TEORÍA AVANZADA DE LA COMPUTACIÓN

Grado en Ingeniería Informática y Administración de Empresas

MEMORIA

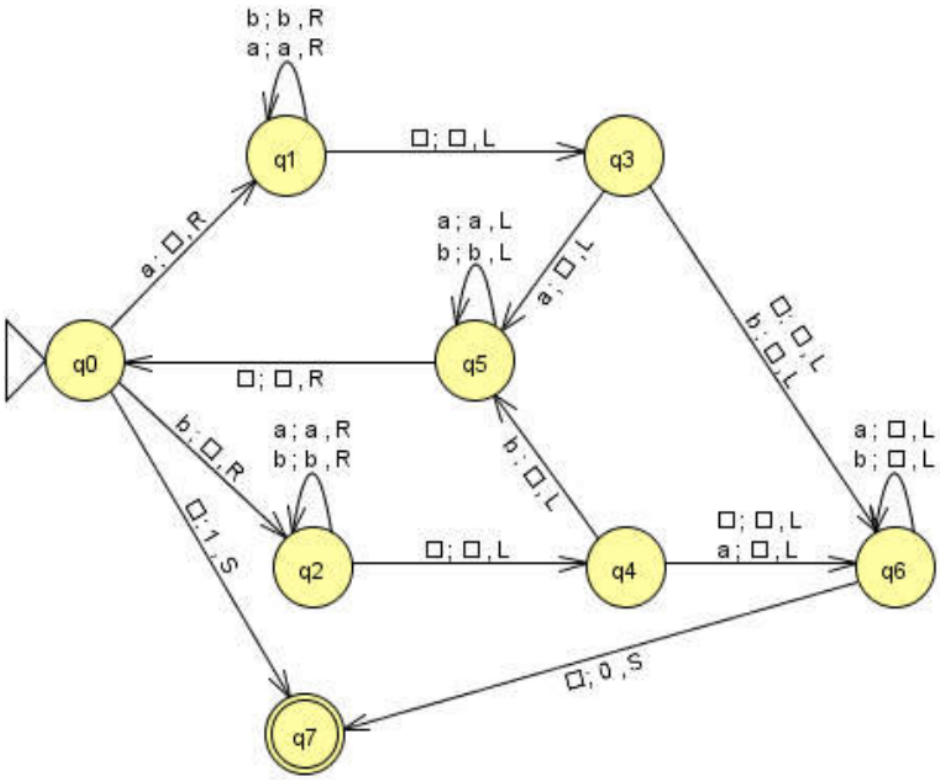
Práctica 2 - Máquinas de Turing

Jaime Martínez de Miguel. 100386281
Ricardo Prieto Álvarez. 100386267
Miriam Valdizán Arce. 100386356

ÍNDICE

1. Palíndromos I:

- **Máquina Determinista con 1 Cinta**
  - ❖ **Diseño propuesto:**



Funcionamiento:

Ciclo:

Borra una letra en el extremo izquierdo.
Busca la correspondiente a la derecha.
La borra.
Vuelve atrás a la primera letra por la izquierda.

Fin1:

Cinta en blanco Deja 1 en la cinta y Para (Acepta).

Fin2:

Falla una correspondencia Deja 1 en la cinta y Para (“Falla”).

Fin3:

Incluye caso de número impar de letras se trata en Fin2.

Detalles:

- Palabra de tamaño n=2k.
- La MT hace k recorridos si n es el número de símbolos inicial.
- Recorridos sucesivos se acortan en 2 símbolos.
- En caso de fallar la correspondencia ⇨ hay que borrar la cinta (q6) con un recorrido lineal.

❖ Peor caso en cuanto a coste computacional:

Cuando es palíndromo para tamaño par.

❖ Estudio empírico:

Tamaño n	0	2	4	6	8	10	12
Pasos	1	6	15	28	45	66	91

❖ Estudio analítico. Diferencias finitas y T(n):

Tamaño n	0	2	4	6	8	10	12
Pasos	1	6	15	28	45	66	91
Dif. 1		5	9	13	17	21	25
Dif. 2			4	4	4	4	4
Dif. 3				0	0	0	0

Al ser constantes las Diferencias Finitas segundas (y nulas las terceras) ⇨ Podemos aproximar T(N) con un polinomio de segundo orden.

$T(n) = an^2 + bn + c$

$T(0) = 1$

$T(2) = 6$

$T(4) = 15$

$$\begin{cases} T(0) = a \cdot 0 + b \cdot 0 + c = 1 \\ T(2) = 4a + 2b + c = 6 \\ T(4) = 16a + 4b + c = 15 \end{cases}$$

$a = ½$
$b = 3/2$
$c = 1$

$$T(n) = \frac{1}{2}n^2 + \frac{3}{2}n + 1$$

❖ Cota superior asintótica para $n_0 = 10$:

$T(n) = \frac{1}{2}n^2 + \frac{3}{2}n + 1$

$S(n) \mid S(n) \geq T(n)\ para\ n > n_0 = 10$

$S(n) = kn^2$

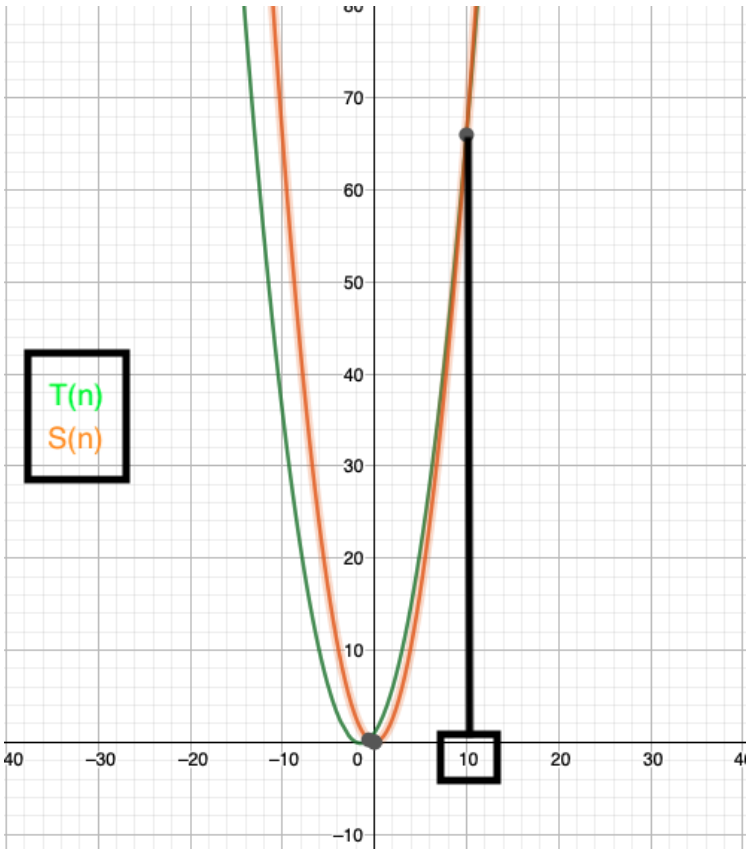
Aseguramos $S(n) = kn^2 \geq T(n) = \frac{1}{2}n^2 + \frac{3}{2}n + 1$

$kn^2 \geq \frac{1}{2}n^2 + \frac{3}{2}n + 1$

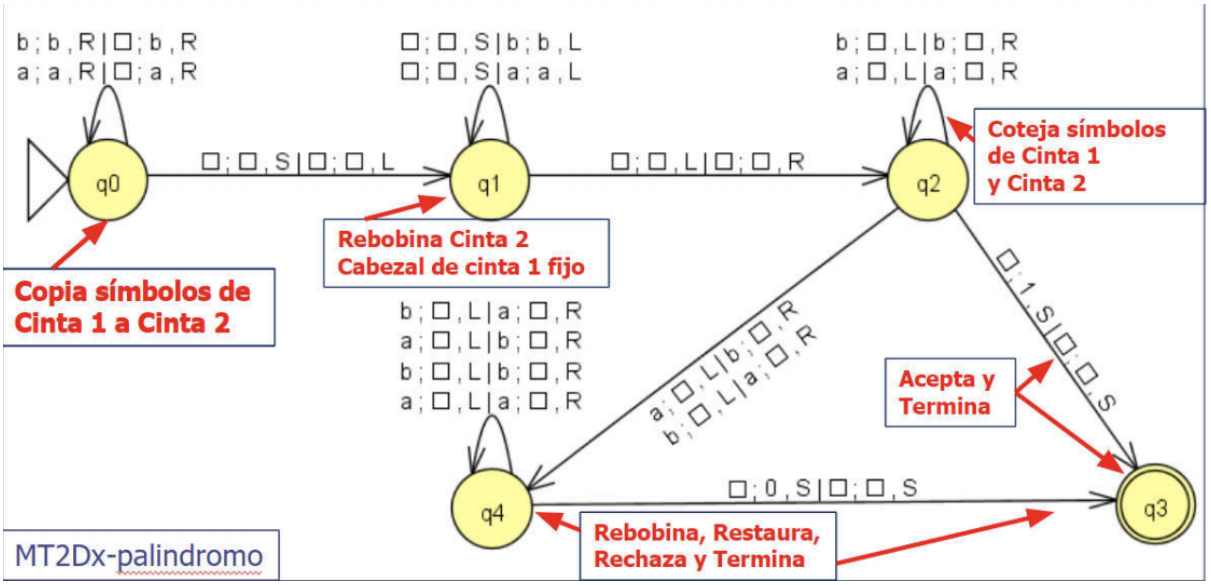
$k \geq \frac{1}{2} + \frac{3}{2n} + \frac{1}{n^2}$

Sustituimos la n por 10:

k > 0,66



- **Máquina Determinista con 2 Cintas**
  - ❖ **Diseño propuesto:**



  - ❖ **Peor caso en cuanto a coste computacional:**

Cuando es palíndromo para tamaño par.

