Computabilidad y Complejidad
Ejercicios Unidad Temática: Máquinas de Registros

Alumno: Díaz-Alejo León, Stéphane

Pueden utilizarse además de las instrucciones básicas las macros ya definidas

1.-Desarrolle un programa para la máquina contador que realice la computación `[i]←[j][k]` donde los registros involucrados no son necesariamente distintos.

Consideremos dos registros distintos Rq, Rp y Rm distintos de Ri, Rj y Rk.

```
        cop(j,q)
        cop(k,m)
        asi(1,p)
bucle:  pre(m,fin)
        mul(q,p,p)
        goto(bucle)
fin:    cop(p,i)
```

2.-Sea la función f: N ⟶ N definida de modo que:

- **f(n) = 1, si n es par**
- **n^n, si n es impar**

Desarrolle un programa para la máquina contador que realice la computación `[i]←f([j])` donde los registros involucrados no son necesariamente distintos.

Consideremos dos registros distintos Rq y Rp distintos de Ri y Rj. A su vez definimos el macro exp(j,k,i) tomando como definición el ejercicio anterior.

```
        cop(j,q)
        div(q,2,p)
        mul(p,2,p)
        ig(p,q,par,impar)
par:    asi(1,p)
        goto(fin)
impar:  exp(q,q,p)
fin:    cop(p,i)
```

3.-Sea la función f: N×N×N → N definida de modo que f(a,b) = g(h(a,b)), donde $h(a,b)=a^b$ y $g(n)=2^n/n$.

Desarrolle un programa para la máquina contador que realice la computación `[k]←f([i],[j])` donde los registros involucrados no son necesariamente distintos.

Consideremos dos registros distintos Rq, Rp y Rm distintos de Ri, Rj y Rk. A su vez definimos el macro exp(j,k,i) tomando como definición el primer ejercicio.

```
        cop(j,q)
        cop(i,m)
h(a,b): exp(m,q,p)
g(n):   exp(2,p,q)
        div(q,p,m)
fin:    cop(m,i)
```

Ejercicios Unidad Temática: Complejidad Computacional

1.-Sea L el lenguaje L = {z / ∃x∈{a,b}*: z = xx}. Calcule del modo más ajustado posible:

- **Una función f(n) de modo que L ∈ DTIME(f(n)).**
- **Una función g(n) de modo que L ∈ DSPACE(g(n)).**

F(n) sería una función en la que primero mediríamos la mitad de la longitud de la cadena de entrada. Para esto iremos procesando la cadena de entrada escribiendo un símbolo "a" en una segunda cinta en los pasos pares y sobrescribiremos el símbolo "a" por un símbolo "b". Si al acabar de procesar la cadena de entrada el último símbolo de la segunda cinta es una "a", rechazaríamos, en el caso contrario, seguiríamos el proceso.

Desplazamos el cabezal de la cinta de entrada tantas posiciones cómo "b" tenemos en la segunda cinta, copiando a su vez los símbolos de la cinta de entrada en una tercera cinta. Con esto tendríamos el cabezal de la cinta de entrada en la mitad y una x copiada en la tercera cinta. Ahora solo nos faltaría comprobar la x con la primera mitad de la cinta de entrada, si todas las comprobaciones salen igual, aceptamos, en el caso contrario, rechazamos.

El coste temporal de esta función se podría dividir en:

- Calcular la mitad de la cadena de entrada: **O(n)**.
- Mover el cabezal a la mitad: **O(n/2)**.
- Comprobar la otra mitad de la cadena de entrada: **O(n/2)**.

siendo n la longitud de la cadena de entrada.

El coste temporal total sería **O(2n)** -> **DTIME(n)**.

