Bachelor's degree in Computer Science and Engineering

Teoría Avanzada de la Computación

2023-2024

Práctica 1

"Máquinas de Turing"

Ignacio Arnaiz Tierraseca – 100428997

Luis Daniel Casais Mezquida – 100429021

Profesor:

Juan Manuel Alonso Weber

Índice

Índice de figuras

Índice de tablas

Parte I

Resumen

El presente informe recoge los trabajos realizados para el desarrollo de la práctica de máquinas de Turing, incluyendo la elaboración de máquinas tanto deterministas como no deterministas, con una única cinta o varias. En cada apartado además de describir el funcionamiento de la máquina planteada, se analiza su rendimeinto en terminos de coste y se indican las pruebas llevadas a cabo.

El desarrollo de las máquinas de Turing se ha llevado a cabo empleado el programa JFLAP[1] que permite a través de una interfaz gráfica, el trabajo de manera intuitiva con máquinas de Turing incluyendo el poder realizar pruebas con múltiples entradas, y llevar acabo ejecuciones detalladas paso a paso.

Para la realización de pruebas y poder obtener el coste de cada máquina en términos de pasos requieridos hasta finalizar su ejecución, se ha empleado en Python un simulador que permite, dadas una máquina y una palabra de entrada determinar el resulatdo de dicha ejecución y el número de pasos requeridos, siendo todos estos resultados guardados para su posterior revisión. Para facilitar la interpretación de los resultados de una forma visual se ha empleado la librería de Pyhton matplotlib[2] a fin de crear diferentes gráficas presentando los resultados obtenidos.

Parte II

Desarrollo

0. Palíndromos

Dado el lenguaje $\Sigma = \{a, b\}$, definimos el palíndromo $x \mid x \in \Sigma*$, $|x| = 2k$, $k \in \mathbb{Z}^+$, $\exists w \in \Sigma*$, $|w| = k \mid x = w \cdot w^{-1}$

0.A. MT Determinista de 1 cinta

Diseño propuesto

El algoritmo de resolución es el siguiente:

- Ciclo:
  1. Si no hay ninguna letra, poner un `1` (son palíndromos) y **parar**.
  2. Borrar una letra en el extremo izquierdo.
  3. Buscar la letra correspondiente en el extremo derecho.
  4. Si la letra coincide:
     1. Borrarla.
     2. Volver al extremo izquierdo.
  5. En caso contrario:
     1. Volver al extremo izquierdo, borrando todas las letras.
     2. Poner un `0` (no son palíndromos) y **parar**.

El diseño de la máquina queda representado en la Figura 1.

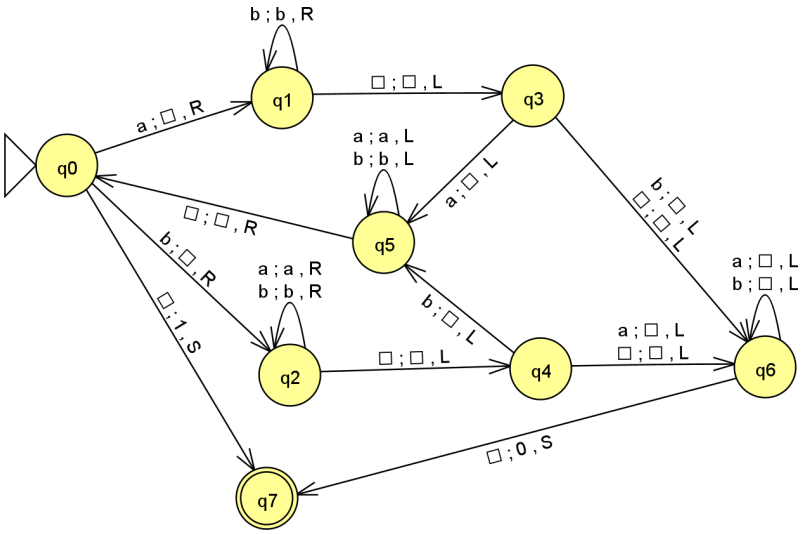

Figura 1: Implementación en JFLAP de MT-0A (`src/tm/MT-0A.jff`)

Peor caso

El peor caso sería cuando es un palíndromo, dado que en el momento en el que una letra no coincida, deja de buscar el resto de letras. Dentro de los palíndromos, el peor caso es cuando es un palíndromo de tamaño par, ya que tiene que hacer una comprobación extra.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la palabra, y midiendo el número de pasos realizados para resolver el problema[1]:

Entrada	n	Pasos
λ	0	1
`aa`	2	6
`abba`	4	15
`abaaba`	6	28
`aaabbaaa`	8	45
`bbaabbaabb`	10	66
`bbbbbaabbbbb`	12	91

Tabla 1: Tamaño de palabra y número de pasos realizados para MT-0A

[1] Los datos se pueden encontrar en `data/MT-0A.csv`.

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[2] basándonos en los datos de la evaluación empírica:

n	**0**	**2**	**4**	**5**	**8**	**10**	**12**
$T(n)$	**1**	**6**	**15**	**28**	**45**	**66**	**91**
$A(n) = T(n) - T(n-2)$		5	9	13	17	21	25
$B(n) = A(n) - A(n-2)$			4	4	4	4	4
$C(n) = B(n) - B(n-2)$				0	0	0	0

Tabla 2: Aplicación de Diferencias Finitas a MT-0A

Al ser constantes las diferencias finitas segundas, y nulas las terceras, podemos aproximar $T(n)$ con un polinomio de segundo orden, es decir, $T(n) = an^2 + bn + c$.

Para obtener los valores de a, b, y c, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 0,\ T(0) = 1 \rightarrow c = 1$$
$$n = 2,\ T(2) = 6 \rightarrow 4a + 2b + c = 6$$
$$n = 4,\ T(4) = 15 \rightarrow 16a + 4b + c = 6$$

Resolviendo, $a = \frac{1}{2}$ y $b = \frac{3}{2}$, por lo que:

$$T_{0A}(n) = 2n^2 + 6n + 2 \tag{1}$$

Cota asintótica

La cota asintótica queda definida[3,4] como:

$$O(n) = k \cdot g(n) \geq T(n) \tag{2}$$

Donde $g(n)$ representa el orden de la cota superior, en este caso $g(n) = n^2$.

Al conocer $T_{0A}(n)$, si asumimos $n_0 = 10$, obtenemos $k \geq \frac{131}{50}$, por lo que la cota asintótica para esta máquina es:

$$O_{0A}(n) = \frac{131}{50} n^2 \tag{3}$$

[2] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits.*
[3] ver 4, pg. 277: *Definition 7.2.*
[4] ver 5, pg. 97: *Definition of Big-Oh.*

Se puede observar la cota en comparación con el coste computacional en la Figura 2 (asumiendo $n_0 = 10$).

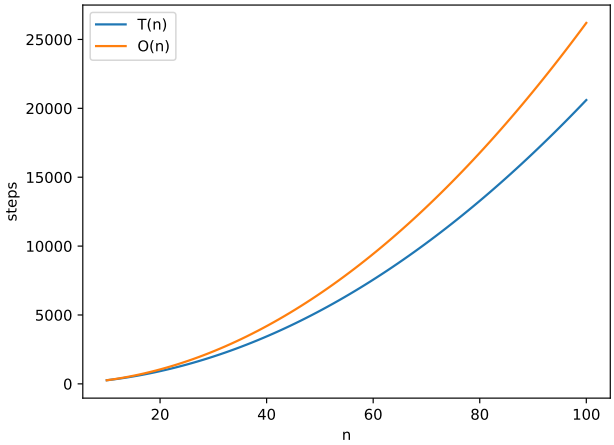

Figura 2: Coste computacional de MT-0A

0.B. MT Determinista de 2 cintas

Diseño propuesto

El algoritmo de resolución es el siguiente:

1. Copiar los símbolos de la cinta 0 a la cinta 1.
2. Rebobinar la cinta 1.
3. Cotejar los símbolos de la cinta 0 y la cinta 1, avanzando ambas cintas paso a paso, y borrando los símbolos.
    1. Si en algún punto no coinciden, rebobinar la cinta 0, limpiar la cinta 1, poner un `0` (no son palíndromos) y **parar**.
4. Poner un `1` (son palíndromos) y **parar**.

El diseño de la máquina queda representado en la Figura 3.

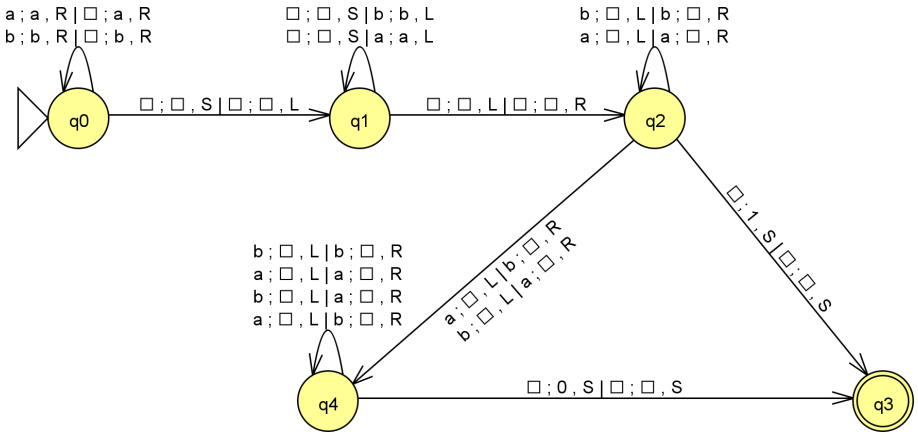

Figura 3: Implementación en JFLAP de MT-0B (`src/tm/MT-0B.jff`)

Peor caso

El peor caso vuelve a ser un palíndromo par, puesto que tiene que comprobar toda la cadena.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la palabra, y midiendo el número de pasos realizados para resolver el problema[5]:

Entrada	n	Pasos
λ	0	3
`aa`	2	9
`abba`	4	15
`abaaba`	6	21
`aaabbaaa`	8	27
`bbaabbaabb`	10	33
`bbbbbaabbbbb`	12	39

Tabla 3: Tamaño de palabra y número de pasos realizados para MT-0B

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[6] basándonos en los datos de la evaluación empírica:

[5] Los datos se pueden encontrar en `data/MT-0B.csv`.

[6] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

n	**0**	**2**	**4**	**5**	**8**	**10**	**12**
$T(n)$	**3**	**9**	**15**	**21**	**27**	**33**	**39**
$A(n) = T(n) - T(n-2)$		6	6	6	6	6	6
$B(n) = A(n) - A(n-2)$			0	0	0	0	0

Tabla 4: Aplicación de Diferencias Finitas a MT-0B

Al ser constantes las diferencias finitas primeras, y nulas las segundas, podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

Para obtener los valores de a y b, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 0,\ T(0) = 3 \rightarrow b = 3$$
$$n = 2,\ T(2) = 9 \rightarrow 2a + b = 9$$

Resolviendo, $a = b = 3$, por lo que:

$$T_{0\text{B}}(n) = 3n + 3 \tag{5}$$

También podemos calcular el coste computacional mediante una evaluación analítica del algoritmo. MT-0B realiza tres recorridos, en el peor de los casos:

1. Copia los datos de la cinta 0 a la cinta 1 $\rightarrow n + 1$ pasos
2. Rebobina la cinta 1 $\rightarrow n + 1$ pasos
3. Coteja la cinta 1 con la cinta 2, en reverso $\rightarrow n + 1$ pasos

Sumando los tres recorridos, nos queda el mismo resultado que en la ecuación 5.

Cota asintótica

Al conocer $T_{0\text{B}}(n)$, podemos afirmar que $g(n) = n$. Si asumimos $n_0 = 10$, obtenemos $k \geq \frac{33}{10}$, por lo que la cota asintótica (definida en la ecuación 2) para esta máquina es:

$$O_{0\text{B}}(n) = \frac{33}{10}n \tag{6}$$

Se puede observar la cota en comparación con el coste computacional en la Figura 4 (asumiendo $n_0 = 10$).

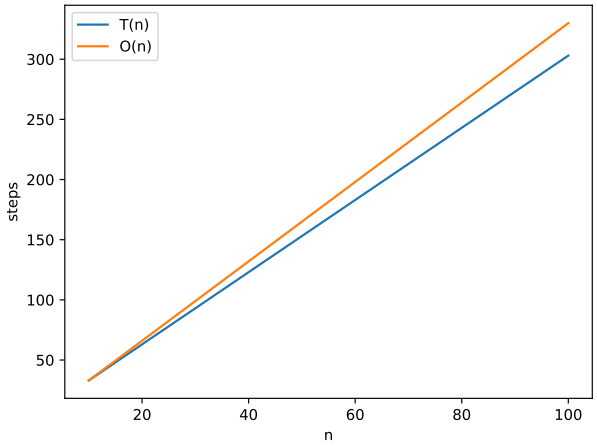

Figura 4: Coste computacional de MT-0B

0.C. MT No Determinista de 2 cintas

Diseño propuesto

El algoritmo de resolución es el siguiente:

1. Copiar la primera mitad de la palabra a la segunda cinta. Al ser no determinista, asumimos a cada paso que hemos llegado a la mitad y pasamos al siguiente paso.
2. Cotejar ambas cintas, leyendo la cinta 1 en reverso y avanzando la cinta 0, y borrando ambas cintas a su paso. En el momento en el que no coincidan los elementos de ambas cintas, acaba sin llegar a un estado final, por lo que la palabra no es aceptada.
3. Poner un 1 en la cinta 0 y **parar**.