  - ❖ **Estudio empírico:**

Tamaño n	2	4	6	8	10
Pasos	9	15	21	27	33

  - ❖ **Estudio analítico. Diferencias finitas y T(n):**

Tamaño n	2	4	6	8	10
Pasos	9	15	21	27	33
Dif. 1		6	6	6	6
Dif. 2			0	0	0

Al ser constantes las Diferencias Finitas primeras (y nulas las segundas) es posible aproximar T(n) con un polinomio de primer grado.

$T(n) = an + b$

$T(2) = 9$

$T(4) = 15$

$$\begin{cases} T(2) = 2a + b = 9 \\ T(4) = 4a + b = 15 \end{cases}$$

a = 3
b = 3

$$T(n) = 3n + 3$$

❖ Cota superior asintótica para $n_0 = 10$:

$T(n) = 3n + 3$

$S(n) \mid S(n) \geq T(n)\ para\ n > n_0 = 10$

$S(n) = kn$

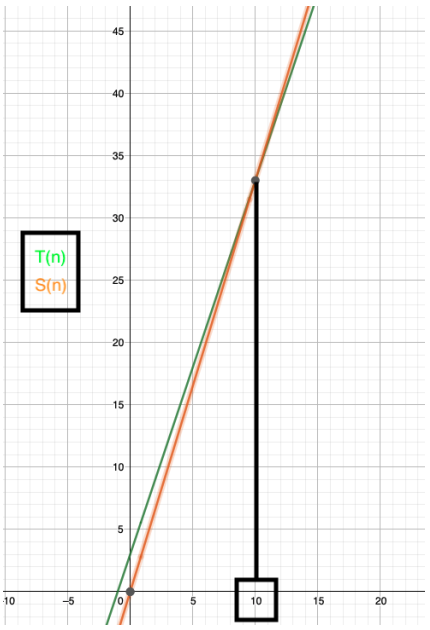
Aseguramos $S(n) = kn \geq T(n) = 3n + 3$

$kn \geq 3n + 3$

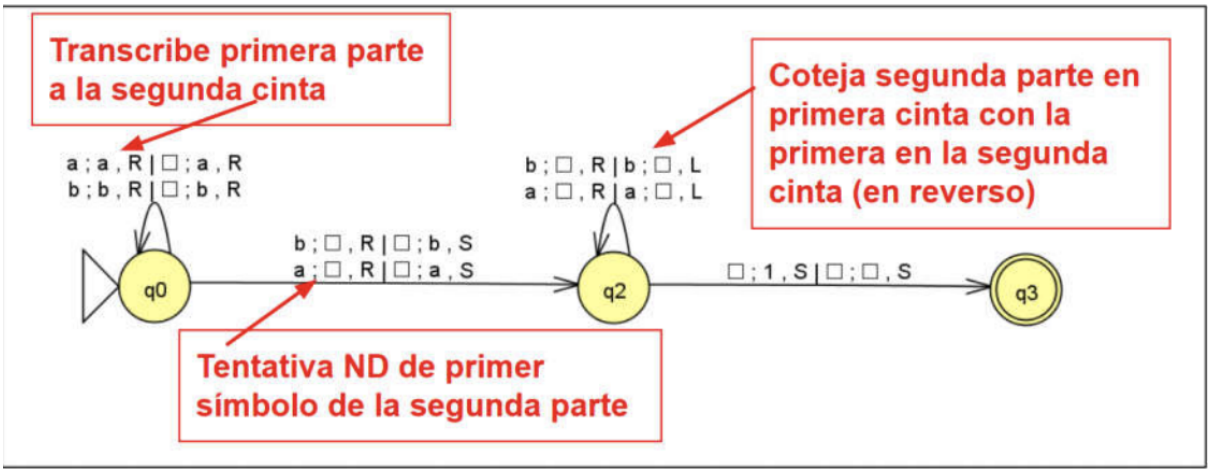
$k \geq 3 + \frac{3}{n}$

Sustituimos la n por 10:

k > 3,3



- **Máquina No Determinista con 2 Cintas**
  - ❖ **Diseño propuesto:**



Se basa en:

- Transcribir la primera mitad de la palabra de la primera cinta a la segunda.
- A partir de la mitad de la palabra se coteja el resto en la primera cinta con lo transcrito en la segunda (en reverso).
- Problema. A priori la MT NO SABE dónde está la mitad.
- Solución: va bifurcando de forma no determinista en cada símbolo en la transición q0 → q1 suponiendo que el último símbolo procesado es el último de la primera mitad.
- Si w es palíndromo alguno de ellos será el último de la primera mitad.
- Si alguna de las instancias de la MT llega al estado final ⇒ palabra w es palíndromo.

● En este caso, no se incluye la opción de dejar un 0 en la cinta en caso de que no sea palíndromo, porque sólo debe sobrevivir la instancia que reconozca el palíndromo.

❖ Peor caso en cuanto a coste computacional:

Cuando es palíndromo para tamaño par.

❖ Estudio empírico:

Tamaño n	2	4	6	8	10
Pasos	3	5	7	9	11

❖ Estudio analítico. Diferencias finitas y T(n):

Tamaño n	2	4	6	8	10
Pasos	3	5	7	9	11
Dif. 1		2	2	2	2
Dif. 2			0	0	0

Al ser constantes las Diferencias Finitas primeras (y nulas las segundas) es posible aproximar T(n) con un polinomio de primer grado.

$T(n) = an + b$

$T(2) = 3$

$T(4) = 5$

$T(6) = 7$

$$\begin{cases} T(2) = 2a + b = 3 \\ T(4) = 4a + b = 5 \end{cases}$$

a = 1
b = 1

$$T(n) = n + 1$$

❖ Cota superior asintótica para $n_0 = 10$:

$T(n) = n + 1$

$S(n) \mid S(n) \geq T(n)\ para\ n > n_0 = 10$

$S(n) = kn$

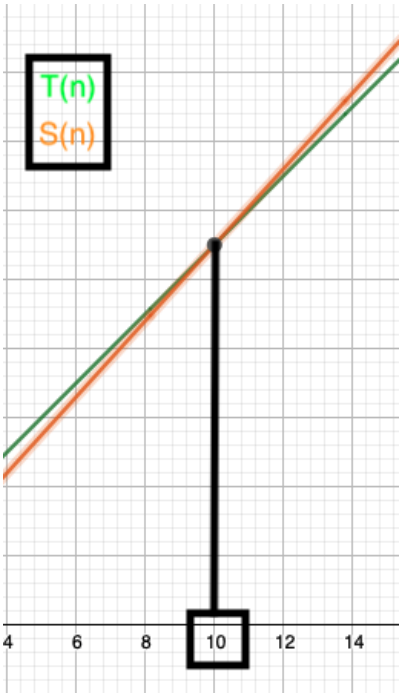
Aseguramos $S(n) = kn \geq T(n) = n + 1$

$kn \geq n + 1$

$k \geq 1 + \frac{1}{n}$

Sustituimos la n por 10:

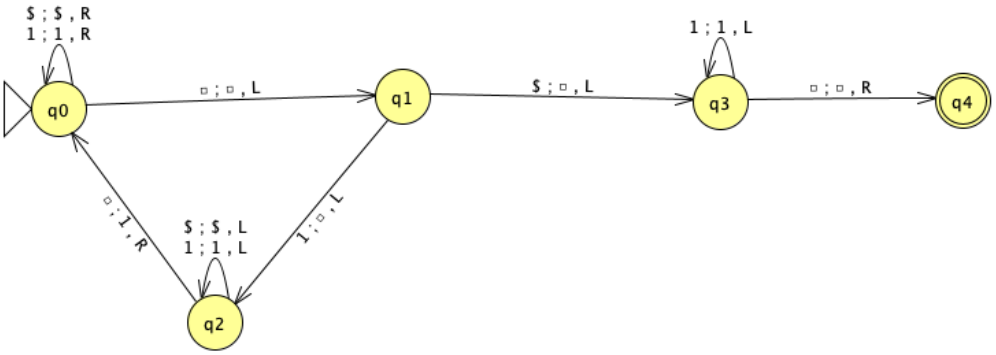
k > 1,1



2. Suma enteros en base 1

● Máquina Determinista con 1 Cinta

❖ Diseño propuesto:



Funcionamiento:
Ciclo (estados q_0 , q_1 y q_2):

1. Recorre la cinta de izquierda a derecha hasta llegar a un 1 en el extremo derecho.
2. Borra este 1.
3. Recorre toda la cinta de derecha a izquierda.
4. Añade el 1 borrado en la derecha a la izquierda de la cinta.

