IIC1001 – Algoritmos y Sistemas Computacionales.
Solución Ejercicios I2
2-junio-2023

Ejercicio 1:

Alice está desarrollando una nueva técnica para almacenar strings de forma que utilicen menos espacio en la memoria del computador. En su estudio, descubrió que hay muchas palabras que se leen igual de izquierda a derecha y viceversa, como "oso", "radar" o "reconocer". Estas palabras son conocidas como palíndromos.

Por ejemplo, la palabra "reconocer" cuando se codifica en ASCII, se representa de la siguiente manera:

```
 r  e  c  o  n  o  c  e  r
72 65 63 6F 6E 6F 63 65 72
```

Alice notó que en estas palabras se repiten muchos bytes. Entonces, con el fin de ahorrar espacio, Alice decidió que al encontrar un palíndromo sólo almacenaría la primera mitad de la palabra, seguida de un byte especial 2A para indicar que la palabra es un palíndromo y otro byte adicional correspondiente a la longitud total de la palabra. Por lo tanto, en la nueva codificación de Alice, la palabra "reconocer" se almacenaría de esta manera:

```
 r  e  c  o  n  *  9
72 65 63 6F 6E 2A 09
```

De esta forma no se pierde información y se ahorran 2 bytes de memoria. Nota: El byte que denota la longitud de la palabra no debe estar codificado en ASCII, sino que debe convertirse de decimal a hexadecimal. El resto de las palabras se codifican de manera normal usando ASCII.

a) Codifica el siguiente string usando la codificación de Alice.

```
nos sometemos al rio, solos en el kayak de bob
```

Re:

```
6E 6F 73 20 73 6F 6D 65 74 2A 09 20 61 6C 20 72
69 6F 2C 20 73 6F 6C 2A 05 20 65 6E 20 65 6C 20
6B 61 79 2A 05 20 64 65 20 62 6F 2A 03 00 00 00
```

b) Decodifica el siguiente string que fue almacenado usando la codificación de Alice:

```
61 6E 2A 03 20 75 73 6F 20 73 75 20 72 61 64 2A
05 20 70 61 72 61 20 72 65 63 6F 6E 2A 09 20 65
6C 20 6B 61 79 2A 05 00 00 00 00 00 00 00 00 00
```

Re:

```
ana uso su radar para reconocer el kayak
```

Ejercicio 2:

Alice y Bob pronto tendrán que cuidar la I2 de Introducción a la Programación. Para no tener problemas como la vez anterior, deciden minimizar el tiempo que pierden pasando lista guardando todos los nombres de los alumnos en una tabla de hash.

Para esto Alice define la siguiente función de hash:

$$f_a(\text{nombre}) = \text{largo del nombre} + \text{cantidad de letras 'a'}$$

Por ejemplo:

$$f_a(\text{“María”}) = 5 + 2 = 7$$
$$f_a(\text{“Víctor”}) = 6 + 0 = 6$$

Bob por su lado define otra función de hash:

$$f_b(\text{nombre}) = 3 * \text{cantidad de letras 'i'} + \text{cantidad de letras 'a'}$$

Por ejemplo:

$$f_b(\text{“María”}) = 3*1 + 2 = 5$$
$$f_b(\text{“Víctor”}) = 3*1 + 0 = 3$$

Considerando que la lista tiene los siguientes nombres:

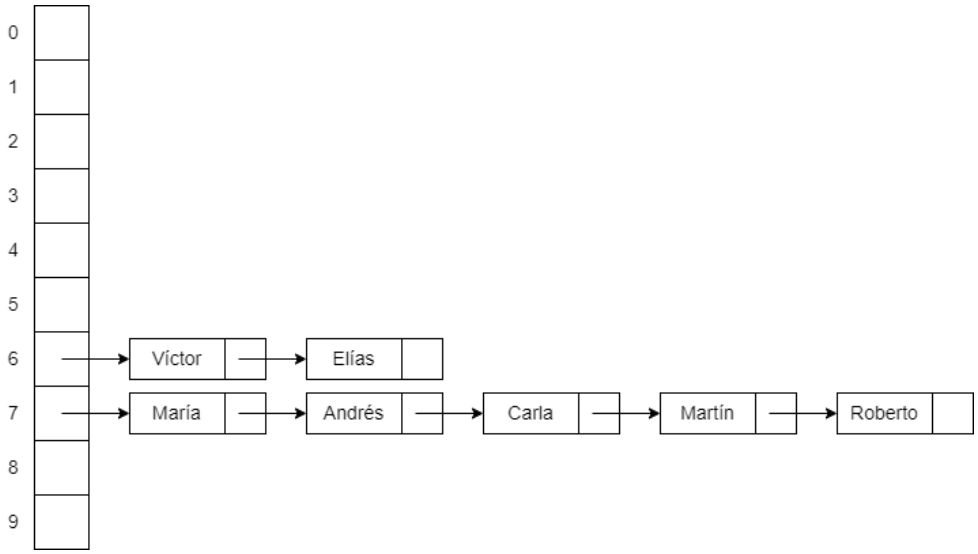
María, Víctor, Andrés, Carla, Martín, Roberto, Elías

a) Construye una tabla de hash usando la función de Alice

Re: *Primero se le aplica la función de hash a cada nombre de la lista:*

$f_a(\text{“María”}) = 5 + 2 = 7$
$f_a(\text{“Víctor”}) = 6 + 0 = 6$
$f_a(\text{“Andrés”}) = 6 + 1 = 7$
$f_a(\text{“Carla”}) = 5 + 2 = 7$
$f_a(\text{“Martín”}) = 6 + 1 = 7$
$f_a(\text{“Roberto”}) = 7 + 0 = 7$
$f_a(\text{“Elías”}) = 5 + 1 = 6$