El diseño de la máquina queda representado en la Figura 5.

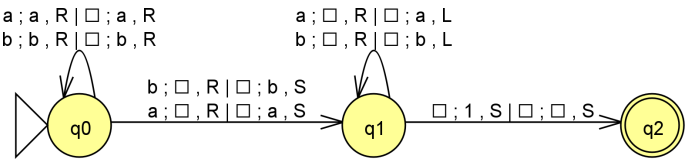

Figura 5: Implementación en JFLAP de MT-0C (`src/tm/MT-0C.jff`)

Peor caso

El peor caso vuelve a ser un palíndromo par, puesto que tiene que comprobar toda la cadena, y en caso de que no sea un palíndromo acaba antes.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la palabra, y midiendo el número de pasos realizados para resolver el problema[7]:

Entrada	n	Pasos
`aa`	2	4
`abba`	4	6
`abaaba`	6	8
`aaabbaaa`	8	10

Tabla 5: Tamaño de palabra y número de pasos realizados para MT-0C

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[8] basándonos en los datos de la evaluación empírica:

n	**2**	**4**	**6**	**8**
$T(n)$	**4**	**6**	**8**	**10**
$A(n) = T(n) - T(n-2)$		2	2	2
$B(n) = A(n) - A(n-2)$			0	0

Tabla 6: Aplicación de Diferencias Finitas a MT-0C

Al ser constantes las diferencias finitas primeras, y nulas las segundas, podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

Para obtener los valores de a y b, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 2,\ T(2) = 4 \rightarrow 2a + b = 4$$
$$n = 4,\ T(4) = 6 \rightarrow 4a + b = 6$$

Resolviendo, $a = 1$ y $b = 2$, por lo que:

$$T_{0C}(n) = n + 2 \tag{8}$$

[7] Los datos se pueden encontrar en `data/MT-0C.csv`.

[8] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

Cota asintótica

Al conocer $T_{0B}(n)$, podemos afirmar que $g(n) = n$. Si asumimos $n_0 = 10$, obtenemos $k \geq \frac{12}{10}$, por lo que la cota asintótica (definida en la ecuación 2) para esta máquina es:

$$O_{0C}(n) = \frac{12}{10}n \tag{9}$$

Se puede observar la cota en comparación con el coste computacional en la Figura 6 (asumiendo $n_0 = 10$).

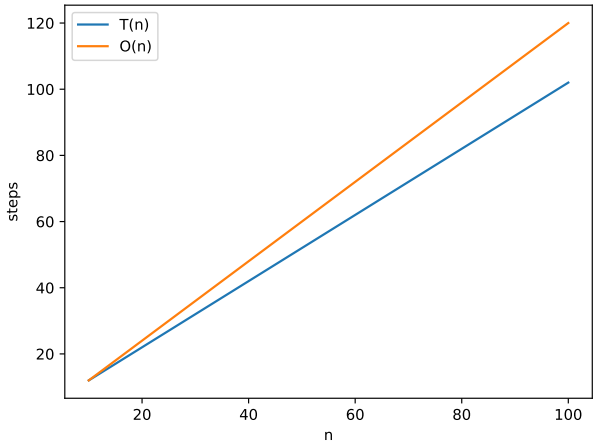

Figura 6: Coste computacional de MT-0C

1. Suma de enteros en base uno

1.A. MT Determinista de 1 cinta

Diseño propuesto

El algoritmo de resolución es el siguiente:

- Ciclo:
  1. Desplazarse hasta el extremo derecho.
  2. Si el elemento es un 1:
     1. Borrarlo.
     2. Volver al extremo izquierdo.
     3. Añadir un 1 en la primera posición.
  3. Si el elemento es un $:
     1. Borrarlo.
     2. Volver al extremo izquierdo.
     3. **Parar**.

El diseño de la máquina queda representado en la Figura 7.

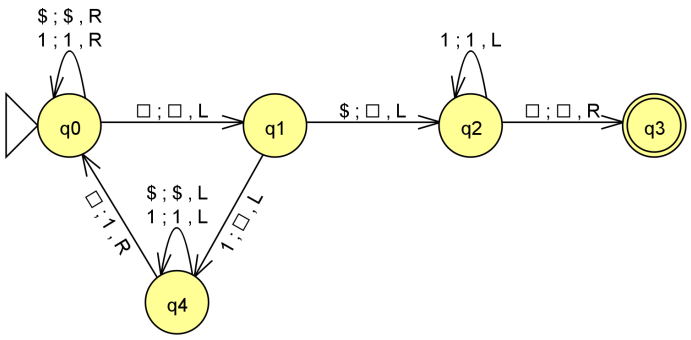

Figura 7: Implementación en JFLAP de MT-1A (`src/tm/MT-1A.jff`)

Peor caso

Dado que el formato de palabra que acepta la Máquina de Turing son dos conjuntos de 1 separados por el simbolo $, el peor caso sería aquel en el que haya una mayor cantidad de 1 a la derecha del $, ya que por cada 1 se requerirá una iteración para eliminarlo de la derecha y añadirlo en la izquierda, por lo que a más 1 más iteraciones serán necesarias.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el número de 1 a la derecha de \$, ya que se trataría del peor caso, y midiendo el número de pasos realizados para resolver el problema[9]:

Entrada	n	Pasos
λ	0	N/A
\$1	1	11
\$11	2	22
\$111	3	37
\$1111	4	56
\$11111	5	79

Tabla 7: Tamaño de palabra y número de pasos realizados para MT-1A

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[10] basándonos en los datos de la evaluación empírica:

n	**1**	**2**	**3**	**4**	**5**
$T(n)$	**11**	**22**	**37**	**56**	**79**
$A(n) = T(n) - T(n-2)$		11	15	19	23
$B(n) = A(n) - A(n-2)$			4	4	4
$C(n) = B(n) - B(n-2)$				0	0

Tabla 8: Aplicación de Diferencias Finitas a MT-1A

Se observa que las diferencias finitas segundas son constantes y nulas las terceras, por lo que podemos aproximar $T(n)$ con un polinomio de segundo orden, es decir, $T(n) = an^2 + bn + c$.

Para obtener los valores de a, b, y c, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 1,\ T(1) = 11 \rightarrow a + b + c = 11$$
$$n = 2,\ T(2) = 22 \rightarrow 4a + 2b + c = 22$$
$$n = 3,\ T(3) = 37 \rightarrow 9a + 3b + c = 37$$

Resolviendo, $a = 2$, $b = 5$ y $b = 4$, por lo que:

$$T_{1A}(n) = 2n^2 + 5n + 4 \tag{11}$$

[9] Los datos se pueden encontrar en `data/MT-1A.csv`.

[10] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

Cota asintótica

Al conocer $T_{1A}(n)$, podemos afirmar que $g(n) = n^2$. Si asumimos $n_0 = 10$, obtenemos $k \geq \frac{127}{50}$, por lo que la cota asintótica (definida en la ecuación 2) para esta máquina es:

$$O_{1A}(n) = \frac{127}{50} n^2 \tag{12}$$

Se puede observar la cota en comparación con el coste computacional en la Figura 8 (asumiendo $n_0 = 10$).

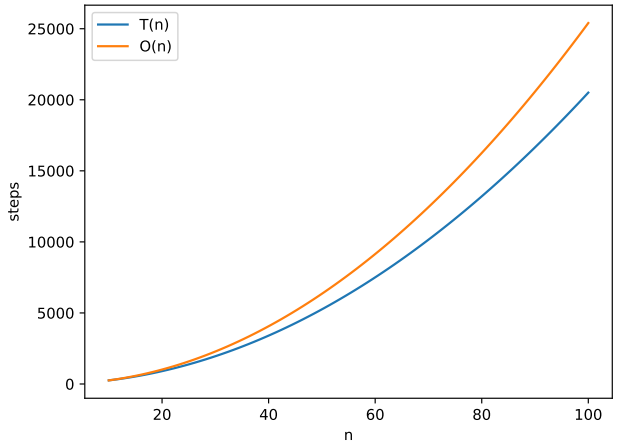

Figura 8: Coste computacional de MT-1A

1.B. MT Determinista de 2 cintas

Diseño propuesto

El algoritmo de resolución es el siguiente:

- Ciclo:
    1. Comenzar desde la izquierda en ambas cintas e ir desplazandose hasta el extremo derecho.
    2. Si se encuentra un 1 en la cinta 0:
        1. Borrarlo de la cinta 0.
        2. Añadir un 1 en la cinta 1.
    3. Si se encuentra un $ en la cinta 0:
        1. Borrarlo de la cinta 0.
        2. No añadir nada en la cinta 1 y retroceder una posisción.

3. Avanzar hacia la derecha en la cinta 0 hasta llegar al final y mantenerse sin moverse en la cinta 1.
4. Por cada 1 en la cinta 1 borrarlo y desplazarse hacia la izquierda en dicha cinta, añadir un 1 en la cinta 0 y avanzar a la derecha
5. Cuando la cinta 1 queda vacia, **parar**.

De esta forma al finalizar la ejecución, la primera cinta contiene el resultado de la suma de ambos números y la segunda cinta queda vacía.

Se ha aplicado una optimización respecto al algortimo original, por el cual una vez se ha copiado el primer número en la segunda cinta, tras eliminarse el $ de la primera cinta, se empiezan a pasar el número de la segunda cinta a la primera desde la izquierda, en vez de copiarlo desde la derecha ya que que en este caso concreto no influye el orden de los dígitos.

El diseño de la máquina queda representado en la Figura 9.

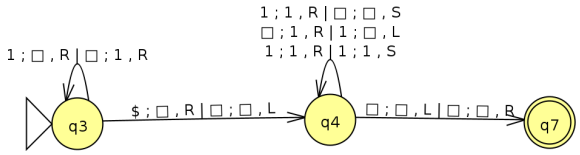

Figura 9: Implementación en JFLAP de MT-1B (`src/tm/MT-1B.jff`)

Peor caso

En este algoritmo, el peor caso será aquel en el que el primer número, esto es a al izquierda del $, tenga un mayor número de dígitos. Esto se debe a que para cada digito a la izquierda del $, este deberá copiarse a la sgunda cinta para posteriormente escribirse de nuevo en la primera.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n la longitud de la cadena de entrada.[11]:

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[12] basándonos en los datos de la evaluación empírica:

[11] Los datos se pueden encontrar en `data/MT-1A.csv`.

[12] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

Entrada	n	Pasos
1$	3	4
11$	4	6
111$	5	8
1111$	6	10
11111$	7	12

Tabla 9: Tamaño de palabra y número de pasos realizados para MT-1B

n	**3**	**4**	**5**	**6**	**7**
$T(n)$	**4**	**6**	**8**	**10**	**12**
$A(n) = T(n) - T(n-2)$		2	2	2	2
$B(n) = A(n) - A(n-2)$			0	0	0

Tabla 10: Aplicación de Diferencias Finitas a MT-1B

Se observa que las diferencias finitas primeras son constantes y nulas las segundas, por lo que podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

Para obtener los valores de a y b, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 3,\ T(3) = 4 \rightarrow 3a + b = 4$$
$$n = 4,\ T(4) = 6 \rightarrow 4a + b = 6$$

Resolviendo, $a = 2$ y $b = -2$, por lo que:

$$T_{1B}(n) = 2n - 2 \tag{14}$$

Cota asintótica

La ecuación 2 no es aplicable para polinomios con términos negativos ($a,\ b,\ c < 0$), cuando n_0 es pequeño ($n \rightarrow 0$), por lo que para aproximar $O(n)$ podemos suprimir esos términos, es decir, $T'_{1B}(n) = 2n$.

Al conocer $T_{1B}(n)$, podemos afirmar que $g(n) = n$. Si asumimos $n_0 = 10$, usando $T'_{1B}(n)$, obtenemos $k \geq 2$, por lo que la cota asintótica (definida en la ecuación 2) para esta máquina es:

$$O_{1A}(n) = 2n \tag{15}$$

Se puede observar la cota en comparación con el coste computacional en la Figura 10 (asumiendo $n_0 = 10$).

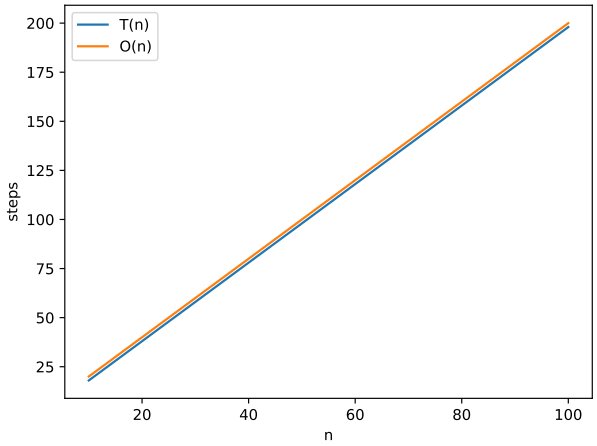

Figura 10: Coste computacional de MT-1B

2. Suma de enteros en base dos

2.A. MT Determinista de 1 cinta

Diseño propuesto

El algoritmo de resolución es el siguiente:

- Ciclo:
  1. Comienzo desde la izquierda, desplazamiento hasta el extremo derecho.
  2. Si el último elemento es un 1:
     1. Borrarlo.
     2. Volver al extremo izquierdo.
     3. Añadir un 1 en la primera posición.
  3. Si el último elemento es un 0:
     1. Borrarlo y sustituirlo por un 1.
     2. Volver hacia la izquierda.
     3. Si se encuentra un 1, sustituirlo por un 0
     4. Si se encuentra un 0
        1. sustituirlo por un 1
        2. Si se encuntra un $
           1. Avanzar hasta el extremo derecho
           2. Volver hacia la izquierda eliminando todos los elementos hasta el $ incluido
           3. **Parar**.
     5. **Parar**.