Una vez terminado de sumar (finalizado el ciclo), transiciona borrando el "$" de q_1 al estado q_3 , donde se recorre la cinta desde el "$" borrado hasta la izquierda.

Tras esto, posiciona el cabezal de la máquina sobre el 1 más a la izquierda de la cinta, esta es la transición de q_3 a q_4 , el estado final.

❖ Peor caso en cuanto a coste computacional:

El peor caso de esta Máquina de Turing sería cuanto mayor sea el número de la derecha, debido a que el funcionamiento de la máquina consiste en restar 1 de la derecha y sumarlo en la izquierda.

Por ejemplo, 1111$1 solo sumaría una vez, mientras que 1$1111 tendría que sumar cuatro veces, lo que implicaría recorrer la cinta cuatro veces hacia delante y detrás.

❖ Estudio empírico:

Al realizar este estudio empírico hemos probado siempre con el peor caso posible para cada tamaño. Como ya hemos explicado anteriormente este caso sería 1$111… (tantos 1's como sean necesarios para rellenar el tamaño probado).

Tamaño n	2	4	6	8	10
Pasos	6	28	66	120	190

❖ Estudio analítico. Diferencias finitas y T(n):

Tamaño n	2	4	6	8	10
Pasos	6	28	66	120	190
Dif. 1		22	38	54	70
Dif. 2			16	16	16
Dif. 3				0	0

Al ser constantes las Diferencias Finitas segundas (y nulas las terceras) es posible aproximar T(n) con un polinomio de segundo grado.

$$T(n) = an^2 + bn + c$$

$$T(2) = 6$$

$$T(4) = 28$$

$$T(6) = 66$$

$$\begin{cases} T(2) = 4a + 2b + c = 6 \\ T(4) = 16a + 4b + c = 28 \\ T(6) = 36a + 6b + c = 66 \end{cases}$$

a = 2
b = -1
c = 0

$$T(n) = 2n^2 - n$$

❖ Cota superior asintótica para $n_0 = 10$:

$T(n) = 2n^2 - n$

$S(n) \mid S(n) \geq T(n)\ para\ n > n_0 = 10$

$S(n) = kn^2$

Aseguramos $S(n) = kn^2 \geq T(n) = 2n^2 - n$

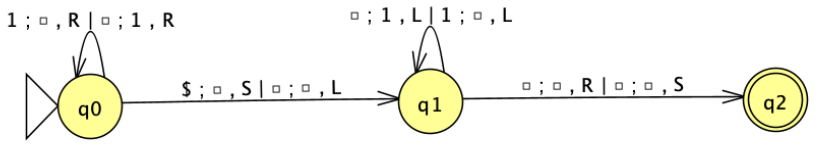
$kn^2 \geq 2n^2 - n$

$k \geq 2 - \frac{1}{n}$

Sustituimos la n por 10:

k > 1,9

- # Máquina Determinista con 2 Cintas
  - ## ❖ Diseño propuesto:



Funcionamiento:
Se copia el número de la izquierda en la 2ª cinta (borrándose de la 1ª), esto se corresponde con el estado q_0.
Se elimina el "$" de la 1ª cinta, dejando el cabezal quieto (esta es la transición de q_0 a q_1).
Se suman tantos 1's en la 1ª cinta como haya en la 2ª, eliminándose de esta (esto es q_1).
Por último se coloca el cabezal en el 1 más a la izquierda de la 1ª cinta (la última transición de q_1 a q_2).

❖ Peor caso en cuanto a coste computacional:

El peor caso de esta Máquina de Turing, al contrario que en la anterior, sería cuanto mayor sea el número de la izquierda, debido a que el funcionamiento de la máquina consiste en copiar el número de la izquierda (de la 1ª cinta) en la 2ª cinta (borrándose de la 1ª) y sumarlo al número de la derecha (de la 1ª cinta) borrándolo de la 2ª.

Por ejemplo, 1$1111 solo sumaría una vez, mientras que 1111$1 tendría que sumar cuatro veces.

❖ Estudio empírico:

Tamaño n	2	4	6	8	10
Pasos	2	6	10	14	18

❖ Estudio analítico. Diferencias finitas y T(n):

Tamaño n	2	4	6	8	10
Pasos	2	6	10	14	18
Dif. 1		4	4	4	4
Dif. 2			0	0	0

Al ser constantes las Diferencias Finitas primeras (y nulas las segundas) es posible aproximar T(n) con un polinomio de primer grado.

$T(n) = a \times n + b$

$T(2) = 2$

$T(4) = 6$

$$\begin{cases} T(2) = 2a + b = 2 \\ T(4) = 4a + b = 6 \end{cases}$$

$$\boxed{\begin{array}{l} a = 2 \\ b = -2 \end{array}}$$

$$T(n) = 2n - 2$$

❖ Cota superior asintótica para $n_0 = 10$:

$T(n) = 2n - 2$

$S(n) \mid S(n) \geq T(n) \; para \; n > n_0 = 10$

$S(n) = kn$

Aseguramos $S(n) = kn \geq T(n) = 2n - 2$

$kn \geq 2n - 2$

$k \geq 2 - \frac{2}{n}$

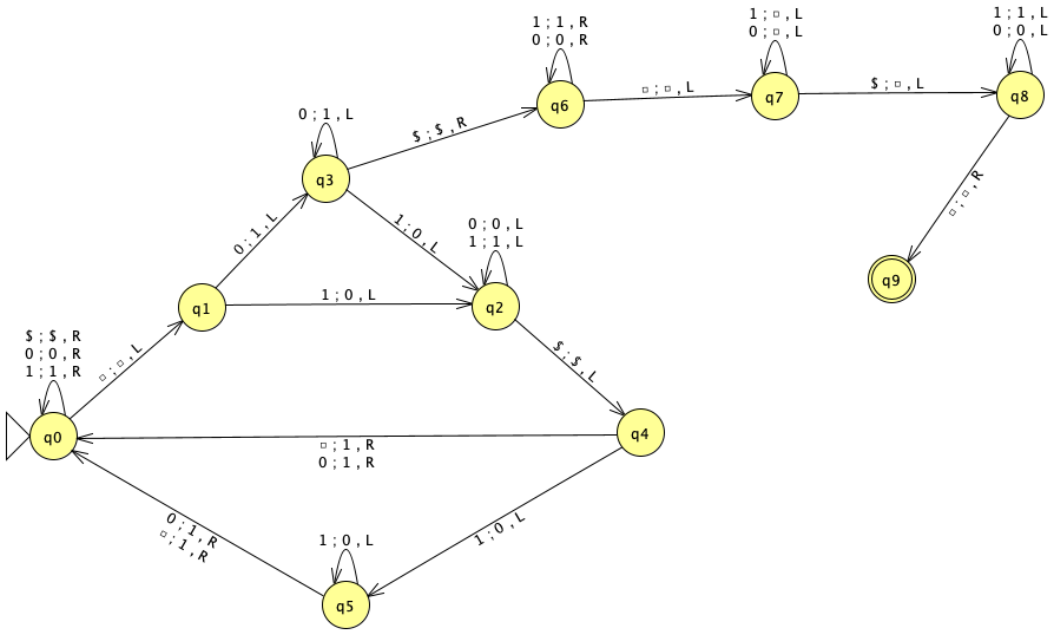
Sustituimos la n por 10:

$$\boxed{\mathbf{k > 1{,}8}}$$

3. Suma enteros en base 2

❖ Máquina Determinista con 1 Cinta

❖ Diseño propuesto:



Funcionamiento:

Ciclo:

Se recorre la cinta de izquierda a derecha (estado q_0).

