794	2.28E-05

Tabla 56: Resultados A4 – World Leaders m=4, m=8

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
4	ASAP FKKP(AP) RLE	0.38152495	1.62E-05
4	ASAP FKKP(AP) RLE	0.38279456	1.38E-05
4	ASAP FKKP(AP) RLE	0.39438513	1.01E-05
4	ASAP FKKP(AP) S18	0.62245957	1.84E-05
4	ASAP FKKP(AP) S18	0.62372918	1.58E-05
4	ASAP FKKP(AP) S18	0.63531975	1.22E-05
4	ASAP FKKP(GMR) RLE	0.35749055	1.03E-05
4	ASAP FKKP(GMR) RLE	0.35876016	9.10E-06
4	ASAP FKKP(GMR) RLE	0.37035074	7.93E-06
4	ASAP FKKP(GMR) S18	0.59842518	1.24E-05
4	ASAP FKKP(GMR) S18	0.59969479	1.14E-05
4	ASAP FKKP(GMR) S18	0.61128536	9.76E-06
4	ASAP RLMN(AP) RLE	0.40614005	6.83E-06
4	ASAP RLMN(AP) S18	0.64707467	9.02E-06
4	ASAP RLMN(INT) RLE	0.39729867	6.15E-06
4	ASAP RLMN(INT) S18	0.63823329	8.09E-06
8	ASAP FKKP(AP) RLE	0.38152495	3.22E-05
8	ASAP FKKP(AP) RLE	0.38279456	2.60E-05
8	ASAP FKKP(AP) RLE	0.39438513	1.79E-05
8	ASAP FKKP(AP) S18	0.62245957	3.65E-05
8	ASAP FKKP(AP) S18	0.62372918	3.04E-05
8	ASAP FKKP(AP) S18	0.63531975	2.21E-05
8	ASAP FKKP(GMR) RLE	0.35749055	2.03E-05
8	ASAP FKKP(GMR) RLE	0.35876016	1.72E-05
8	ASAP FKKP(GMR) RLE	0.37035074	1.42E-05
8	ASAP FKKP(GMR) S18	0.59842518	3.01E-05
8	ASAP FKKP(GMR) S18	0.59969479	2.15E-05
8	ASAP FKKP(GMR) S18	0.61128536	1.80E-05
8	ASAP RLMN(AP) RLE	0.40614005	1.24E-05
8	ASAP RLMN(AP) S18	0.64707467	1.64E-05
8	ASAP RLMN(INT) RLE	0.39729867	1.10E-05
8	ASAP RLMN(INT) S18	0.63823329	1.54E-05

Tabla 57: Resultados A4 – World Leaders m=16

Fuente: Elaboración Propia

m	estructura	bits/palabra	Tiempo [s]
16	ASAP FKKP(AP) RLE	0.38152495	5.56E-05
16	ASAP FKKP(AP) RLE	0.38279456	4.37E-05
16	ASAP FKKP(AP) RLE	0.39438513	2.94E-05
16	ASAP FKKP(AP) S18	0.62245957	6.37E-05
16	ASAP FKKP(AP) S18	0.62372918	5.14E-05
16	ASAP FKKP(AP) S18	0.63531975	3.71E-05
16	ASAP FKKP(GMR) RLE	0.35749055	3.53E-05
16	ASAP FKKP(GMR) RLE	0.35876016	2.97E-05
16	ASAP FKKP(GMR) RLE	0.37035074	2.42E-05
16	ASAP FKKP(GMR) S18	0.59842518	5.60E-05
16	ASAP FKKP(GMR) S18	0.59969479	3.74E-05
16	ASAP FKKP(GMR) S18	0.61128536	3.23E-05
16	ASAP RLMN(AP) RLE	0.40614005	2.10E-05
16	ASAP RLMN(AP) S18	0.64707467	2.88E-05
16	ASAP RLMN(INT) RLE	0.39729867	1.89E-05
16	ASAP RLMN(INT) S18	0.63823329	2.66E-05