El diseño de la máquina queda representado en la Figura 11.

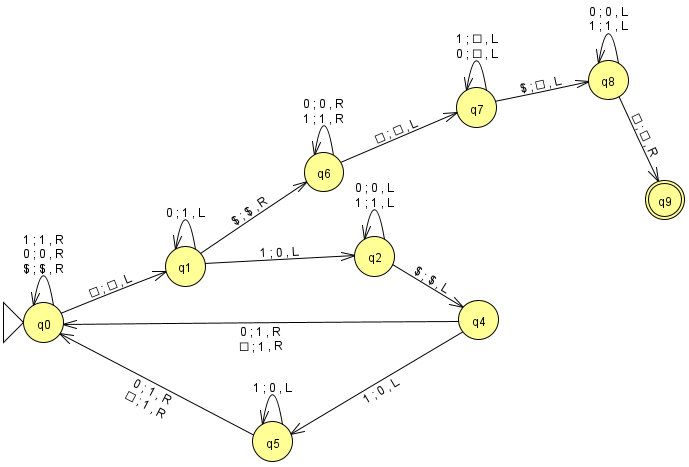

Figura 11: Implementación en JFLAP de MT-2A (`src/tm/MT-2A.jff`)

Peor caso

Dado que la máquina funciona restando de uno en uno al número de la derecha y sumando en la misma medida al número de de la izquierda, el peor caso será aquel en el que se requieran llevar acabo más acarreo de bits puesto que esto implica modificar el valor de todos los dígitos del número al que se le está sumando. Esto ocurrirá cuando para un mismo número de dígitos el número de 1 sea mayor, es es decir, a mayor número de dígitos con valor 1 más ciclos se requerirán.
Del mismo modo al restar a la derecha para sumar a la izquierda, a más dígitos en la derecha más ciclos de suma-resta se requerirán.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la cadena de entrada, compuesta por por 1 a la derecha del $ ya que se trataría del peor caso, y midiendo el número de pasos realizados para resolver el problema[13]:

Entrada	n_d	Pasos
1$1	3	21
1$11	4	48
1$111	5	107
1$1111	6	238
1$11111	7	529

Tabla 11: Tamaño de palabra y número de pasos realizados para MT-2A

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[14] basándonos en los datos de la evaluación empírica:

n	**3**	**4**	**5**	**6**	**7**
$T(n)$	**21**	**48**	**107**	**238**	**529**
$A(n) = T(n) - T(n-2)$		27	59	131	291
$B(n) = A(n) - A(n-2)$			32	72	160
$C(n) = B(n) - B(n-2)$				40	88

Tabla 12: Aplicación de Diferencias Finitas a MT-2A

Se observa que las diferencias finitas primeras, las segundas y terceras, no son nulas ni constantes. Esto se debe a que el funcionamiento de la máquina de Turing incluye recursividad, a pesar de que se han llevado pruebas para tratar de aproximarlo a un polinomio

[13] Los datos se pueden encontrar en `data/MT-2A.csv`.
[14] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

de grado dos o incluso tres, no se ha hallado una aproximación válida, se podría tratar con polinomios de mayor grado pero se requeriría de extraer más datos empíricos y realizar las diferencias finitas, pero aún así no se asegura el poder obtener una aproximación correcta, por lo que no es posible aproximarlo a un polinomio con exactitud.

2.B. MT Determinista de 2 cintas

Diseño propuesto

En el diseño propuesto se copia la palabra de la primera cinta a la segunda, para luego escribir en la primera cinta el número a al derecha del $, restándole uno, y el número a la izquierda del $, sumandole uno. Este ciclo se repite hasta que a la derecha del $ no queda nada (cero), lo que significa que se ha terminado pues el número a la izquierda del $ es la suma de los dos números originales.

El algoritmo de resolución es el siguiente:

- Ciclo:
    1. Avance de derecha a izquierda copiando la palabra a la segunda cinta y borrandolo de la primera hasta llegar al final
    2. Comienza a moverse de izquierda a derecha
    3. Si el ultimo elemento es un 0:
        a) Borra los 0 de la segunda cinta y pone en la primera un 1, hasta encontrar un 1
        b) Borra los 0 de la segunda cinta y pone en la primera un 1, hasta encontrar un 1
        c) Cuando encuentra el primer 1, lo borra de la segunda cinta y pone un 0 en la primera
        d) Sigue avanzando copiando los elemntos de la segunda cinta a la primera hasta llegar al $
    4. Si el último elemento es un 1:
        a) Borra el primer 1 de la segunda cinta y pone un 0 en la primera cinta
        b) Continua copiando los elementos a la primera cinta y borrandolos de la segunda hasta llegar al $
    5. Si el último elemento es un $
        a) Borra el $ de la segunda cinta sin poner nada en la primera cinta
        b) Continua copiando los elementos a la primera cinta y borrandolos de la segunda hasta llegar al final

c) **Parar**.

6. Si encuentra el $ en la segunda cinta, lo copia a la primera cinta y lo borra de la segunda
7. Si el siguiente elemento es un 0:
   a) Borra el 0 de la segunda cinta y pone en la primera un 1
   b) Copia el resto de elemento a la primera cinta elimianndolos de la segunda
8. Si el siguiente elemento es un 1:
   a) Borra los 1 de la segunda cinta y pone 0
   b) Cuando llega a un 0 o al final, lo borra de la segunda cinta y pone un 1 en la primera
   c) Borra los elementos de la segunda cinta y los copia en la primera

El diseño de la máquina queda representado en la Figura 12.

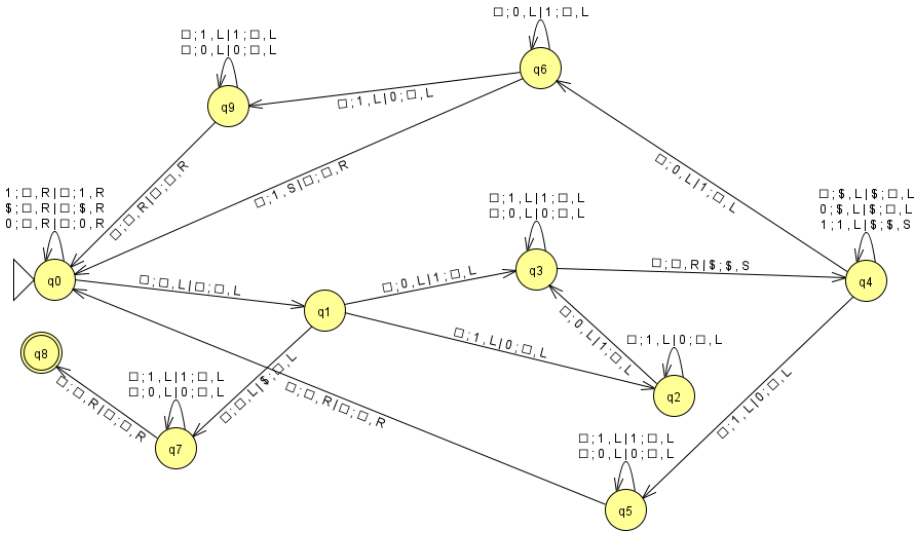

Figura 12: Implementación en JFLAP de MT-2B (`src/tm/MT-2B.jff`)

Peor caso

Dado que la máquina funciona copiando de la primera cinta a la segunda para luego escribir en la primera cinta pero restando a al derecha y sumando a la izquierda una unidad en cada ciclo, el peor caso será aquel en el que se requieran llevar acabo más acarreo de bits puesto que esto implica modificar el valor de todos los dígitos del número al que se le está sumando. Esto ocurrirá cuanto mayor sea el número de la derecha, es decir a más 1 a la derecha del $ más ciclos se requerirán.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la cadena de entrada, compuesta por por 1 a la derecha del \$ ya que se trataría del peor caso, y midiendo el número de pasos realizados para resolver el problema[15]:

Entrada	n_d	Pasos
1\$1	3	17
1\$11	4	46
1\$111	5	119
1\$1111	6	296
1\$11111	7	713

Tabla 13: Tamaño de palabra y número de pasos realizados para MT-2B

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[16] basándonos en los datos de la evaluación empírica:

n	**3**	**4**	**5**	**6**	**7**
$T(n)$	**17**	**46**	**119**	**296**	**713**
$A(n) = T(n) - T(n-2)$		29	73	177	417
$B(n) = A(n) - A(n-2)$			44	104	240
$C(n) = B(n) - B(n-2)$				60	136

Tabla 14: Aplicación de Diferencias Finitas a MT-2B

Se observa que las diferencias finitas primeras, las segundas y terceras, no son nulas ni constantes. Esto se debe a que el funcionamiento de la máquina de Turing incluye recursividad, a pesar de que se han llevado pruebas para tratar de aproximarlo a un polinomio de grado dos o incluso tres, no se ha hallado una aproximación válida, se podría tratar con polinomios de mayor grado pero se requeriría de extraer más datos empíricos y realizar las diferencias finitas, pero aún así no se asegura el poder obtener una aproximación correcta, por lo que no es posible aproximarlo a un polinomio con exactitud.

[15] Los datos se pueden encontrar en `data/MT-2B.csv`.

[16] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

3. Comparativa ejercicios 1 y 2

3.A. Eficiencia algoritmos MT 1 cinta

Para cada algoritmo, es decir el sumador en base uno y el sumador en base dos, con una única cinta se han introducido diferentes valores a fin de comparar la eficiencia, con respecto al número de pasos requieridos para llevar acabo la suma de ambos números. Para cada caso se han introducido los números en el formato que para cada algoritmo es el peor caso.Para cada algoritmo, es decir el sumador en base uno y el sumador en base dos, con una única cinta se han introducido diferentes valores a fin de comparar la eficiencia, con respecto al número de pasos requieridos para llevar acabo la suma de ambos números. Para cada caso se han introducido los números en el formato que para cada algoritmo es el peor caso.

Operación	Entrada base 1	Pasos	Entrada base 2	Pasos
$1+1$	1$1	15	1$1	21
$2+3$	11$111	53	10$11	47
$4+5$	1111$11111	127	100$101	83
$6+10$	111111$1111111111	386	110$1010	172
$14+21$	11111111111111$11111111111111111111	1607	1110$10101	372

Tabla 15: Tamaño de palabra y número de pasos realizados para MT-1A y MT-2A

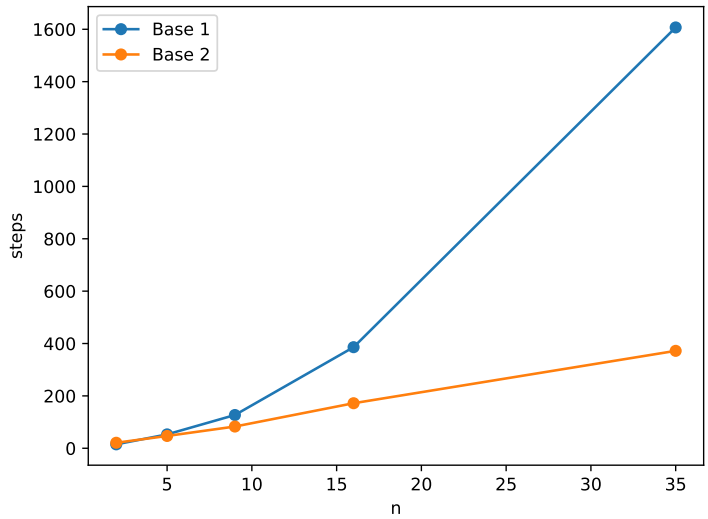

Figura 13: Comparativa coste MT-1A y MT-2A

Se observa que mientras n es un número bajo ambos algoritmos tienen costes similares, pero a medida que n va incrmentandose la diferencia de costes aumenta en mayor

proporción para el algoritmo que suma en base uno.
Esto se debe a que dicho algoritmo en cada ciclo debe desplazarse desde el inicio de la cadena donde se restará un 1, hasta el final para añadir otro 1, y al ser en base uno a mayor sea el número, el tamaño de la cadena incrmentará en mayor medida. Mientras que en base dos, aunque los números sean mayores, el tamaño de la cadena aumenta en una proporción menor.

Esto también es observable comparando las complejidades, donde para el algortimo de suma en base uno dicha complejidad es cuadrática y queda definida por un polinomio de grado dos, mientras que para el algortimo de suma en base dos no posible definir la complejidad a través de un polinomio pues tiene un comportamiento que incluye recursividad, pero sigue una tendencia similar a una expresión cudratica que crece en menor medida al algoritmo en base uno.

3.B. Eficiencia algoritmos MT 2 cintas

Al igual que en el apartado previo, para cada algortimo tanto la suma en base uno como en base dos, en este caso empleando un automata de dos cintas, se han llevado acabo una serie de pruebas con las mismas palabras que en las pruebas realizadas con los automatas de una cinta, a fin de poder comparar la eficiencia en terminos de coste. Para cada caso se han introducido los números en el formato que para cada algoritmo es el peor caso.