Se resta 1 al número de la derecha, ya sea sin acarreo (transición de q_1 a q_2) o con él (ejecución q_1 - q_3 - q_2).

Se recorre la cinta de izquierda a derecha hasta llegar al "$" (estado q_2).

Se suma 1 al número de la izquierda, ya sea sin acarreo (transición de q_4 a q_0) o con él (ejecución q_4 - q_5 - q_0).

Tras detectar en q_3 que el número de la derecha son todo 0's (lo que implica que ya no queda nada por sumar a la izquierda) se transiciona a los estados q_6 - q_7 - q_8, donde se elimina tanto los 0's que han quedado a la derecha del "$" como este último símbolo.

Tras esto se posiciona el cabezal en el primer dígito a la izquierda de la cinta, lo que corresponde con la transición al estado final, dejando en la cinta únicamente el resultado de la suma.

❖ Peor caso en cuanto a coste computacional:

En este caso, el peor caso en cuanto a coste computacional es cuando la entrada es del tipo 1$111… debido a que la parte derecha representaría el máximo número de restas posibles. Como el algoritmo se basa en ir a la derecha, restar 1 del término de la derecha, ir a la izquierda y sumarlo en el término de la izquierda, el peor caso sería cuantas más restas hubiera que hacer.

Por ejemplo, si la entrada fuese 1111111$1, solamente se sumaría una vez, pero si fuese 1$1111111, habría que sumar 127 veces (y por ello restar 127 veces también).

❖ Estudio empírico:

Tamaño n	2	3	4	5	6	7	8	9	10	11	12
Pasos	17	21	48	107	238	529	1172	2583	5658	12317	26656

❖ Estudio analítico. T(n):

Tamaño n	2	3	4	5	6	7	8	9	10	11	12
Pasos	17	21	48	107	238	529	1172	2583	5658	12317	26656
Dif. 1		4	27	59	131	291	643	1411	3075	6659	14339
Dif. 2			23	32	72	160	352	768	1664	3584	7680
Dif. 3				9	40	88	192	416	896	1920	4096
Dif. 4					31	48	104	224	480	1024	2176
Dif. 5						17	56	120	256	544	1152
Dif. 6							39	64	136	288	608
Dif. 7								25	72	152	320
Dif. 8									47	80	168

Tamaño n	2	3	4	5	6	7	8	9	10	11	12
Div. 1		1,23529412	2,28571429	2,22916667	2,22429907	2,22268908	2,21550095	2,20392491	2,19047619	2,17691764	2,16416335
Div. 2			6,75	2,18518519	2,22033898	2,22137405	2,20962199	2,19440124	2,17930546	2,16552846	2,15332633
Div. 3				1,39130435	2,25	2,22222222	2,2	2,18181818	2,16666667	2,15384615	2,14285714
Div. 4					4,44444444	2,2	2,18181818	2,16666667	2,15384615	2,14285714	2,13333333
Div. 5						1,5483871	2,16666667	2,15384615	2,14285714	2,13333333	2,125
Div. 6							3,29411765	2,14285714	2,13333333	2,125	2,11764706
Div. 7								1,64102564	2,125	2,11764706	2,11111111
Div. 8									2,88	2,11111111	2,10526316
Div. 9										1,70212766	2,1

Como en este caso la complejidad de la máquina ya no es polinómica, dado que las diferencias finitas no eran nulas, hemos desarrollado el estudio realizando divisiones entre los diferentes términos de estas diferencias, por lo que hemos llegado a la conclusión de que es una complejidad exponencial.

En este caso en concreto, hemos determinado, realizando aparte un pequeño estudio empírico (en el que hemos podido observar que si se multiplica por 2 la parte exponencial, el resultado de la complejidad es más ajustado a los pasos que realmente se han dado), que las divisiones tienen una tendencia a 2.1, debido a que a medida que aumentamos los niveles donde se realizan estas divisiones, cada vez hay una convergencia más cercana a 2.1, por lo que la complejidad se podría aproximar a:

$$T(n) = 2 \times 2,1^{n} + C$$

donde la C es una constante que varía según el tamaño del problema pero que no podemos calcularla de forma exacta debido a la complejidad de éste.

❖ Cota superior asintótica para $n_0 = 10$:

$T(n) = 2 \times 2,1^{n} + C$

$S(n) \mid S(n) \geq T(n)\ para\ n > n_0 = 10$

$S(n) = k \times 2,1^{n}$

Aseguramos $S(n) \geq T(n) = 2 \times 2,1^{n} + C$

$k \times 2,1^{n} \geq 2 \times 2,1^{n} + C$

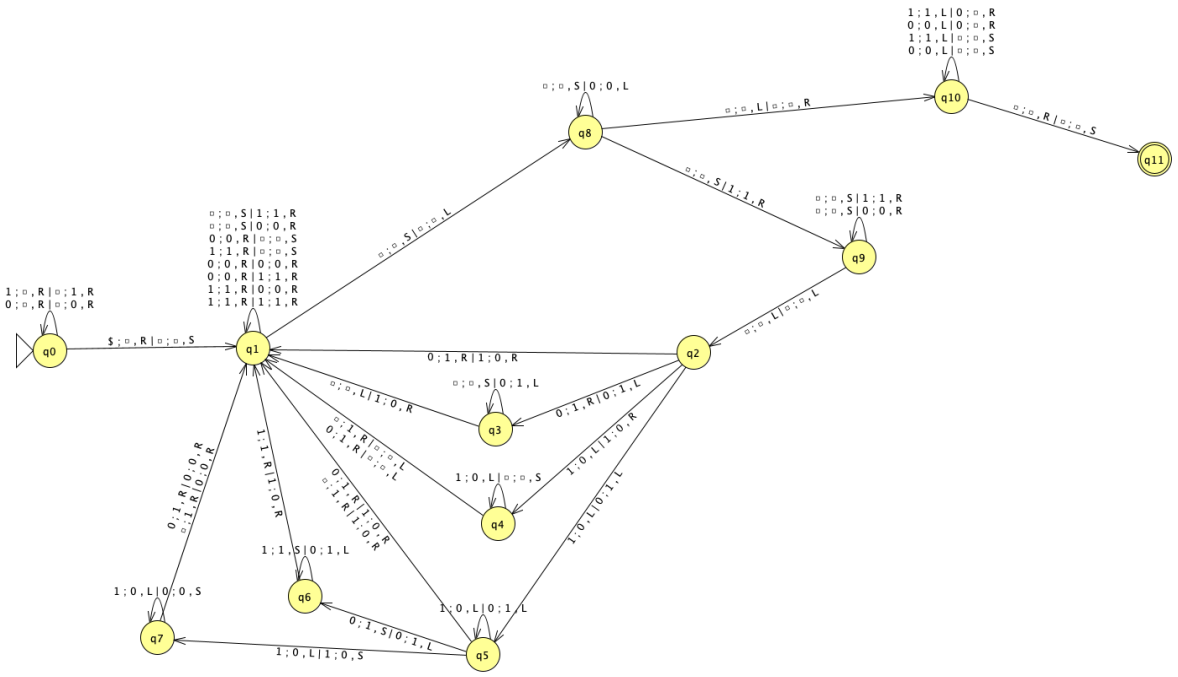
$k \geq 2 + \frac{C}{2,1^{n}}$

Sustituimos n por 10:

$$\mathbf{k > 2 + C/2.1^{10}}$$

Si considerásemos la constante C como irrelevante (es decir, igual a 0), habría una cota superior asintótica aproximada si $k > 2$.

- **Máquina Determinista con 2 Cintas**
  - ❖ **Diseño propuesto:**



Funcionamiento:
Se copia el número de la izquierda en la 2ª cinta (borrándose de la 1ª) así como el “$”. Después hay una comprobación de si el número de la 2ª cinta es todo 0’s (lo que implicaría haber acabado con la suma), esto se realiza en los estados q_8 - q_9 , donde también se vuelve a dejar el cabezal de la 2ª cinta a la derecha, tras esta comprobación.

Acto seguido hay 4 opciones que representan las combinaciones de suma y resta en binario:

1. Suma y resta sin acarreo → Transición de q_2 a q_1.
2. Suma sin acarreo y resta con él → Ejecución q_2 - q_3 - q_1.
3. Suma con acarreo y resta sin él → Ejecución q_2 - q_4 - q_1.
4. Suma y resta con acarreo → Ejecución q_2 - q_5 - q_6 - q_1 ó q_2 - q_5 - q_7 - q_1 , según donde se acabe antes el acarreo. Si se acaba primero en la suma se transiciona a q_6 , si en cambio se acaba antes en la resta, se transiciona a q_7.