Luego se usan los valores obtenidos para guardar los nombres en una tabla de hash:

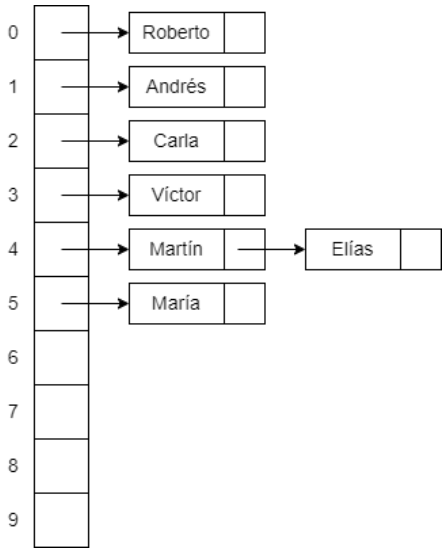


b) Construye otra tabla de hash usando la función de Bob

Re: *Se aplica la función a cada nombre, esta vez usando la función de Bob:*

f_b(“María”) = 3*1 + 2 = 5
f_b(“Víctor”) = 3*1 + 0 = 3
f_b(“Andrés”) = 3*0 + 1 = 1
f_b(“Carla”) = 3*0 + 2 = 2
f_b(“Martín”) = 3*1 + 1 = 4
f_b(“Roberto”) = 3*0 + 0 = 0
f_b(“Elías”) = 3*1 + 1 = 4

Luego se construye la tabla:



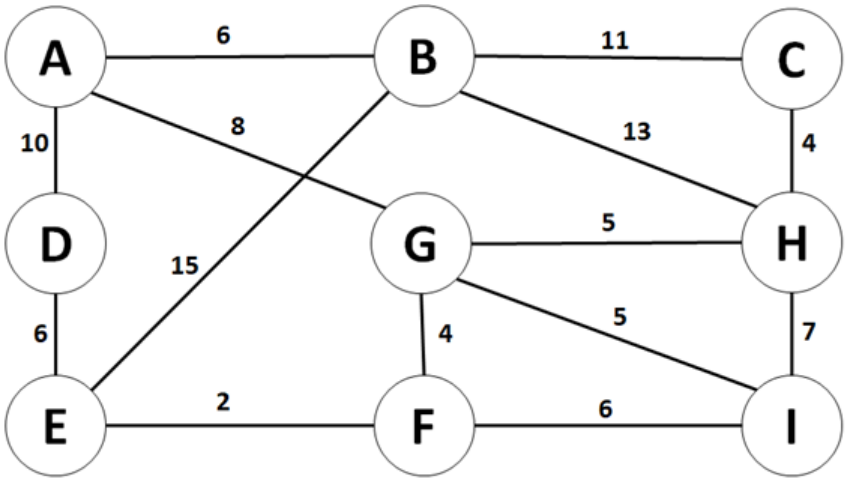
c) ¿Cuál función de hash es mejor para crear la tabla? ¿Por qué?

Re: *Idealmente, una buena función de hash debería distribuir los elementos de manera uniforme a lo largo de las posibles ubicaciones de la tabla. Esto ayuda a minimizar las colisiones, que ocurren cuando dos o más elementos resultan en el mismo valor de hash y deben colocarse en la misma ubicación en la tabla.*

Cuando evaluamos las funciones de Alice y Bob, podemos ver que la función de Alice resulta en una distribución con muchas colisiones. Todos, a excepción de Víctor y Elías, resultan en el mismo valor de hash (7). Por lo tanto, en este caso la función de hash de Bob es mejor porque resulta en menos colisiones y una mejor distribución de los nombres en la tabla, lo que significa un rendimiento más eficiente al buscar elementos en la tabla.

Ejercicio 3:

Utiliza el algoritmo de Dijkstra en el siguiente grafo:



a) Comenzando desde el nodo D, determina la distancia más corta hasta cada nodo
b) Determina los caminos desde D hasta cada nodo.

Re:

Nodo	Distancia	Predecesor	Camino
A	10	D	D > A
B	16	A	D > A > B
C	21	H	D > E > F > G > H > C
D	0		D
E	6	D	D > E
F	8	E	D > E > F
G	12	F	D > E > F > G
H	17	G	D > E > F > G > H
I	14	F	D > E > F > I