Operación	Entrada base 1	Pasos	Entrada base 2	Pasos
$1+1$	1\$1	5	1\$1	17
$2+3$	111\$11	10	10\$11	50
$4+5$	11111\$1111	16	100\$101	93
$6+10$	111111111\$111111	28	110\$1010	204
$14+21$	11111111111111111111\$111111111111111	58	1110\$10101	507

Tabla 16: Tamaño de palabra y número de pasos realizados para MT-1B y MT-2B

Se observa que desde un inicio, ya los costes en terminos de pasos requeridos aumentan significativamente más en el algoritmo de suma en base dos, haciendose cada vez más notable a medida que n aumenta.
Esto se debe a que el algortimo de base uno tranfiere todos los unos a la izquierda del \$ a la segunda cinta para luego ir borrandolos de uno en uno y sumándolos al final de la primera cinta, mientras que el algortimo de base dos debe ir desplazandose de izquierda a derecha sumando uno al número a la derecha del \$, y restandoselo al de la izquierda, lo que requiere varios ciclos de avanzar y retroceder.
Esto se corrobora observando las expresiones de la complejidad, donde para el algoritmo de base uno se puede expresar con un polinomio de grado uno, pues tiene una complejidad

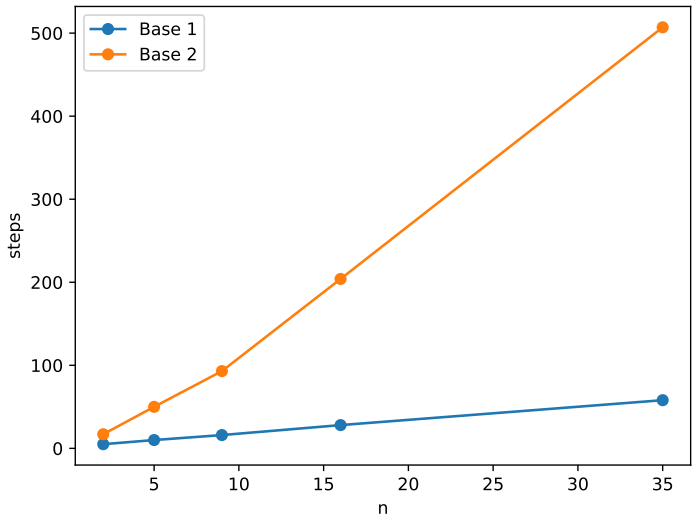

Figura 14: Comparativa coste MT-1B y MT-2B

lineal. Mientras que para el algoritmo de base dos no es posible extraer un polinomio, ya que tiene un comportamiento que incluye recursividad, pero cuyo comportamiento sería más próximo a una tendencia cuadratica.

4. Suma de enteros en base tres

4.A. MT Determinista de 2 cintas

Diseño propuesto

En el diseño propuesto se copia la palabra de la primera cinta a la segunda, para luego escribir en la primera cinta el número a al derecha del $ restandole uno y el número a la izquierda del $ sumandole uno. Este ciclo se repite hasta que a la derecha del $ no queda nada (cero), lo que significa que se ha terminado pues el número a la izquierda del $ es la suma de los dos números originales.

El algoritmo de resolución es el siguiente

- Ciclo:
    1. Avance de derecha a izquierda copiando la palabra a la segunda cinta y borrandolo de la primera hasta llegar al final
    2. Comienza a moverse de izquierda a derecha
    3. Si el ultimo elemento es un `0`:
        1. Borra los 0 de la segunda cinta y pone en la primera un `2`, hasta encontrar un `1` o un `2`
        2. Si encuentra un 1, Lo borra de la segunda cinta y pone un `0` en la primera
    4. Si encuentra un `2`, lo borra de la segunda cinta y pone un `1` en la primera.
    5. Sigue avanzando copiando los elemntos de la segunda cinta a la primera hasta llegar al $
    6. Si el ultimo elemento es un `1`
        1. Borra el primer `1` de la segunda cinta y pone un `0` en la primera cinta
        2. Continua copiando los elementos a la primera cinta y borrandolos de la segunda hasta llegar al $
    7. Si el ultimo elemento es un `2`:
        1. Borra el primer 2 de la segunda cinta y pone un 1 en la primera cinta
        2. Continua copiando los elementos a la primera cinta y borrandolos de la segunda hasta llegar al $
    8. Si el último elemento es un $:
        1. Borra el $ de la segunda cinta sin poner nada en la primera cinta
        2. Continua copiando los elementos a la primera cinta y borrandolos de la segunda hasta llegar al final
        3. **Parar**.

9. Si encuentra el $ en la segunda cinta, lo copia a la primera cianta y lo borra de la segunda.
10. Si el siguiente elemento es un `0`:
    1. Borra el `0` de la segunda cinta y pone en la primera un `1`
    2. Copia el resto de elemento a la primera cinta eliminándolos de la segunda
11. Si el siguiente elemento es un `1`:
    1. Borra el `1` de la segunda cinta y pone en la primera un `2`
    2. Copia el resto de elementos a la primera cinta eliminándolos de la segunda
12. Si el siguiente elemento es un `2`:
    1. Borra los `2` de la segunda cinta y pone `0`
    2. Cuando llega a un `0` o al final, lo borra de la segunda cinta y pone un 1 en la primera
    3. Borra los elementos de la segunda cinta y los copia en la primera cinta

El diseño de la máquina queda representado en la Figura 15.

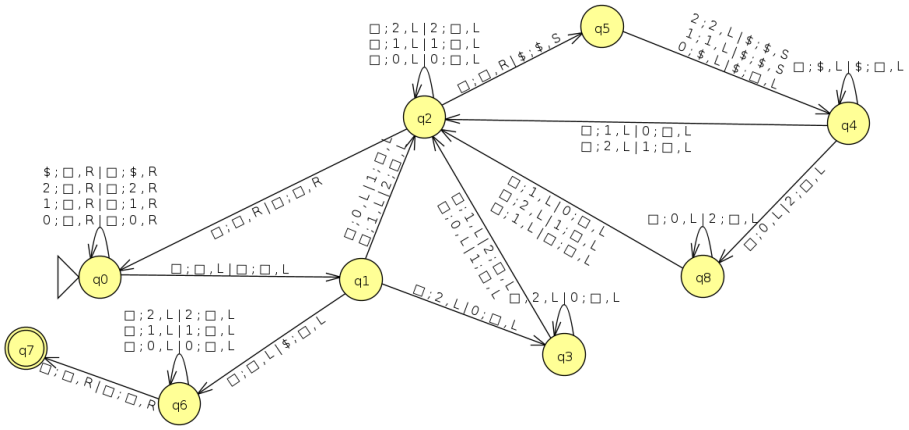

Figura 15: Implementación en JFLAP de MT-4A (`src/tm/MT-4A.jff`)

Peor caso

Dado que la máquina funciona restando de uno en uno al número de la derecha y sumando en la misma medida al número de de la izquierda, el peor caso será aquel en el que se requieran llevar acabo más acarreo de bits puesto que esto implica modificar el valor de todos los dígitos del número al que se le está sumando. Esto ocurrirá cuando para un mismo número de dígitos el número de 2 sea mayor, es decir a mayor número de dígitos con valor 2 más ciclos se requerirán.
Del mismo modo al restar a la derecha para sumar a la izquierda, a más dígitos en la derecha más ciclos de suma/resta serán requeridos.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la cadena de entrada, compuesta por por 2 a la derecha del $ ya que se trataría del peor caso, y midiendo el número de pasos realizados para resolver el problema[17]:

Entrada	n_d	Pasos
2$2	3	30
2$22	4	118
2$222	5	446
2$2222	6	1638
2$22222	7	5854

Tabla 17: Tamaño de palabra y número de pasos realizados para MT-4A

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[18] basándonos en los datos de la evaluación empírica:

n	**3**	**4**	**5**	**6**	**7**
$T(n)$	**30**	**118**	**446**	**1638**	**5854**
$A(n) = T(n) - T(n-2)$		88	328	1192	4216
$B(n) = A(n) - A(n-2)$			240	864	3024
$C(n) = B(n) - B(n-2)$				624	2160

Tabla 18: Aplicación de Diferencias Finitas a MT-4A

Se observa que las diferencias finitas primeras, las segundas y terceras, no son nulas ni constantetes. Esto se debe a que el funcionamiento de la máquina de Turing incluye recursividad, lo que hace que no sea lineal, por lo que no es posible aproximarlo a un polinomio con exactitud.

Evaluación respecto a los apartados 1 y 2

Se han llevado acabo una serie de tests con diferentes operaciones empleando los tres algoritmos con máquinas de dos cintas, en todos los casos las operaciones se han planteado con el formato de peor caso para poder llevar a cabo una comparación más rigurosa.

[17] Los datos se pueden encontrar en `data/MT-4A.csv`.

[18] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

Operación	Entrada base 1	Pasos
$1+1$	1\$1	5
$2+3$	111\$11	10
$4+5$	11111\$1111	16
$6+10$	1111111111\$111111	28
$14+21$	11111111111111111111\$11111111111111	58

Operación	Entrada base 2	Pasos	Entrada base 3	Pasos
$1+1$	1\$1	17	1\$1	15
$2+3$	10\$11	50	2\$10	43
$4+5$	100\$101	93	11\$12	75
$6+10$	110\$1010	204	20\$101	162
$14+21$	1110\$10101	507	112\$210	384

Tabla 19: Tamaño de palabra y número de pasos realizados para MT-1B, MT-2B y MT-4A

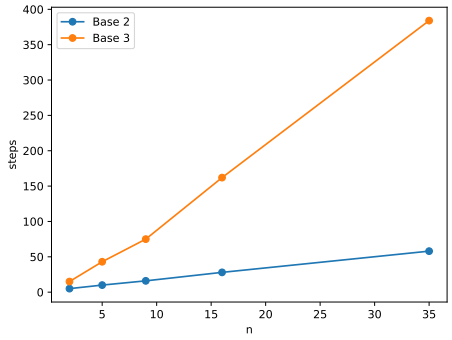

Figura 16: Comparativa apartados 1 y 3

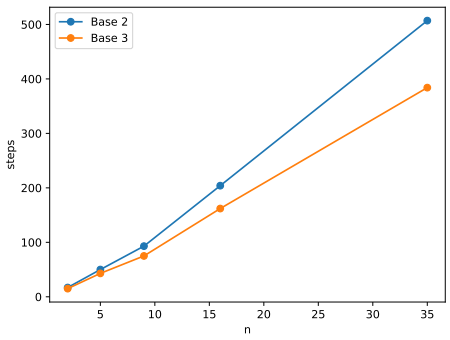

Figura 17: Comparativa apartados 2 y 3

Se observa que en terminos de coste el algortimo de base dos y base tres son basante próximos, pues en ambos casos para realizar una misma operación el tamaño de palabra y el funcionamiento de el algoritmo son similares, teniendo una complejidad que no es posible expresar como una expresión polinómica al tener un comportamiento recursivo.

En contraposición el algortimo en base uno, muestra tener un coste notablemente inferior al de base tres, pues su expresión de complejidad queda definida por un polinomio de grado uno, se trata de un crecimiento lineal frente a un crecimiento con tendecia cuadrática del algortimo de base tres.

5. Palabras de estructura triplicada (I)

Dado el alfabeto $\Sigma = \{a, b\}$ y la palabra $x \in \Sigma$, $\exists w \mid x = w \cdot w^{-1} \cdot w'$ $(w = w')$.

5.A. MT Multicinta Determinista

Diseño propuesto

El algoritmo de resolución es el siguiente:

1. Leer `a` o `b` en la cinta 0 tres veces, y poner un `1` en la cinta 1.
2. Si no es el final de la cinta, volver al paso anterior. En caso afirmativo, tendremos en la cinta 1 el tamaño de la palabra $w = n/3$, en base uno.
3. Copiar w' a la cinta 1. Por cada `1` en la cinta 1:
    1. Copiar la letra de la cinta 0 a la cinta 1, y borrarla de la cinta 0, sobreescribiendo los `1` y moviendo a la izquierda ambas cintas.
4. Comprobar w^{-1}. Por cada letra de la cinta 1:
    1. Validar que las letras de ambas cintas son iguales, mover la cinta 0 a la izquierda, borrando las letras, y la cinta 1 a la derecha.
5. Comprobar w. Por cada letra de la cinta 1:
    1. Validar que las letras de ambas cintas son iguales y mover ambas cintas a la izquierda, borrando las letras.
6. Poner un `1` en la cinta 0 y **parar**.

El diseño de la máquina queda representado en la Figura 18.

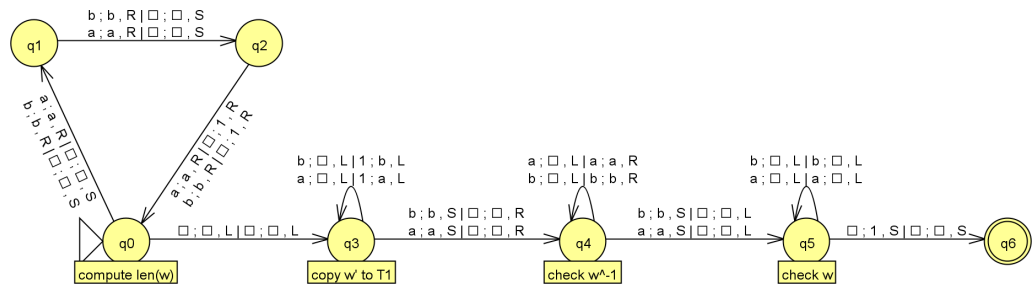

Figura 18: Implementación en JFLAP de MT-5A (`src/tm/MT-5A.jff`)

Peor caso

El peor caso es cuando es una palabra de la gramática ($w \cdot w^{-1} \cdot w$), puesto que tiene que comprobarla entera. La estructura de w es irrelevante, puesto que las transiciones no dependen de cómo esté formada.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la palabra, y midiendo el número de pasos realizados para resolver el problema[19]:

Entrada	n	Pasos
`aaa`	3	10
`aaaaaa`	6	16
`aaaaaaaaa`	9	22
`aaaaaaaaaaaa`	12	28

Tabla 20: Tamaño de palabra y número de pasos realizados para MT-5A

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[20] basándonos en los datos de la evaluación empírica:

n	**3**	**6**	**9**	**12**
$T(n)$	**10**	**16**	**22**	**28**
$A(n) = T(n) - T(n-2)$		6	6	6
$B(n) = A(n) - A(n-2)$			0	0

Tabla 21: Aplicación de Diferencias Finitas a MT-5A

Al ser constantes las diferencias finitas primeras, y nulas las segundas, podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

Para obtener los valores de a y b, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 3,\ T(3) = 10 \rightarrow 3a + b = 10$$
$$n = 6,\ T(6) = 16 \rightarrow 6a + b = 16$$

Resolviendo, $a = 2$ y $b = 4$, por lo que:

[19] Los datos se pueden encontrar en `data/MT-5A.csv`.