Tras realizar la suma, se llegará a un punto donde el número de la 2ª cinta sea 0, entonces se transiciona de q_8 a los estados q_{10} y q_{11} , encargados de borrar la 2ª cinta y dejar el cabezal de la 1ª a la izquierda de la suma realizada.

❖ Peor caso en cuanto a coste computacional:

En este caso la peor entrada sería semejante a la de la anterior máquina, pero en este caso es cuantos más 1's hay en la parte izquierda, ya que es este término el que se copia en la 2ª cinta y se resta sobre él. Un ejemplo de esta entrada sería 1111111$1, donde habría que realizar 127 restas (y por ello también sumas).

❖ Estudio empírico:

Tamaño n	2	3	4	5	6	7	8	9	10	11	12
Pasos	6	15	34	73	152	361	630	1269	2548	5107	10226

❖ Estudio analítico. Diferencias finitas y T(n):

Tamaño n	2	3	4	5	6	7	8	9	10	11	12
Pasos	6	15	34	73	152	361	630	1269	2548	5107	10226
Dif. 1		9	19	39	79	209	269	639	1279	2559	5119
Dif. 2			10	20	40	130	60	370	640	1280	2560
Dif. 3				10	20	90	-70	310	270	640	1280

Tamaño n	2	3	4	5	6	7	8	9	10	11	12
Div. 1		2,5	2,26666667	2,14705882	2,08219178	2,375	1,74515235	2,01428571	2,00788022	2,00431711	2,00234972
Div. 2			2,11111111	2,05263158	2,02564103	2,64556962	1,28708134	2,37546468	2,00156495	2,00078186	2,00039078
Div. 3				2	2	3,25	0,46153846	6,16666667	1,72972973	2	2
Div. 4					2	4,5	-0,7777778	-4,4285714	0,87096774	2,37037037	2

Como en este caso la complejidad de la máquina ya no es polinómica, dado que las diferencias finitas ya no eran nula, hemos desarrollado el estudio realizando divisiones entre los diferentes términos de estas diferencias, por lo que vemos que la máquina tiene una complejidad exponencial.
En este caso en concreto, hemos determinado que las divisiones tienen una tendencia a 2, debido a que a medida que aumentamos los niveles donde se realizan estas divisiones, cada vez hay una convergencia más cercana a 2, por lo que la complejidad se podría aproximar a:

$$T(n) = 2^{n} + C$$

donde la C es una constante que varía según el tamaño del problema pero que no podemos calcularla de forma exacta debido a la complejidad de éste.

❖ Cota superior asintótica para $n_0 = 10$:

$T(n) = 2 + C$

$S(n) \mid S(n) \geq T(n)\ para\ n > n_0 = 10$

$S(n) = k \times 2^n$

Aseguramos $S(n) \geq T(n) = 2^n + C$

$k \times 2^n \geq 2^n + C$

$k \geq 1 + \frac{C}{2^n}$

Sustituimos n por 10:

$$\mathbf{k > 1 + C/2^{10}}$$

Si considerásemos la constante C como irrelevante (es decir, igual a 0), habría una cota superior asintótica aproximada si $k > 1$.

4. Comparativa ejercicios 2 y 3:

● Determinar la complejidad de cada algoritmo:

Antes de pasar a comparar la complejidad de los algoritmos, queremos detallar que las comparaciones son siempre entre los peores casos de cada algoritmo, lo que también implica que hayamos escogido la misma suma. Un detalle de esto último es que nuestros algoritmos tienen el peor caso de suma (tanto en base 1, como binaria) al contrario que los proporcionados de una cinta. Por ejemplo, si el peor caso en una cinta es 2 + 3, el peor caso del mismo algoritmo pero con dos cintas es 3 + 2.

- Ejercicio 2 (suma base 1):

Tamaño n	2	4	6	8	10
Pasos 1 cinta	6	28	66	120	190
Pasos 2 cintas	2	6	10	14	18

- Ejercicio 3 (suma binaria):

Tamaño n	2	3	4	5	6	7	8	9	10	11	12
Pasos 1 cinta	17	21	48	107	238	529	1172	2583	5658	12317	26656
Pasos 2 cintas	6	15	34	73	152	361	630	1269	2548	5107	10226

● ¿Por qué la diferencia de complejidades?

En el primer caso (suma en base 1) se obtiene una importantísima mejora en la complejidad debido al no uso de la aritmética de Peano, dado que al no tener que recorrer de izquierda a derecha y viceversa en cada iteración, y hacer solo 1 vez, se obtiene una complejidad muchísimo más baja para las mismas sumas.

En el segundo caso (suma binaria) se ha seguido usando la aritmética de Peano y ahora la complejidad se ha visto reducida debido a la adición de una cinta extra para realizar las operaciones. En consecuencia, con 1 cinta hay que recorrer de izquierda a derecha ambos términos (los cuales al haber solo 1 cinta están unidos por el "$"), resultando un recorrido mayor cada vez comparado con el uso de 2 cintas, donde en cada cinta se recorre únicamente uno de los dos números, pero no ambos.

Con esta mejora se obtiene una suma binaria mucho más eficiente, dentro de que ambas complejidades son exponenciales.

- ## ¿Cómo se interpretan las diferencias en complejidad y en eficiencia?

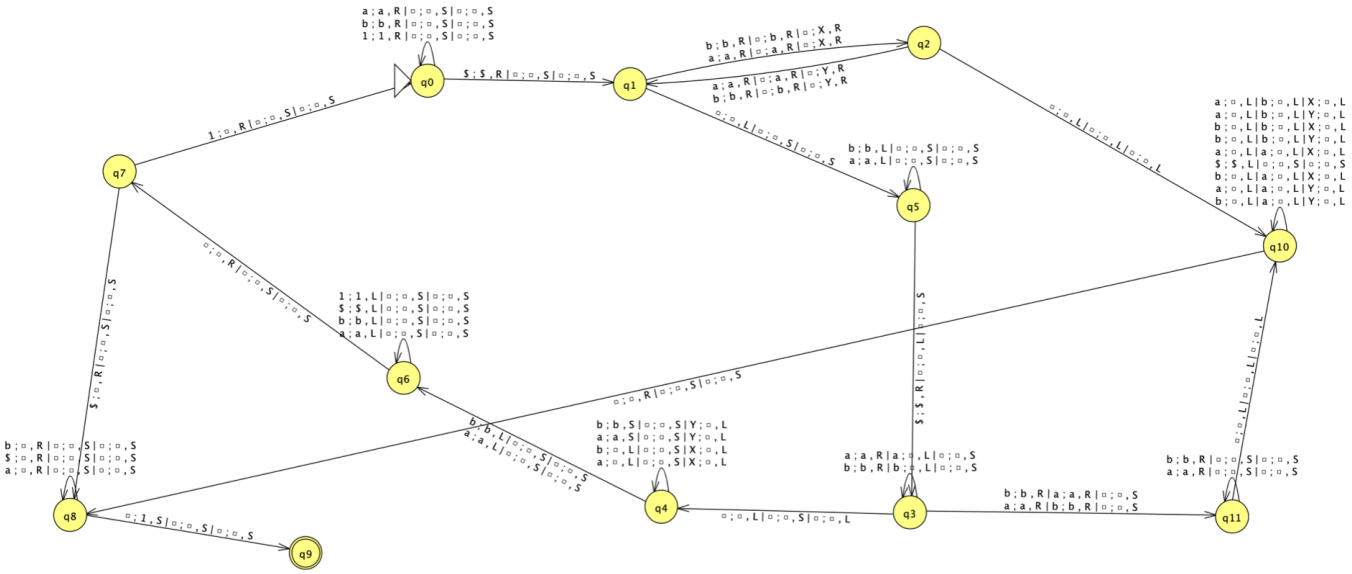
Inicialmente el indicador que hemos utilizado para determinar qué máquina es más eficiente ha sido el número de pasos, con lo cual se ve claramente que las máquinas desarrolladas con 2 cintas han necesitado un número de pasos mucho menor que la versión de 1 cinta. Esta diferencia de pasos se hace mucho más grande a medida que aumenta el tamaño de la entrada, llegando por ejemplo a la suma binaria de tamaño 12, donde la máquina de 1 cinta necesita más de 26000 pasos mientras que la de 2 cintas necesita poco más de 10000 para la misma suma.

Está claro que a medida que aumenta la eficiencia disminuye la complejidad y este aumento de eficiencia ha sido consecuencia de optimizar estos algoritmos iniciales y trasladarlos a máquinas de 2 cintas donde el "trabajo" a realizar quedaba repartido entre ambas. Esto no era así en 1 cinta, dado que los recorridos son mucho mayores y por ende, la eficiencia es menor en comparación con la máquina anterior.