G(n) sería una función que comenzaría igual que f(n) para calcular la mitad de la cadena de entrada. Una vez hecho esto, tendríamos que comparar los símbolos de las dos mitades uno a uno, para ello, los símbolos de la parte izquierda los tacharemos con un símbolo "n" y los de la derecha con un símbolo "n", todo en la segunda cinta. Cabe mencionar que la primera vez se moverán los cabezales al principio, se tachará el primer símbolo y después el cabezal se moverá el número de símbolo que tengamos en la segunda cinta, hacia la derecha. Después es una simple cuestión de moverse hasta encontrar el símbolo "n" o "m" hasta procesar toda la cadena de entrada. Si en todas las comprobaciones sale que tienen el mismo símbolo, entonces aceptamos, en caso contrario, se rechaza.

El coste espacial de esta función se podría dividir en:

- Calcular la mitad de la cadena de entrada: **O(n/2)**.

siendo n la longitud de la cadena de entrada.

El coste temporal total sería **O(2n)** -> **DSPACE(n)**.

2.-Sea L el lenguaje L = {$x^{|x|}$ / x∈{a,b}* }. Calcule del modo razonablemente más ajustado posible una función T(n) de modo que L sea reconocido por una MT no determinista con complejidad temporal O(T(n)).

T(n) sería una función en la que se comenzaría a copiar x en una segunda cinta, tercer y cuarta cinta, y que, de forma no determinista, parase, y empezara a comparar la cadena copiada con la cinta. Esta comparación se haría de la siguiente forma: la segunda y tercera cinta tendrían los cabezales posicionados al principio y el otro al final de manera que cada vez que se produce un paso se mueven en sentido inverso. Con esto, podemos empezar las comparaciones con la segunda cinta, luego con la tercera y de nuevo con la segunda, realizando este ciclo hasta terminar de comprobar toda la cadena de entrada. Si alguna comprobación fallase, se rechazaría. A su vez por cada vez que terminamos de comparar la segunda o tercera cinta borraríamos un símbolo de la cuarta cinta, de manera que si hemos borrado todos menos un símbolo y no nos queda por borrar ninguno más habríamos comprobado que la longitud de la cadena de entrada es |x| veces x. En concreto, esto se podría comprobar retrocediendo y encontrándonos un blanco si la máquina de Turing es infinita por la izquierda, o habiendo marcado el inicio de la cadena de entrada en otra cinta o en ésta misma con un símbolo especial.

El coste temporal de esta función se podría dividir en:

- Copiar la cinta de entrada: **O(m)**.
- Mover el cabezal al principio de la segunda cinta: **O(m)**.
- Comprobar la cinta de entrada: **O(n - m)**.

siendo n la longitud de la cadena de entrada y m la longitud de x, teniendo en cuenta que n > m.

El coste temporal total sería **O(n + m)** -> **NTIME(n)**.

3.-Sea Σ = {a,b}. Sea L el lenguaje L = {$a^n b^m a^{n+m} b^m a^n$ /n,m ≥ 1 }. Calcule, del modo razonablemente más ajustado posible, una función T(n) de modo que L sea reconocido por una MT determinista con complejidad temporal O(T(n)).

T(n) sería una función en la que se procede de la siguiente manera:

- En una segunda cinta se copian los símbolos “a” del principio hasta encontrar símbolos “b”. Con esto hemos obtenido a^n.
- En una tercera cinta se copian los símbolos “b” que preceden a los símbolos “a” del principio hasta volver a encontrarnos con un símbolo “a”, obteniendo b^m.
- En una cuarta cinta se escriben tantos símbolos “a” como pasos hemos dados en los últimos dos puntos. Así obtenemos a^{n+m}.

Una vez finalizada está parte sólo quedaría comparar los símbolos “a” que hay hasta encontrar símbolos “b” con la cuarta cinta, después los símbolos “b” con la tercera, y finalmente los símbolos “a” con la segunda. Si alguna comprobación falla rechazaríamos, y si todas resultan correctas, aceptaríamos.

El coste temporal de esta función se podría dividir en:

- Copiar las partes de la cadena de entrada: O(n/3).
- Comprobar el resto de la cadena de entrada: O(2*n/3)

siendo n la longitud de la cadena de entrada.

El coste temporal total sería **O(n)** -> **DTIME(n)**.