[20] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

$$T_{5A}(n) = 2n + 4 \tag{17}$$

Cota asintótica

Al conocer $T_{5A}(n)$, podemos afirmar que $g(n) = n$. Si asumimos $n_0 = 10$, obtenemos $k \geq \frac{12}{5}$, por lo que la cota asintótica (definida en la ecuación 2) para esta máquina es:

$$O_{5A}(n) = \frac{12}{5}n \tag{18}$$

Se puede observar la cota en comparación con el coste computacional en la Figura 19 (asumiendo $n_0 = 10$).

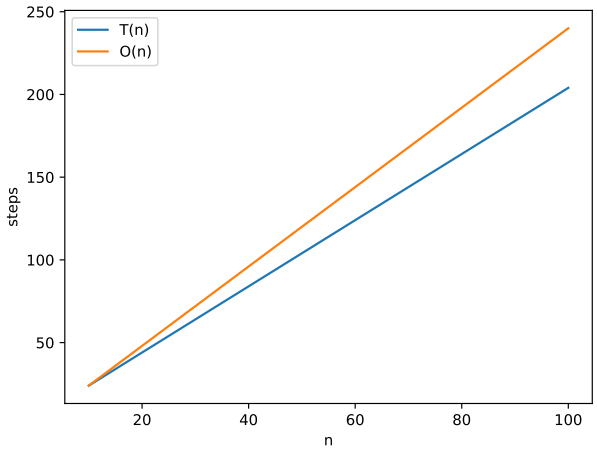

Figura 19: Coste computacional de MT-5A

Inclusión de una tercera cinta

El diseño se basa en el uso de una MT de dos cintas, dado que la inclusión de una tercera cinta no proporcionaría ningún beneficio.

En este problema no sólo hay que localizar w (en este caso, dividimos la entrada entre tres y cogemos el último elemento), sino que hay que validar w y w^{-1}.
Con dos cintas lo comprobamos secuencialmente, y la idea sería comprobar w^{-1} en la cinta 0 mientras comprobamos w en la cinta 2. Podríamos copiar la entrada también a la tercera cinta, pero no habría forma de dejar el cabezal de la tercera cinta apuntando al principio de la entrada (al principio de w) sin realizar pasos extras, puesto que nuestro algoritmo realiza una pasada para calcular la longitud de w y la pasada de vuelta para comprobar las palabras.

5.B. MT Multicinta No Determinista

Diseño propuesto

El algoritmo de resolución es el siguiente:

1. Copiar w a la cinta 1, dejando el cabezal 0 al principio de w^{-1} y el cabezal 1 al final de w. Dado que es una máquina no determinista, asumiremos a cada paso que hemos copiado toda la palabra.
2. Comprobar w^{-1}. Por cada letra de la cinta 1:
    1. Validar que las letras de ambas cintas son iguales, mover la cinta 1 a la izquierda, borrando las letras, y la cinta 0 a la derecha.
3. Comprobar w. Por cada letra de la cinta 1:
    1. Validar que las letras de ambas cintas son iguales y mover ambas cintas a la izquierda, borrando las letras.
4. Poner un `1` en la cinta 0 y **parar**.

El diseño de la máquina queda representado en la Figura 20.

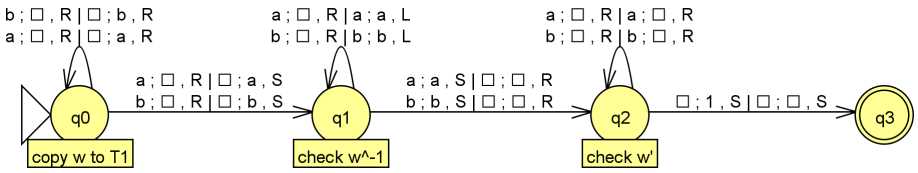

Figura 20: Implementación en JFLAP de MT-5B (`src/tm/MT-5B.jff`)

Peor caso

Al igual que en la determinista, el peor caso es cuando es una palabra de la gramática ($w \cdot w^{-1} \cdot w'$), independientemente de la estructura.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la palabra, y midiendo el número de pasos realizados para resolver el problema[21]:

[21] Los datos se pueden encontrar en `data/MT-5B.csv`.

Entrada	n	Pasos
`aaa`	3	5
`aaaaaa`	6	8
`aaaaaaaaa`	9	11
`aaaaaaaaaaaa`	12	14

Tabla 22: Tamaño de palabra y número de pasos realizados para MT-5B

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[22] basándonos en los datos de la evaluación empírica:

n	**3**	**6**	**9**	**12**
$T(n)$	**5**	**8**	**11**	**14**
$A(n) = T(n) - T(n-2)$		3	3	3
$B(n) = A(n) - A(n-2)$			0	0

Tabla 23: Aplicación de Diferencias Finitas a MT-5B

Al ser constantes las diferencias finitas primeras, y nulas las segundas, podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

Para obtener los valores de a y b, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 3,\ T(3) = 5 \rightarrow 3a + b = 5$$
$$n = 6,\ T(6) = 8 \rightarrow 6a + b = 8$$

Resolviendo, $a = 1$ y $b = 2$, por lo que:

$$T_{5B}(n) = n + 2 \tag{20}$$

Cota asintótica

Al conocer $T_{5B}(n)$, podemos afirmar que $g(n) = n$. Si asumimos $n_0 = 10$, obtenemos $k \geq \frac{6}{5}$, por lo que la cota asintótica (definida en la ecuación 2) para esta máquina es:

$$O_{5B}(n) = \frac{6}{5}n \tag{21}$$

Se puede observar la cota en comparación con el coste computacional en la Figura 21 (asumiendo $n_0 = 10$).

[22] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

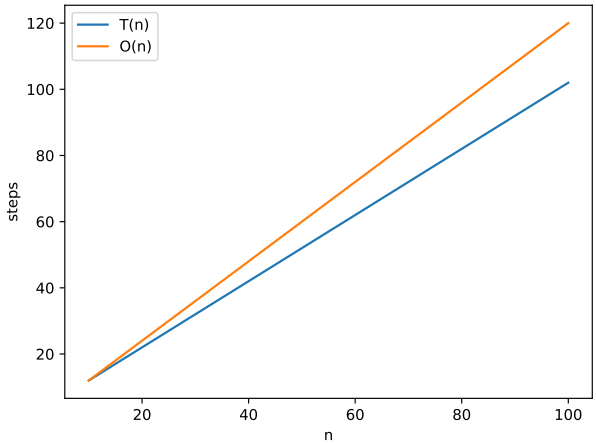

Figura 21: Coste computacional de MT-5B

Inclusión de una tercera cinta

El diseño se basa en el uso de una MT de dos cintas, dado que la inclusión de una tercera cinta no proporcionaría ningún beneficio, por un motivo similar al de la máquina determinista: hacemos una única pasada de la entrada.

6. Palabras de estructura triplicada (II)

Dado el alfabeto $\Sigma = \{a, b\}$ y la palabra $x \in \Sigma$, $\exists w \mid x = w \cdot w' \cdot w''$ $(w = w' = w'')$.

6.A. MT Multicinta Determinista

Diseño propuesto

El algoritmo de resolución es el siguiente:

1. Leer `a` o `b` en la cinta 0 tres veces, y poner un `1` en la cinta 1.
2. Si no es el final de la cinta, volver al paso anterior. En caso afirmativo, tendremos en la cinta 1 el tamaño de la palabra $w = n/3$, en base uno.
3. Copiar w'' a la cinta 1. Por cada `1` en la cinta 1:
    1. Copiar la letra de la cinta 0 a la cinta 1, y borrarla de la cinta 0, sobreescribiendo los `1` y moviendo a la izquierda ambas cintas.
4. Poner el cabezal 1 al final de w.
5. Comprobar w'. Por cada letra de la cinta 1:
    1. Validar que las letras de ambas cintas son iguales, moviendo la cinta 0 a la izquierda, borrando las letras, y la cinta 1 a la derecha.
6. Poner el cabezal 1 al final de w.
7. Comprobar w. Por cada letra de la cinta 1:
    1. Validar que las letras de ambas cintas son iguales moviendo la cinta 0 a la izquierda, y la cinta 1 a la derecha, borrando ambas cintas.
8. Poner un `1` en la cinta 0 y **parar**.

El diseño de la máquina queda representado en la Figura 22.

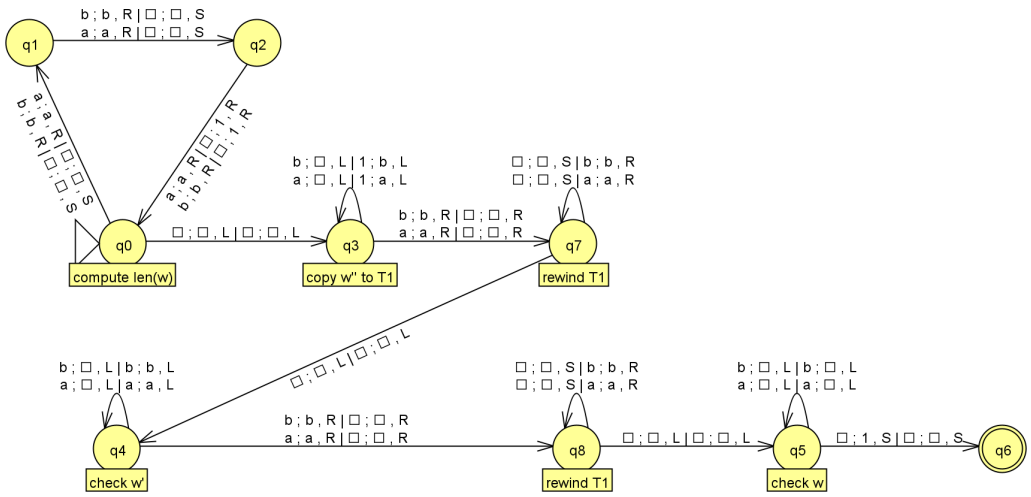

Figura 22: Implementación en JFLAP de MT2T-6A (`src/tm/MT2T-6A.jff`)

Peor caso

El peor caso es cuando es una palabra de la gramática ($w \cdot w' \cdot w''$), puesto que tiene que comprobarla entera. La estructura de w es irrelevante, puesto que las transiciones no dependen de cómo esté formada.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la palabra, y midiendo el número de pasos realizados para resolver el problema[23]:

Entrada	n	Pasos
`aaa`	3	14
`aaaaaa`	6	22
`aaaaaaaaa`	9	30
`aaaaaaaaaaaa`	12	38

Tabla 24: Tamaño de palabra y número de pasos realizados para MT2T-6A

Coste computacional

Para obtener el coste computacional del algoritmo, aplicaremos Diferencias Finitas,[24] basándonos en los datos de la evaluación empírica:

[23] Los datos se pueden encontrar en `data/MT2T-6A.csv`.

[24] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

n	**3**	**6**	**9**	**12**
$T(n)$	**14**	**22**	**30**	**38**
$A(n) = T(n) - T(n-2)$		8	8	8
$B(n) = A(n) - A(n-2)$			0	0

Tabla 25: Aplicación de Diferencias Finitas a MT2T-6A

Al ser constantes las diferencias finitas primeras, y nulas las segundas, podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

Para obtener los valores de a y b, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 3,\ T(3) = 14 \rightarrow 3a + b = 14$$
$$n = 6,\ T(6) = 22 \rightarrow 6a + b = 22$$

Resolviendo, $a = \frac{8}{3}$ y $b = 6$, por lo que:

$$T_{2T-6A}(n) = \frac{8}{3}n + 6 \tag{23}$$

Cota asintótica

Al conocer $T_{6A}(n)$, podemos afirmar que $g(n) = n$. Si asumimos $n_0 = 10$, obtenemos $k \geq \frac{49}{15}$, por lo que la cota asintótica (definida en la ecuación 2) para esta máquina es:

$$O_{2T-6A}(n) = \frac{49}{15}n \tag{24}$$

Se puede observar la cota en comparación con el coste computacional en la Figura 23 (asumiendo $n_0 = 10$).

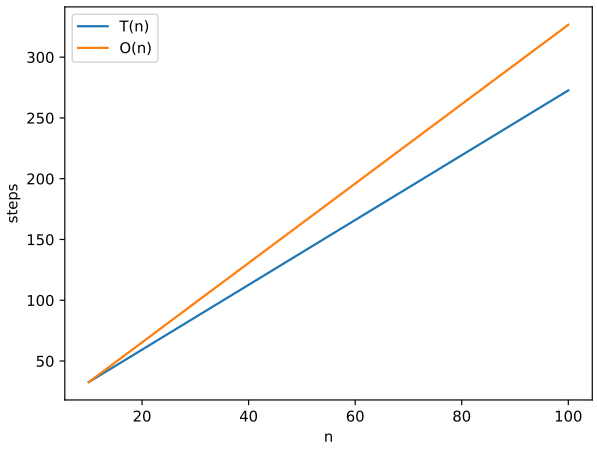

Figura 23: Coste computacional de MT2T-6A

Inclusión de una tercera cinta

El diseño se basa en el uso de una MT de dos cintas, pero la inclusión de una tercera cinta proporcionaría beneficios.