5. Palíndromos de orden k:

● MT Multicinta (3 Cintas) Determinista:

❖ Diseño propuesto:



Funcionamiento:
Saltamos los 1's y al encontrarnos el "$", se copia la palabra en la 2ª cinta y una sucesión de XYXY… en la 3ª cinta. Esto nos sirve para eliminar justo la mitad de los símbolos de la palabra (al borrar un símbolo de la 1ª cinta solo cuando hay una X, de manera que al haber las mismas X's que Y's, borramos justo la mitad).

Tras haber comprobado que es palíndromo y borrar la mitad de la palabra, recorremos la 1ª cinta hacia la izquierda hasta encontrar un hueco en blanco, yendo a la posición a la derecha. Si lo encontrado tras el hueco en blanco es un 1, se deja en blanco y se repite todo el proceso. Si en cambio es un "$", se borra la 1ª cinta e indicamos que el palíndromo es de al menos orden k (escribiendo un 1 en la 1ª cinta).

Si la palabra es impar o deja de ser palíndromo, transicionará a un estado auxiliar donde se comprobará si queda algún 1 restante, si es así la máquina falla. En caso contrario, borrará toda la 1ª cinta y escribirá un 1.

Estados encargados:
Ciclo si es palíndromo → q_0 - q_1 - q_2 - q_5 - q_3 - q_4 - q_6 - q_7. Si queda algún 1 transiciona a q_0 de nuevo. Si es un "$", transiciona a q_8 - q_9, siendo este último el estado final.

Si es impar la secuencia → q_2 - q_{10} , y si no hay 1's, transiciona a q_8 .

Si la secuencia no es palíndromo → q_3 - q_{10} , y si no hay 1's, transiciona a q_8 .

❖ Peor caso en cuanto a coste computacional:

El peor caso en cuanto a coste computacional sería cuando el palíndromo es tamaño par (y los sub-palíndromos también lo son) y se compara con un orden igual al que es, dado que así deberá haber más iteraciones para determinar que el palíndromo es al menos de ese orden.

Un ejemplo de este tipo de entrada sería 11$abbaabba, donde la palabra es un palíndromo de orden 2, así debería hacer iteraciones hasta determinar que es de orden 2.
Si fuese 1$abbaabba, con una iteración menos le valdría al algoritmo para determinar que la palabra es de al menos orden 1.

❖ Estudio empírico:

Hemos separado los casos en función del orden que sea la palabra a evaluar, dado que el peor caso para cada palabra es que sea del mismo orden, como ya hemos comentado.

Orden 1:

Tamaño n	4	6	8	10	12	14	16	18	20
Pasos	26	43	48	69	70	95	92	121	114

Orden 2:

Tamaño n	7	11	15	19	23	27
Pasos	55	90	113	152	171	214

❖ Estudio analítico. Diferencias finitas y T(n):

En este caso solo nos centramos en los pasos obtenidos con orden 1 por facilidad de cálculo. Hemos podido observar que hay dos comportamientos distintos dentro del mismo orden según sea el tamaño de la entrada.
Para las entradas T(4), T(8), T(12), etc hay un comportamiento distintos que las entradas T(6), T(10), T(14), etc. Aún así, este comportamiento de ambas coincide en que es polinómico que se puede aproximar por diferencias finitas a un polinomio de primer grado.

Tabla 1:

Tamaño n	4	8	12	16	20
Pasos	26	48	70	92	114
Dif. 1		22	22	22	22
Dif. 2			0	0	0

Tabla 2:

Tamaño n	6	10	14	18
Pasos	43	69	95	121
Dif. 1		26	26	26
Dif. 2			0	0

Al ser constantes las Diferencias Finitas primeras (y nulas las segundas) es posible aproximar T(n) con un polinomio de primer grado.

Para la Tabla 1:

$T(n) = an + b$
$T(4) = 26$
$T(8) = 48$

T(4) = 4a + b = 26

T(8) = 8a + b = 48

a = 11/2

b = 4

$$T(n) = \frac{11}{2}n + 4$$

Para la Tabla 2:

$T(n) = an + b$
$T(6) = 46$
$T(10) = 69$

T(6) = 6a + b = 46

T(10) = 10a + b = 69

a = 23/4

b = 23/2

$$T(n) = \frac{23}{4}n + \frac{23}{2}$$

❖ Cota superior asintótica para $n_0 = 10$:

Para la Tabla 1:

$T(n) = \frac{11}{2}n + 4$

$S(n) \mid S(n) \geq T(n)\ para\ n > n_0 = 10$

$S(n) = kn$

Aseguramos $S(n) = kn \geq T(n) = \frac{11}{2}n + 4$

$kn \geq \frac{11}{2}n + 4$

$k \geq \frac{11}{2} + \frac{4}{n}$

Sustituimos la n por 10:

k > 5,9

Para la Tabla 2:

$T(n) = \frac{23}{4}n + \frac{23}{2}$

$S(n) \mid S(n) \geq T(n)\ para\ n > n_0 = 10$

$S(n) = kn$

Aseguramos $S(n) = kn \geq T(n) = \frac{23}{4}n + \frac{23}{2}$

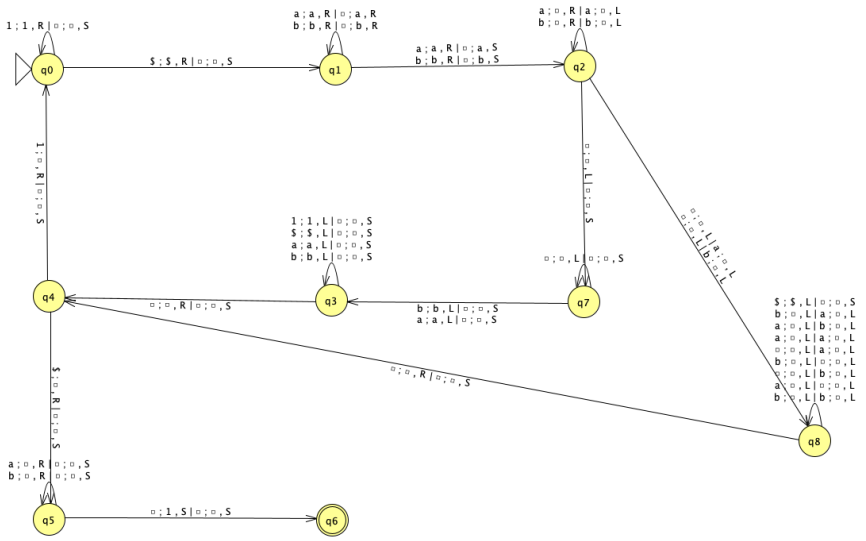
$kn \geq \frac{23}{4}n + \frac{23}{2}$

$k \geq \frac{23}{4} + \frac{23}{2n}$

Sustituimos la n por 10:

k > 6,9

- **MT Multicinta (2 Cintas) No Determinista:**
  - **Diseño propuesto:**



Funcionamiento:
Saltamos los 1's y al encontrarnos el "$", se copia la primera mitad de la palabra en la 2ª cinta. Aquí es donde podemos ver una clara ventaja del no determinismo, dado que antes necesitábamos una cinta extra con una secuencia copiada para poder quedarnos con la mitad de la palabra mientras que ahora automáticamente (a través de ir dividiéndose y creando nuevas instancias de la MT) podemos obtener la mitad de la palabra, logrando así una eficiencia superior.

Tras copiar la primera mitad de la palabra en la 2ª cinta, comparamos esto con la segunda mitad de la palabra de la 1ª cinta, y si son iguales, la palabra es un palíndromo. Mientras comparamos, borramos la segunda mitad de la 1ª cinta para así quedarnos con la primera mitad de cara a futuras iteraciones (para así lograr aplicar el algoritmo recursivamente).

Tras ver que es un palíndromo, al igual que en el caso determinista con 3 cintas, recorremos hasta la izquierda de la 1ª cinta, hasta encontrar un hueco blanco. Si tras este hueco hay un 1, se borra y se vuelve a realizar todo el algoritmo. Si en cambio hay un "$", aceptamos que la palabra original es un palíndromo de orden k, escribiendo un 1 en la 1ª cinta.

Al igual que antes, si la palabra es impar o deja de ser palíndromo, se recorre hacia la izquierda y vemos si quedan 1's, si es así la MT da un fallo. Si no quedan y lo que hay es un "$", se realiza el proceso de aceptar, comentado en el párrafo anterior.

Estados encargados:
Ciclo si es palíndromo → q_0 - q_1 - q_2 - q_7 - q_3 - q_4. Si queda algún 1 transiciona a q_0 de nuevo.
Si es un "$", transiciona a q_5 - q_6, siendo este último el estado final.

Si es impar la secuencia o no es palíndromo → q_2 - q_8, y si no hay 1's, transiciona a q_4.

❖ Peor caso en cuanto a coste computacional:

El peor caso en cuanto a coste computacional sigue siendo el mismo que en el caso de la máquina determinista de 3 cintas.

❖ Estudio empírico:

Orden 1:

Tamaño n	4	6	8	10	12	14
Pasos	19	25	31	37	43	49

Orden 2:

Tamaño n	7	11	15	19	23	27
Pasos	36	50	65	78	92	106

❖ Estudio analítico. Diferencias finitas y T(n):

Orden 1:

Tamaño n	4	6	8	10	12	14
Pasos	19	25	31	37	43	49
Dif. 1		6	6	6	6	6
Dif. 2			0	0	0	0

Orden 2:

Tamaño n	7	11	15	19	23	27
Pasos	36	50	64	78	92	106
Dif. 1		14	14	14	14	14
Dif. 2			0	0	0	0

Al ser constantes las Diferencias Finitas primeras (y nulas las segundas) es posible aproximar T(n) con un polinomio de primer grado.

$T(n) = an + b$
$T(4) = 19$
$T(6) = 25$

T(4) = 4a + b = 19

T(6) = 6a + b = 25

a = 3

b = 7

$$T(n) = 3n + 7$$

❖ Cota superior asintótica para $n_0 = 10$:

$T(n) = 3n + 7$

$S(n) \mid S(n) \geq T(n) \; para \; n > n_0 = 10$

$S(n) = kn$

Aseguramos $S(n) = kn \geq T(n) = 3n + 7$

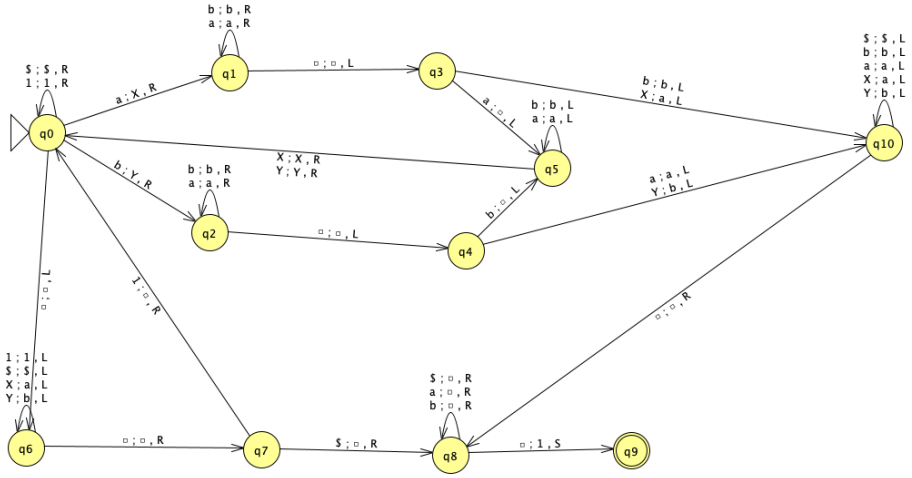
$kn \geq 3n + 7$

$k \geq 3 + \frac{7}{n}$

Sustituimos la n por 10:

k > 3,7

- **MT Determinista 1 cinta (PARTE OPCIONAL):**
  - **Diseño propuesto:**



Funcionamiento:
En este caso operamos parecido a los casos anteriores. La clave de este algoritmo para poder desarrollarlo con una cinta es sustituir el elemento de la izquierda de la palabra por un símbolo de marcado (en nuestro caso X por a e Y por b) y borrar el símbolo de la derecha (que al ser palíndromo deberían ser el mismo). Con esto conseguimos que tras la iteración tengamos en la cinta la mitad de la palabra con símbolos de marcado, y si los desmarcamos se puede volver a aplicar el algoritmo a la mitad de la palabra original.

Tras saltar los 1's y el "$", se realiza la comprobación de si la palabra es palíndromo comentada anteriormente. Si efectivamente lo es, recorremos hasta la izquierda (igual que en los dos casos anteriores) y vemos si después del hueco blanco hay un 1 (se borra y empieza de nuevo el algoritmo con la mitad de la palabra original) o un "$", en cuyo caso se acepta la palabra, dejando escrito un 1 en la cinta.

Si la palabra es impar o deja de ser palíndromo, transiciona a un estado donde se vuelve a la izquierda para ver si quedan 1's, si no es así, se acepta la palabra original como palíndromo de al menos orden k.

Estados encargados:
Ciclo si es palíndromo → q_0 - [q_1 - q_3] - [q_2 - q_4] - q_5. Si queda algún 1 transiciona a q_0 de nuevo. Si es un "$", transiciona a q_6 - q_7 - q_8 - q_9 siendo este último el estado final.

Si es impar la secuencia o no es palíndromo → [q_3 - q_4] - q_{10}, y si no hay 1's, transiciona a q_8.

❖ Peor caso en cuanto a coste computacional:

El peor caso en cuanto a coste computacional sigue siendo el mismo que en el caso de la máquina determinista de 3 cintas y la no determinista de 2 cintas.

❖ Estudio empírico:

Complejidad 1:

Tamaño n	4	6	8	10	12	14
Pasos	22	35	52	73	98	127

Complejidad 2:

Tamaño n	7	11	15	19	23	27
Pasos	47	92	157	242	347	472

❖ Estudio analítico. Diferencias finitas y T(n):

Complejidad 1:

Tamaño n	4	6	8	10	12	14
Pasos	22	35	52	73	98	127
Dif. 1		13	17	21	25	29
Dif. 2			4	4	4	4
Dif. 3				0	0	0

Complejidad 2:

Tamaño n	7	11	15	19	23	27
Pasos	47	92	157	242	347	472
Dif. 1		45	65	85	105	125
Dif. 2			20	20	20	20
Dif. 3				0	0	0

Al ser constantes las Diferencias Finitas segundas (y nulas las terceras) es posible aproximar T(n) con un polinomio de segundo grado.

$T(n) = an^2 + bn + c$

$T(4) = 22$

$T(6) = 35$

$T(8) = 52$

$$\begin{cases} T(4) = 16a + 4b + c = 22 \\ T(6) = 36a + 6b + c = 35 \\ T(8) = 64a + 8b + c = 52 \end{cases}$$

a = ½
b = 3/2
c = 8

$$T(n) = \frac{1}{2}n^2 + \frac{3}{2}n + 8$$

❖ Cota superior asintótica para $n_0 = 10$:

$T(n) = \frac{1}{2}n^2 + \frac{3}{2}n + 8$

$S(n) \mid S(n) \geq T(n)\ para\ n > n_0 = 10$

$S(n) = kn^2$

Aseguramos $S(n) \geq T(n) = \frac{1}{2}n^2 + \frac{3}{2}n + 8$

$kn^2 \geq \frac{1}{2}n^2 + \frac{3}{2}n + 8$

$k \geq \frac{1}{2} + \frac{3}{2n} + \frac{8}{n^2}$

Sustituimos n por 10:

k > 0,73

- **Evaluación de la mejora obtenida con la MT Multicinta NO Determinista.**

Tras haber realizado el diseño determinista con 1 y 3 cintas y el diseño no determinista con 2 cintas, hemos podido comprobar claramente la utilidad tanto de utilizar cintas extra como de implementar el no determinismo en una MT.

Comparando las versiones deterministas, claramente al añadir dos cintas más a la máquina hemos conseguido reducir en un grado la complejidad polinomial de $O(n^2)$ a $O(n)$. Esto es una mejora importante en la eficiencia.

Comparando ahora la versión determinista de 3 cintas con la no determinista de 2 cintas hemos podido observar un ligero aumento en la eficiencia por parte de la no determinista, dado que aún siendo las dos máquinas de orden $O(n)$, en la versión no determinista se consigue una disminución considerable de casi la mitad o más pasos (según el caso) que en la versión determinista.

Todo este aumento de eficiencia se debe a que al utilizar el no determinismo podemos obtener la mitad de la palabra sin necesidad de tener otra cinta donde copiar unos símbolos y borrar una parte de ellos, lo cual hace que cada iteración del problema se reduzca considerablemente en pasos al no tener que recorrer de nuevo toda la cinta borrando estos símbolos en un estado concreto.

Otra ventaja del no determinismo es que si la palabra no es palíndromo o es impar, es que hay un solo estado para tratar estas excepciones, mientras que en la versión determinista hay 3 estados encargados de ello. Con esto también queremos mencionar que al haber menos estados (9 en la versión no determinista contra 12 de la otra), al haber menos saltos entre ellos, se consigue un menor número de pasos.