Se puede utilizar la tercera cinta para copiar w en dos cintas en vez de una, y así evitándonos recorrer w una de las veces para recolocar el cabezal.

El diseño de la máquina queda representado en la Figura 24.

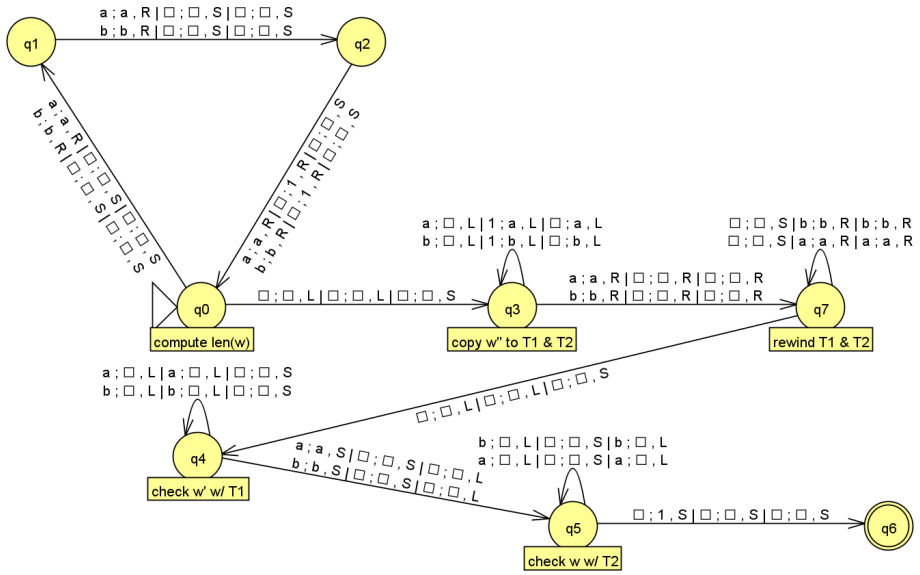

Figura 24: Implementación en JFLAP de MT3T-6A (`src/tm/MT3T-6A.jff`)

Realizamos la evaluación empírica[25]:

Entrada	n	Pasos
`aaa`	3	12
`aaaaaa`	6	19
`aaaaaaaaa`	9	26
`aaaaaaaaaaaa`	12	33

Tabla 26: Tamaño de palabra y número de pasos realizados para MT3T-6A

Calculamos el coste computacional mediante Diferencias Finitas:

n	**3**	**6**	**9**	**12**
$T(n)$	**12**	**19**	**26**	**33**
$A(n) = T(n) - T(n-2)$		7	7	7
$B(n) = A(n) - A(n-2)$			0	0

Tabla 27: Aplicación de Diferencias Finitas a MT3T-6A

[25] Los datos se pueden encontrar en `data/MT3T-6A.csv`.

Al ser constantes las diferencias finitas primeras, y nulas las segundas, podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

Para obtener los valores de a y b, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 3,\ T(3) = 12 \rightarrow 3a + b = 12$$
$$n = 6,\ T(6) = 19 \rightarrow 6a + b = 19$$

Resolviendo, $a = \frac{7}{3}$ y $b = 5$, por lo que:

$$T_{3T-6A}(n) = \frac{7}{3}n + 5 \tag{26}$$

Podemos comparar ambas máquinas gráficamente (con $n_0 = 3$) en la Figura 25, observando que la de tres cintas tiene un coste menor:

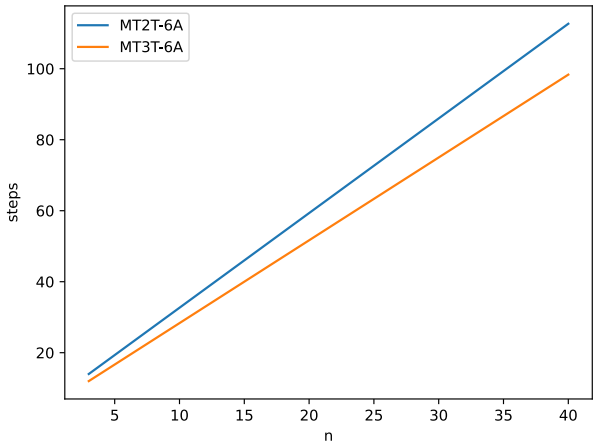

Figura 25: Comparación del coste computacional entre MT2T-6A y MT3T-6A

6.B. MT Multicinta No Determinista

Diseño propuesto

El algoritmo de resolución es el siguiente:

1. Copiar w a la cinta 1, dejando el cabezal 0 al principio de w' y el cabezal 1 al final de w. Dado que es una máquina no determinista, asumiremos a cada paso que hemos copiado toda la palabra.

2. Comprobar w'. Por cada letra de la cinta 1:

1. Validar que las letras de ambas cintas son iguales, moviendo ambas cintas a la derecha, borrando las letras de la cinta 0.

3. Poner el cabezal 1 al principio de w.

4. Comprobar w''. Por cada letra de la cinta 1:

   1. Validar que las letras de ambas cintas son iguales, moviendo ambas cintas a la derecha, borrando las letras de la cinta 0.

5. Poner un `1` en la cinta 0 y **parar**.

El diseño de la máquina queda representado en la Figura 26.

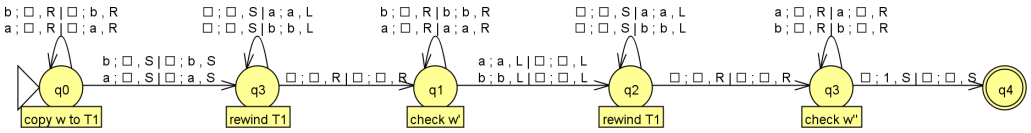

Figura 26: Implementación en JFLAP de MT2T-6B (`src/tm/MT2T-6B.jff`)

Peor caso

Al igual que en la determinista, el peor caso es cuando es una palabra de la gramática ($w \cdot w' \cdot w''$), independientemente de la estructura.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la palabra, y midiendo el número de pasos realizados para resolver el problema[26]:

n	**3**	**6**	**9**	**12**
$T(n)$	**9**	**14**	**19**	**24**
$A(n) = T(n) - T(n-2)$		5	5	5
$B(n) = A(n) - A(n-2)$			0	0

Tabla 28: Tamaño de palabra y número de pasos realizados para MT2T-6B

Al ser constantes las diferencias finitas primeras, y nulas las segundas, podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

Para obtener los valores de a y b, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 3,\ T(3) = 9 \rightarrow 3a + b = 9$$
$$n = 6,\ T(6) = 14 \rightarrow 6a + b = 14$$

[26] Los datos se pueden encontrar en `data/MT2T-6B.csv`.

Resolviendo, $a = \frac{5}{3}$ y $b = 4$, por lo que:

$$T_{2T-6B}(n) = \frac{5}{3}n + 4 \tag{28}$$

Cota asintótica

Al conocer $T_{2T-6B}(n)$, podemos afirmar que $g(n) = n$. Si asumimos $n_0 = 10$, obtenemos $k \geq \frac{31}{15}$, por lo que la cota asintótica (definida en la ecuación 2) para esta máquina es:

$$O_{2T-6B}(n) = \frac{31}{15}n \tag{29}$$

Se puede observar la cota en comparación con el coste computacional en la Figura 27 (asumiendo $n_0 = 10$).

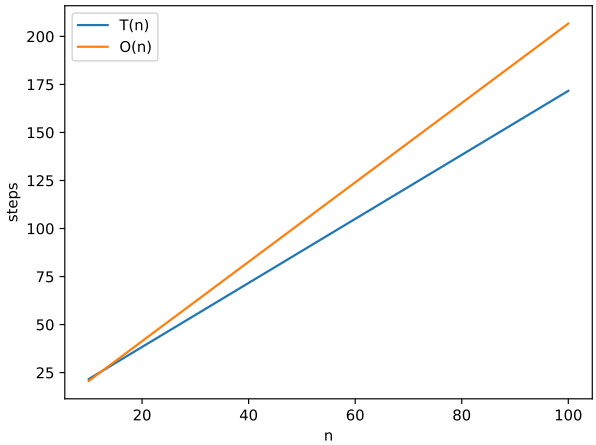

Figura 27: Coste computacional de MT2T-6B

Inclusión de una tercera cinta

El diseño se basa en el uso de una MT de dos cintas, pero la inclusión de una tercera cinta proporcionaría beneficios, por los mismos motivos que en el apartado

Se podría utilizar la tercera cinta para copiar w a dos cintas en vez de una, y así evitándonos recorrer w una de las veces para recolocar el cabezal.

El diseño de la máquina queda representado en la Figura 28.

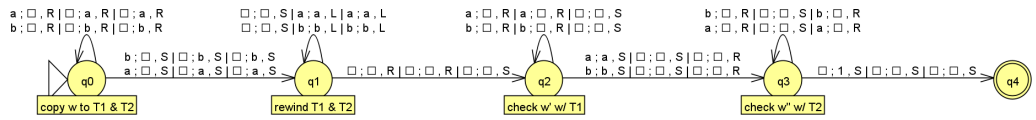

Figura 28: Implementación en JFLAP de MT3T-6B (`src/tm/MT3T-6B.jff`)

Realizamos la evaluación empírica[27]:

Entrada	n	Pasos
`aaa`	3	7
`aaaaaa`	6	11
`aaaaaaaaa`	9	15
`aaaaaaaaaaaa`	12	19

Tabla 29: Tamaño de palabra y número de pasos realizados para MT3T-6B

Calculamos el coste computacional mediante Diferencias Finitas:

n	**3**	**6**	**9**	**12**
$T(n)$	**7**	**11**	**15**	**19**
$A(n) = T(n) - T(n-2)$		4	4	4
$B(n) = A(n) - A(n-2)$			0	0

Tabla 30: Aplicación de Diferencias Finitas a MT3T-6B

Al ser constantes las diferencias finitas primeras, y nulas las segundas, podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

Para obtener los valores de a y b, usaremos valores de n y $T(n)$ obtenidos en la evaluación empírica:

$$n = 3,\ T(3) = 7 \rightarrow 3a + b = 7$$
$$n = 6,\ T(6) = 11 \rightarrow 6a + b = 11$$

Resolviendo, $a = \frac{5}{3}$ y $b = 3$, por lo que:

$$T_{3T-6B}(n) = \frac{4}{3}n + 3 \tag{31}$$

Podemos comparar ambas máquinas gráficamente (con $n_0 = 3$) en la Figura 29, observando que la de tres cintas tiene un coste menor:

[27] Los datos se pueden encontrar en `data/MT3T-6B.csv`.

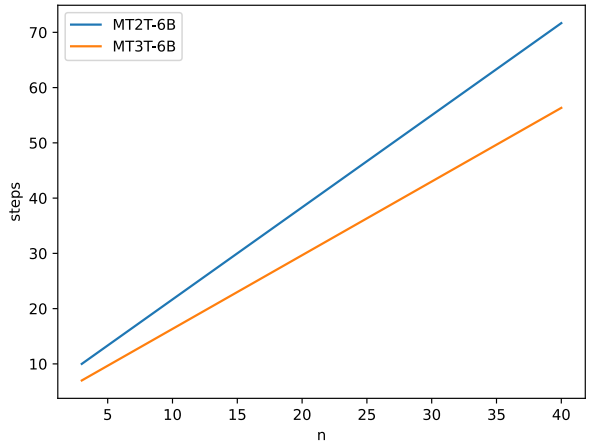

Figura 29: Comparación del coste computacional entre MT2T-6N y MT3T-6N

Comparación

Se pueden comparar todas las máquinas que resuelven este problema en la Figura 30:

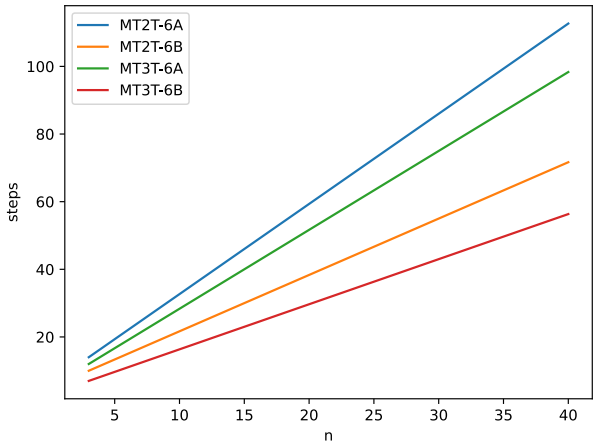

Figura 30: Comparación del coste computacional entre MT2T-6N y MT3T-6N

7. Diseño de una MTND para determinar PRIME(N)

7.A. NOPRIME(N)

Diseño propuesto

El algoritmo de resolución es el siguiente:

1. Copiar un `1` de la cinta 0 a la cinta 1, borrando la cinta 0. Este paso se realiza ya que el número 1 es primo y divisor de todos los números, para así empezar a verificar divisores a partir del número 2.
2. Copiar el posible divisor m a la cinta 1, borrando la cinta 0 y dejando el cabezal 1 al final de m. Dado que es una máquina no determinista, esto generará máquinas $\forall m \leq n$.
3. Comprobar que m es divisor de n, es decir, si $\exists p \mid m \cdot p = n$ ($p \in \mathbb{N}$). Esto se consigue recorriendo la cinta 1 (m) p veces a la vez que recorremos la cinta 0 (n) y comprobando que se llega al final de la cinta 0 a la vez que se llega al final/principio de la cinta 1.
   Hasta que las dos cintas están vacías:
   1. Avanzar la cinta 0, borrándola, y retroceder la cinta 1, mientras se lean `1`.
   2. Al llegar al principio de la cinta 1, avanzar ambas cintas, borrando la cinta 0, mientras se lean `1`.
   3. Al llegar al final de la cinta 1, volver a empezar.
4. Limpiar la cinta 1, moviéndola a la izquierda o derecha dependiendo de si hemos acabado al principio o al final de m, y **parar**.

El diseño de la máquina queda representado en la Figura 31.

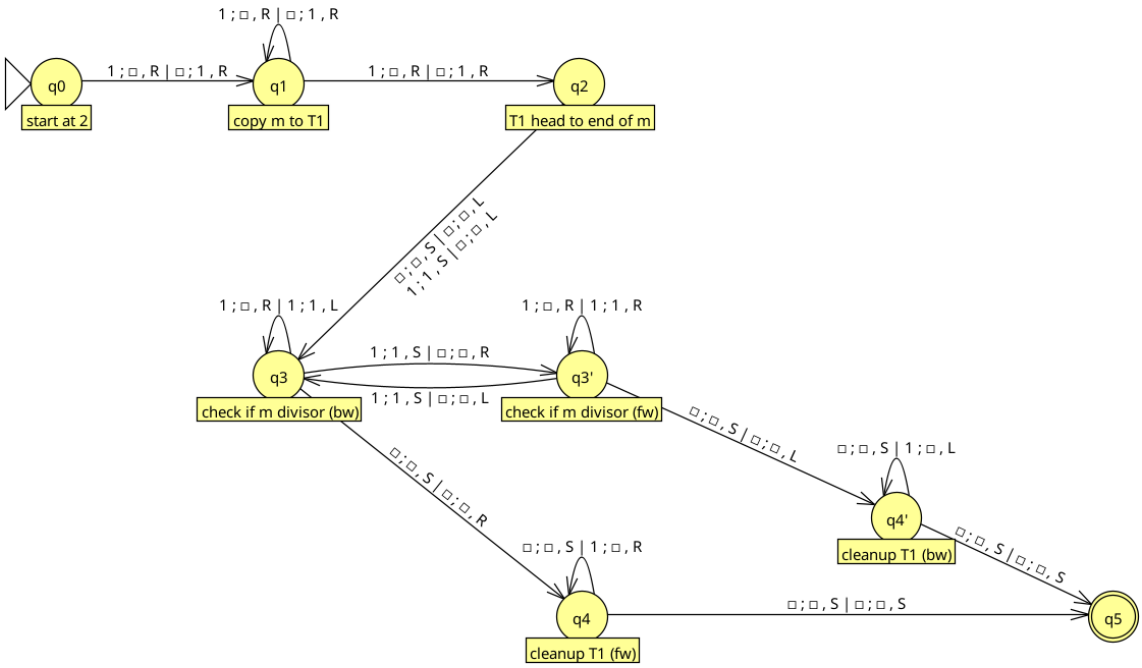

Figura 31: Implementación en JFLAP de MT-7A (`src/tm/MT-7A.jff`)

Peor caso

El peor caso de este algoritmo sería un número primo, ya que tendría que comprobar todos los números $m < n$ hasta darse cuenta que es un número primo. Sin embargo, al estar estos casos que no pertenecen al lenguaje que reconoce la MT, no consideramos que sean casos válidos.
Por lo tanto, el peor caso sería un número no primo con el mayor divisor posible, es decir, sea $q \in \mathbb{P}$, $n = q + 1$, o el número siguiente a un número primo.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como n el tamaño de la palabra, y midiendo el número de pasos realizados para resolver el problema[28]:

Entrada	n	Pasos
1111	4	9
111111	6	12
11111111	8	15
11111111	12	20
11111111	14	24

Tabla 31: Tamaño de palabra y número de pasos realizados para MT-7A

[28] Los datos se pueden encontrar en `data/MT-7A.csv`.

Coste computacional

En este caso no resulta posible aplicar el método de Diferencias Finitas[29] para obtener el coste compuacional, ya que para los peores casos, que son los escenarios a evaluar, el tamaño de palaba (n) no crece de manera lineal. A simple vista se observa que la diferencia entre el coste de $n = 6$ y $n = 8$, dos unidades, aumenta en menor proporción que para el siguiente caso, la diferencia entre $n = 8$ y $n = 12$, cuatro unidades.

Las diferencias finitas no son costantes, pero podemos aproximar $T(n)$ con un polinomio de primer orden, es decir, $T(n) = an + b$.

$$n = 4,\ T(4) = 9 \rightarrow 4a + b = 9$$
$$n = 6,\ T(6) = 12 \rightarrow 6a + b = 12$$

Resolviendo, $a = \frac{3}{2}$ y $b = 3$, por lo que:

$$O_{7B}(n) = \frac{3}{2}n + 3 \tag{33}$$

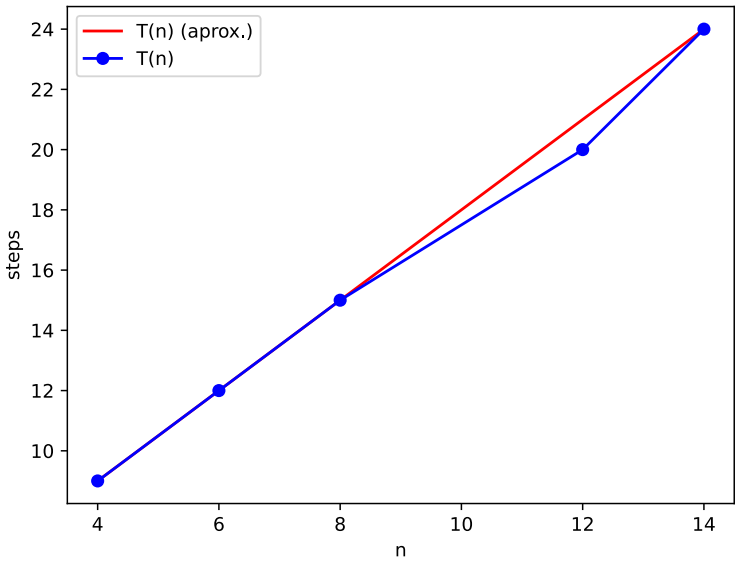

Figura 32: Coste computacional de MT-7A

7.B. PRIME(N)

Diseño propuesto

Una máquina de Turing no determinista, por definición, "acepta una entrada w si existe cualquier secuencia de movimientos que lleva desde la configuración inicial con w como

[29] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

entrada hasta una configuración con un estado de aceptación".[30]
Si vemos una máquina no determinista como una serie de procesos (máquinas de Turing) independientes que son generados cada vez que se da una transición no determinista, que *cualquiera* de esos procesos acabe en aceptación indica que el lenguaje es aceptado, mientras que que *ninguno* lo haga indica que el lenguaje no es aceptado. A la hora de crear una máquina no determinista (MTND) que determine si un número es primo o no, es sencillo hacer como en el apartado 7.A y que la máquina reconozca números no primos, ya que hallar cualquier divisor indica que n no es primo, aceptando el lenguaje.

Sin embargo, para crear una MTND que reconozca números primos, se tendría que comprobar que *ningún* número fuera divisor, es decir, que todos los 'procesos' acaben en rechazo para así aceptar el lenguaje. Esto plantea un problema, que es la intercomunicación entre procesos. Para que, digamos, un 'proceso maestro' determine si n es primo los 'subprocesos' deberían notificárselo de alguna forma.
Esto, teniendo en cuenta nuestro modelo de MTND, no es posible; no hay ningún tipo de comunicación entre procesos.

Por este motivo hemos visto imposible realizar una MTND como la que se ha pedido, así que hemos optado por hacer la versión determinista de ésta máquina, basada en la MT-7A, para posteriormente compararla con la ella.

El algoritmo de resolución es el siguiente:

- Ciclo:

  1. Copiar un `1` de la cinta 0 a la cinta 1. Esto se hace ya que el número 1 es primo y divisor de todos los números, para así empezar a verificar divisores a partir del número 2.
     a) Copiar el posible divisor $m+1$ a la cinta 1, borrando la cinta 0 y dejando el cabezal 1 al final de m.
     b) Si se ha llegado al final de la cinta 0, es $m = n$, por lo que n es primo. Limpiar ambas cintas y **parar**.
     c) Comprobar que m no es divisor de n, es decir, que $\nexists p \mid m \cdot p = n$ ($p \in \mathbb{N}$). Esto se consigue recorriendo la cinta 1 (m) p veces a la vez que recorremos la cinta 0 (n) y comprobando si se llega al final de la cinta 0 a la vez que *no* se llega al final/principio de la cinta 1.
     Hasta llegar al final de la cinta 0:
        1. Avanzar la cinta 0, y retroceder la cinta 1, mientras se lean `1`.
        2. Al llegar al principio de la cinta 1, avanzar ambas cintas, mientras se lean `1`.
        3. Al llegar al final de la cinta 1, volver a empezar.

[30] ver 6, pg. 289.

d) Rebobinar ambas cintas, y avanzarlas *m* veces.

El diseño de la máquina queda representado en la Figura 33.

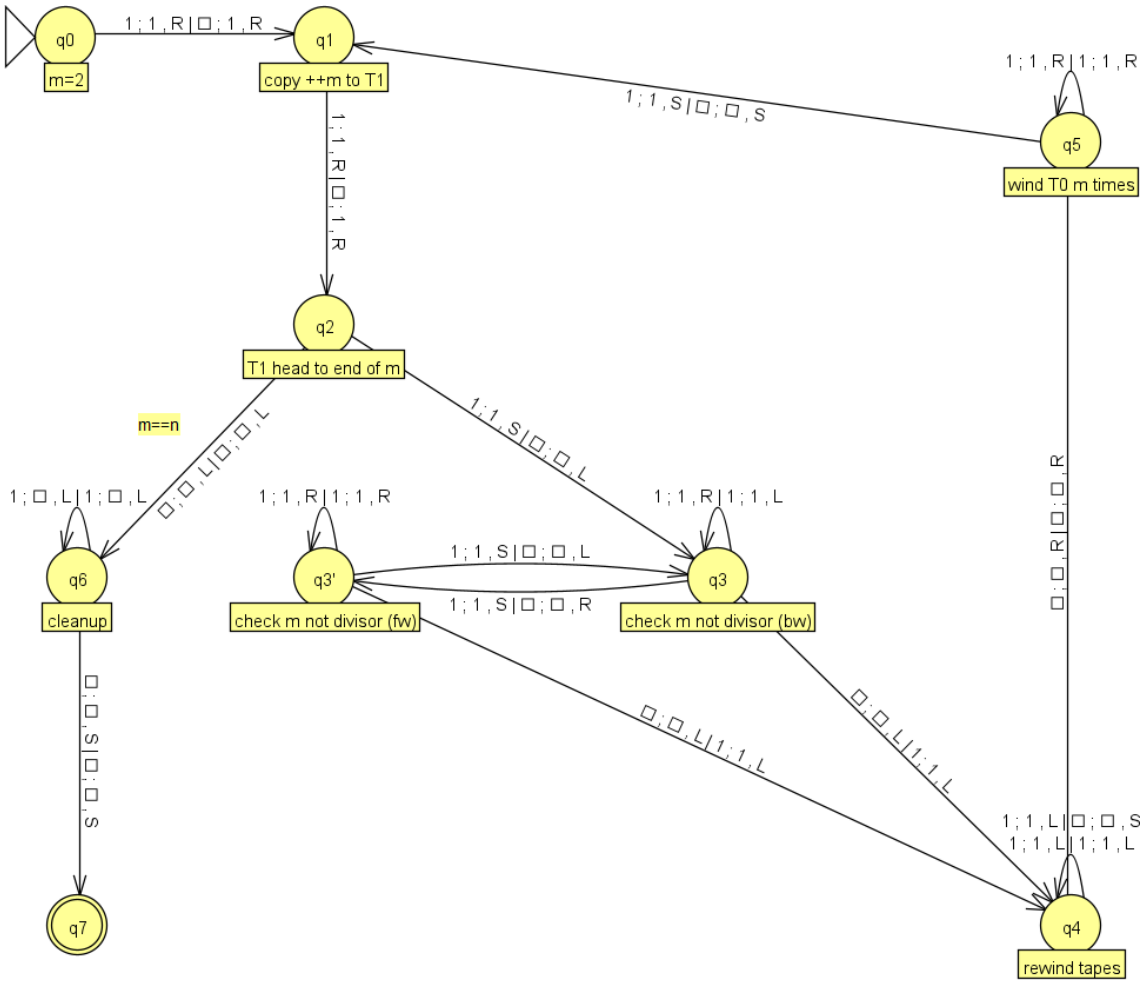

Figura 33: Implementación en JFLAP de MT-7B (`src/tm/MT-7B.jff`)

Peor caso

El peor caso del algoritmo sigue siendo un número primo, pero como en este caso sí es parte del lenguaje, sí podemos usarlo como peor caso.

Evaluación empírica

Realizamos la evaluación empírica en el peor caso, tomando como *n* el tamaño de la palabra, y midiendo el número de pasos realizados para resolver el problema[31]:

[31] Los datos se pueden encontrar en `data/MT-7B.csv`.