6. Conclusiones:

Esta práctica nos ha proporcionado una manera muy clara e intuitiva de observar y calcular la complejidad computacional asociada a las Máquina de Turing. Consideramos que nos ha sido de utilidad para añadir este aspecto al propio diseño de las máquinas, donde previamente no nos parábamos a pensar cuán complejo era el algoritmo que estábamos desarrollando.

Este problema de no optimizar una máquina para que sea menos compleja lo hemos sufrido en esta práctica, debido a que en el ejercicio 4 (palíndromos de orden k) empezamos desarrollando unas máquinas monstruosas, de incluso 30 estados y 4 cintas.

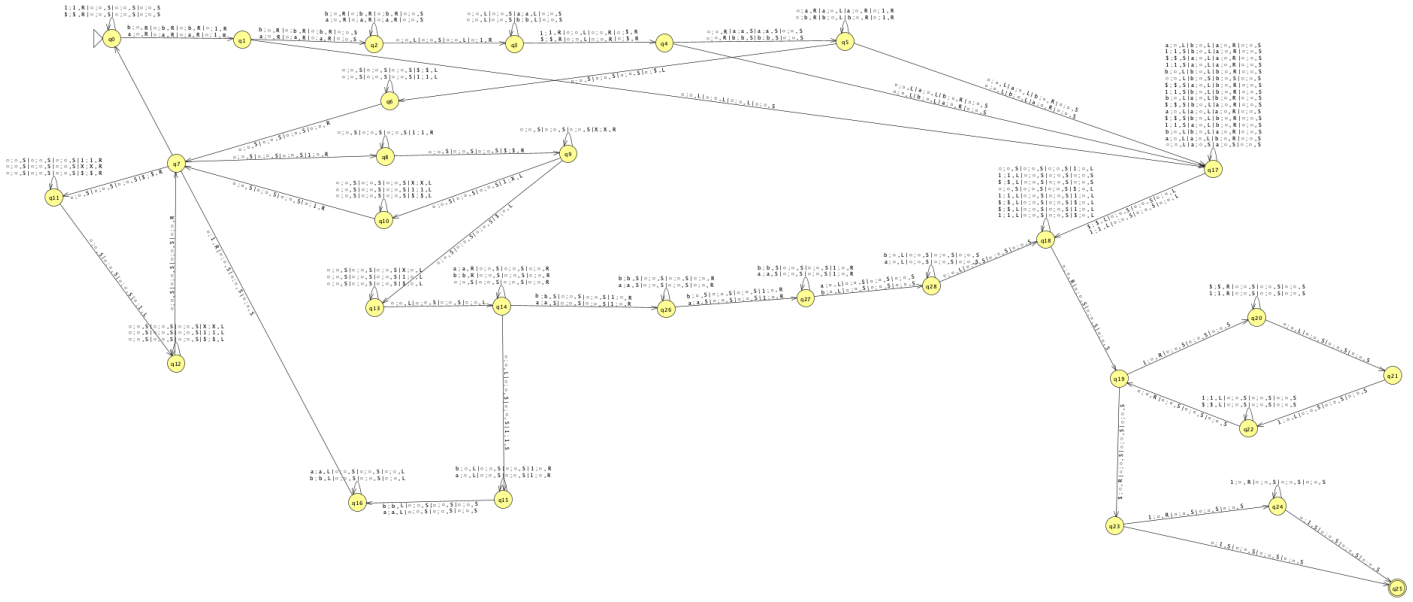
Estas máquinas sí que realizaban la tarea que se pedía, pero tenían una complejidad exponencial debido a la poca optimización de los algoritmos. Fue entonces cuando realizamos una optimización considerable en todas las máquinas, consiguiendo reducir el orden de exponencial a polinómico.

Un pequeño problema con el que nos hemos vuelto a topar (desde que terminamos TALF) es la dificultad de implementar una Máquina de Turing No Determinista, debido a que es complicado imaginarse su funcionamiento de manera secuencial, debido a las sucesivas divisiones en diferentes instancias de la máquina. Esto obviamente es lo que le aporta el beneficio de implementarlo, pero en un inicio es difícil comprenderlo.

A continuación, mostraremos en el anexo el diseño de las primeras máquinas del ejercicio 4, con una pequeña descripción de su funcionamiento.

7. Anexo:

- **MT Multicinta (4 Cintas) Determinista:**



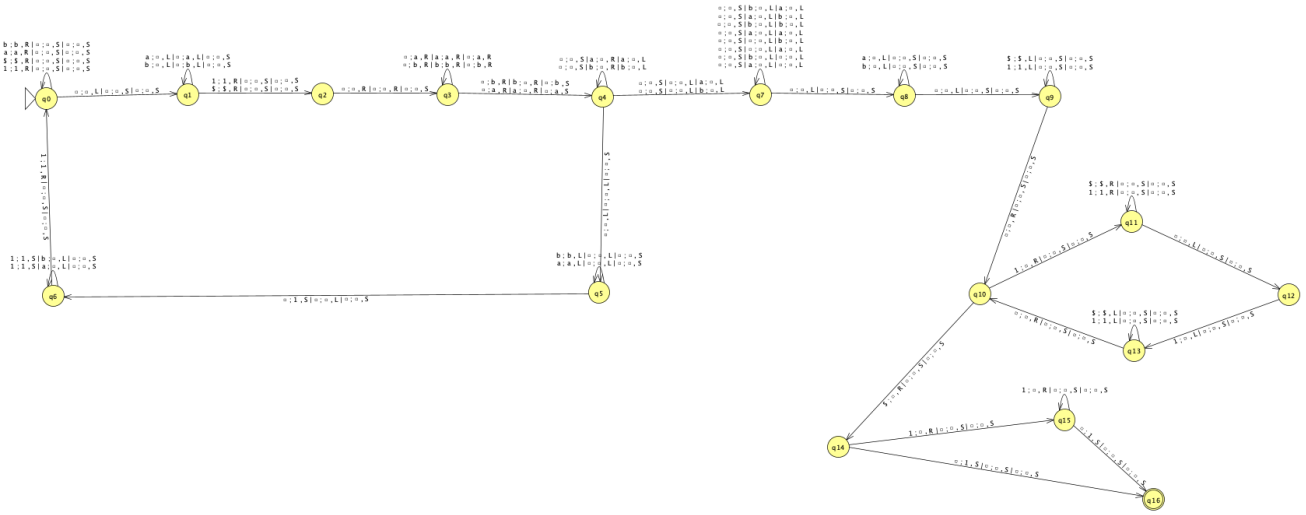
Esta fue nuestra primera versión de los palíndromos de al menos orden k, donde el algoritmo se basaba en copiar la palabra de la 1ª cinta en la 2ª y 3ª (en estas dos se hacía la comparación de si la palabra era palíndromo).

Mientras se copiaba en estas cintas, en la 4ª cinta se copiaban tantos 1's como caracteres tuviese la palabra, para así realizar su división por 2, quedándonos con los caracteres que debían ser eliminados en la 1ª cinta para obtener la mitad de la palabra original y poder realizar el algoritmo recursivamente.

Un detalle a comentar de esta máquina es que no calculaba si la palabra era palíndromo de al menos orden k, sino que primero calculaba su orden (escribiendo en la 1ª cinta, tras el "$", tantos 1's como el orden del palíndromo fuera) y luego calculaba el orden que se comprobaba con el orden que se obtuvo tras la ejecución del algoritmo.

Por ejemplo, la entrada 1$aaaa, primero realiza el algoritmo, deja en la 1ª cinta 1$11 (dado que 'aaaa' es un palíndromo de orden 2), borrando el contenido de las otras 3 cintas. Acto seguido, compara los dos números y como el orden del palíndromo es mayor que el orden k solicitado para comprobar $(1 < 2)$, se acepta la entrada y se escribe un 1 en la 1ª cinta, transicionando así al estado final.

- **MT Multicinta (3 Cintas) No Determinista:**



En este caso, al poder aplicar el no determinismo, no necesitamos la división para calcular la mitad de la palabra. Con esta primera versión, realizamos el mismo procedimiento que con la máquina anterior, que consiste en calcular el orden de la palabra y luego compararlo con el orden solicitado.

Lo que descubrimos es que esto era inútil, pues cuando la máquina llegase al orden solicitado, iba a aceptar la palabra sin tener que calcular el orden total. Debido a esto nos dimos cuenta de que la comparación de los números era inútil y por ello también la necesidad de primero calcular el orden de la palabra.

Fue este descubrimiento el que nos impulsó a optimizar estas dos máquinas, con algoritmos mucho más eficientes (explicados anteriormente en la memoria) y por ello con una complejidad mucho menor.