Entrada	n	Pasos
`11`	2	6
`111`	3	18
`11111`	5	55
`1111111`	7	109
`11111111111`	11	266

Tabla 32: Tamaño de palabra y número de pasos realizados para MT-7B

Coste computacional

En este caso no resulta posible aplicar el método de Diferencias Finitas[32] para obtener el coste compuacional, ya que para los peores casos, que son los escenarios a evaluar, el tamaño de palaba (n) no crece de manera lineal. A simple vista se observa que la diferencia entre el coste de $n = 2$ y $n = 3$, una unidad, aumenta en menor proporción que para el siguiente caso, la diferencia entre $n = 3$ y $n = 5$, dos unidades.

Dado que el algoritmo tiene un bucle principal en el que se va a recorrer n al menos una vez, se puede aproximar la expresión de la complejidad a un comportamiento cuadratico definido por un polinomio de grado dos.

$$n = 2,\ T(2) = 6 \rightarrow 8d + 4a + 2b + c = 6$$
$$n = 3,\ T(3) = 18 \rightarrow 28d + 9a + 3b + c = 18$$
$$n = 5,\ T(5) = 55 \rightarrow 125d + 25a + 5b + c = 55$$

Resolviendo, $a = \frac{13}{6}$, $b = \frac{7}{6}$ y $c = -5$, por lo que:

$$T_{7\text{B}} = \frac{13}{6}n^2 + \frac{7}{6}n - 5 \tag{35}$$

[32] ver 3, pgs. 1-42: *Chapter 1. Difference Tables and Polynomial Fits*.

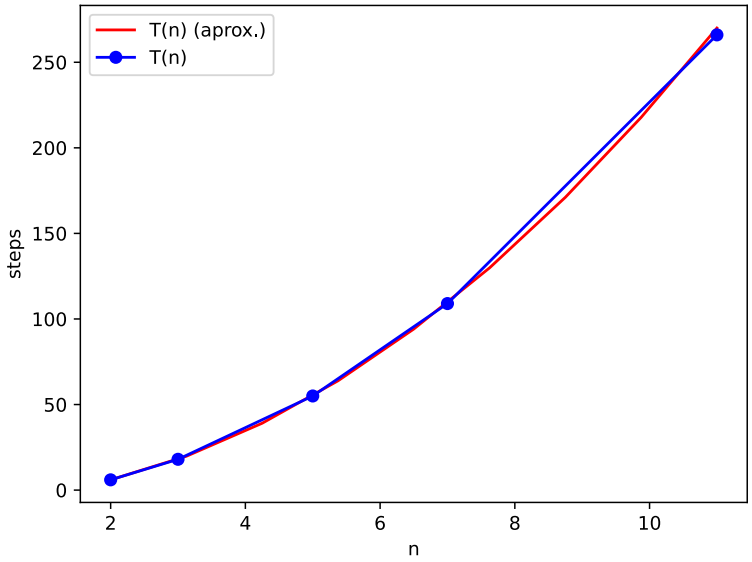

Figura 34: Coste computacional de MT-7B

Es una buena aproximación, pero observamos que no es exacta. Podríamos seguir aproximando con un polinomio de grado tres o, debido a que sólo hemos hecho cinco pruebas, de hasta grado cuatro, lo cual es cierto que aumentaría la aproximación, sin llegar nunca a ser exacta. Dejamos, para el lector curioso, la aproximación de grado cuatro:

$$T'_{7B} = -\frac{1}{3440}n^4 - \frac{271}{92880}n^3 + \frac{205637}{92880}n^2 + \frac{93631}{92880}n - \frac{44993}{9288} \tag{36}$$

7.C. Comparación de MT-7A y MT-7B

Se puede comparar ambas y comprobar que mientras que NOPRIME(N) es lineal, PRIME(N) no lo es, principalmente debido a que NOPRIME es no determinista.

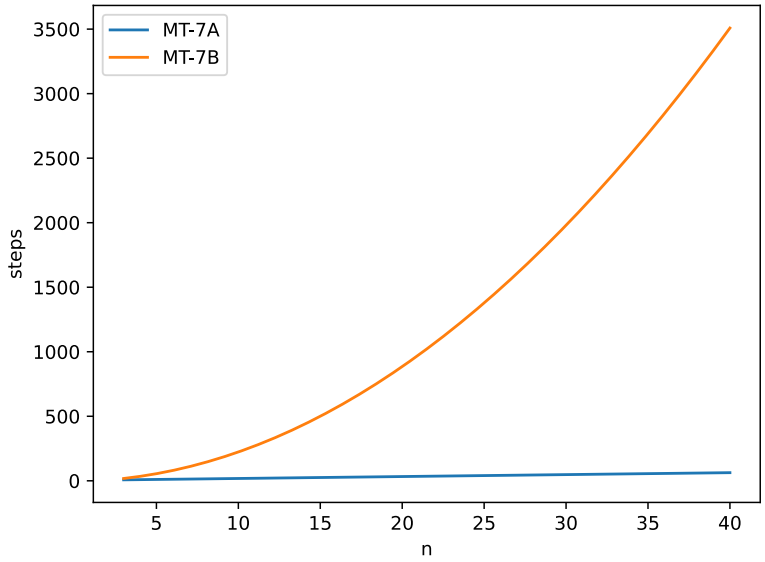

Figura 35: Coste computacional de MT-7B

Parte III

Conclusiones

8. Desafios

Al igual que en la práctica anterior, hemos aprovechado para aprender a automatizar pruebas con Python y a mostrarlas con la librería matplotlib[2]. También aprovechamos para empezar a trabajar con herramientas necesarias para realizar nuestro TFG, como son LaTeX[7] para realizar la memoria, y BibLaTeX[8] para manejar las referencias.

A la hora de automatizar las pruebas, nos decantamos por crear las máquinas de Turing en JFLAP[1], convertirlas mediante el *script*[33] proporcionado (modificándolo para nuestros propósitos) al formato del simulador turingmachinesimulator[9], y simularlas localmente a través de Turing Machine Simulator[10], un simulador open-source que lee el mismo formato que turingmachinesimulator. También se modificó el código de este simulador para que exportase los resultados en formato JSON y así poder trabajar con ellos cómodamente en Python. Posteriormente se creó una *pull request*[34] para contribuír el código.

También nos topamos con otro problema técnico: JFLAP[1] no exporta las imágenes en formato vectorial (SVG, etc.), sino en PNG y con un límite de calidad.
Encontramos entonces JFLAP2Ti*k*Z[11], el cual convierte archivos `.jff` en `.tex`, para así luego exportar a PDF y reimportar a LaTeXa través del comando `\includegraphics`, pero nos topamos con el problema de que se perderían las notas de los estados, lo cual es algo que clarifica y ayuda bastante a la comprensión, por lo tanto no acabó siendo usado.

Más allá de las dificultades técnicas, también tuvimos problemas al diseñar ciertas máquinas, como MT-2A y MT-7B, y al calcular ciertas cotas superiores mediante Diferencias Finitas.

[33] `src/jf2tm.py`
[34] https://github.com/fcortes/turing-machine-simulator/pull/2

9. Conclusiones Generales

En terminos generales consideramos que esta práctica ha resultado de gran utilidad para poder poner en práctica los conocimeinto teóricos obtenidos tanto de desarrollo de máquinas de Turing deterministas y no deterministas, permitiendonos familiarizarnos con una herramienta de gran utilidad para ello es como es JFLAP, así como la evaluación del rendimeinto de algoritmos en terminos de complejidad y coste computacional. Estos conocimientos pueden resultar de gran ayuda de cara a futuros proyectos en los que se lleve acabo el desarrollo propio de algoritmos, a fin de poder optimizarlos y analizar posibles mejoras en su implementación.

También queremos dejar una última figura, los costes computacionales de todas las máquinas en una pequeña ventana donde se puedan apreciar todas:

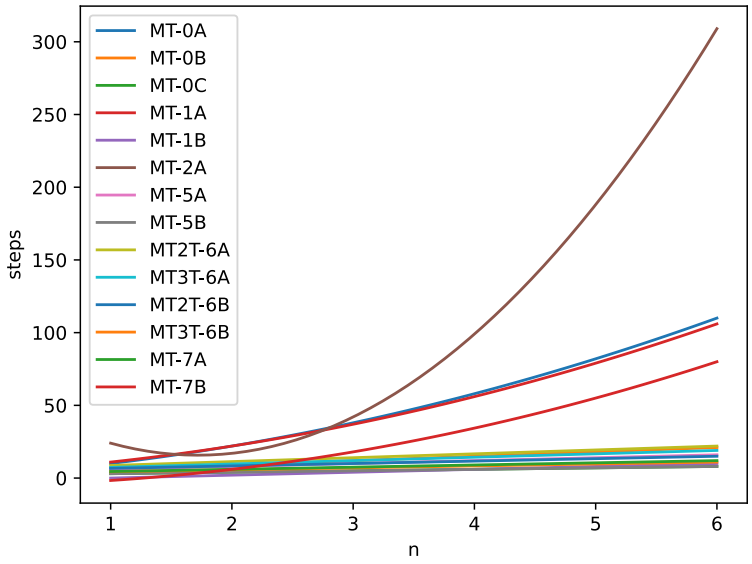

Figura 36: Coste computacional de todas las MT

Parte IV

Bibliografía

Referencias

[1] «JFLAP.» (2018), dirección: https://www.jflap.org/ (visitado 03-03-2024).

[2] «Matplotlib – Visualization with Python.» (2023), dirección: https://matplotlib.org/ (visitado 04-04-2024).

[3] A. Cuoco, *Mathematical Connections: A Companion for Teachers and Others* (Classroom Resource Materials), eng. Mathematical Association of America, 2005.

[4] M. Sipser, *Introduction to the theory of computation*, eng, 3rd. ed. Australia: Cengage Learning, 2013.

[5] A. Aho y J. Ullman, *Foundations of Computer Science: C Edition* (Principles of computer science series). United Kingdom: W. H. Freeman, 1994. dirección: http://infolab.stanford.edu/~ullman/focs.html.

[6] J. E. Hopcroft, R. Motwani y J. D. Ullman, *Introducción a la teoría de autómatas, lenguajes y computación*, spa, 3ª ed. Madrid: Pearson Addison Wesley, 2008.

[7] «LaTeX – A document preparation system.» (2024), dirección: https://www.latex-project.org/ (visitado 04-04-2024).

[8] «plk/biblatex.» (2024), dirección: https://github.com/plk/biblatex (visitado 04-04-2024).

[9] M. Ugarte. «turingmachinesimulator.» (2017), dirección: https://turingmachinesimulator.com/ (visitado 03-03-2024).

[10] F. Cortés. «fcortes/turing-machine-simulator.» (2017), dirección: https://github.com/fcortes/turing-machine-simulator (visitado 05-03-2024).

[11] A. Mertz y W. Slough. «JFLAP2TikZ – Convert JFlap file into a LaTeX file depicting the automaton using TikZ.» (2015), dirección: https://ctan.org/tex-archive/graphics/jflap2tikz (visitado 05-03-2024).

Parte V

Apéndices

A. Instalación y ejecución

Ejecutar en JFLAP[1]

1. Comprueba que tienes Java 8 o superior instalado, ejecutando `java -version`. Si no lo tienes, descarga e instala Java Runtime Enviroment con JDK 8+, tanto el Java SE[35] como OpenJDK[36] .

2. Descarga e instala JFLAP 7.1[37] :

   ```
   wget https://www.jflap.org/jflaptmp/july27-18/JFLAP7.1.jar
   ```

3. Ejecuta JFLAP con:

   ```
   java -jar JFLAP7.1.jar
   ```

   Puedes arreglar errores de escalado con el parámetro `-Dsun.java2d.uiScale`, e.g. `java -Dsun.java2d.uiScale=2.0 -jar JFLAP7.1.jar`

4. Selecciona `File` → `Open...` y el archivo deseado (archivo `.jff` dentro de la carpeta `src/tm/`) para abrirlo. Ábrelo como una `Standard Turing Machine`.

5. Usa `Input` → `Step...` para ejecutar paso a paso o `Input` → `Fast Run...` para ejecutar completo.

Ejecución de los tests en Python

Requiere Python[38] 3.10+.

1. Crea un *virtual enviroment* en la carpeta `.venv/`:

   ```
   python3 -m venv ./.venv
   ```

[35] https://www.oracle.com/java/technologies/java-se-glance.html
[36] https://openjdk.org/
[37] https://www.jflap.org/
[38] https://www.python.org/

2. Activa el entorno:

   - Linux:

     ```
     source .venv/bin/activate
     ```

   - Windows (PowerShell):

     ```
     & .\.venv\Scripts\Activate.ps1
     ```

3. Instala las dependencias:

   ```
   pip install -r requirements.txt
   ```

4. Compila el Turing Machine Simulator[10] con GNU/Make[39] :

   ```
   cd turing-machine-simulator
   make
   cd ..
   ```

   Si estás en Windows, te recomendamos instalar WSL2[40] y ejecutar 'en Linux', o con GCC a través de MinGW-W64[41] (puedes encontrar binarios ya compilados en https://github.com/niXman/mingw-builds-binaries), compilando mediante el comando `gcc` que puedes encontrar dentro de `turing-machine-simulator/Makefile`.

5. Ejecuta el *script* con:

   ```
   python3 src/test.py
   ```

[39] https://www.gnu.org/software/make/
[40] https://learn.microsoft.com/es-es/windows/wsl/install
[41] https://www.mingw-w64.org/