**Escuela Colombiana de Ingeniería Julio Garavito 26 de octubre de 2021**

**Parcial Teórico 02**

**Tiempo Límite:** 120 Minutos

**Profesor Encargado:** Sebastián Camilo Martínez Reyes

**Nombre del estudiante:** Jefer Alexis González Romero

**Número de Carné:** 2171737

**1.** (10 puntos) Dada la siguiente función recurrente, determine las funciones resultantes utilizando la técnica de memorización.

Utilizando la técnica de memorización:

**2.**

Se usará la estructura lineal lista enlazada doble, debido a que el problema nos plantea que debemos saber quien es la cabeza y la cola, además en los comandos se llegan a eliminar cualquiera de las dos.

El luchador en la cabeza es el que está combatiendo.

Los luchadores se llamarán A, B y C:

Tail

Head

|  |  |  |
| --- | --- | --- |
| B | C | None |

|  |  |  |
| --- | --- | --- |
| None | A | B |

|  |  |  |
| --- | --- | --- |
| A | B | C |

Referencia al siguiente

Referencia al anterior

Luchador

Comandos:

**ChgUp**

Lo primero que va a ser este comando es eliminar la cabeza, por lo cual, la referencia del anterior de la nueva cabeza es None

Head

Tail

|  |  |  |
| --- | --- | --- |
| None | B | C |

|  |  |  |
| --- | --- | --- |
| B | C | None |

Después se agregará el luchador que acabamos de eliminar a la cola, para esto la referencia al siguiente del que estaba como cola será el luchador que acabamos de añadir.

Tail

|  |  |  |
| --- | --- | --- |
| C | A | None |

Head

|  |  |  |
| --- | --- | --- |
| None | B | C |

|  |  |  |
| --- | --- | --- |
| B | C | A |

**ChgDown**

Lo primero que va a ser este comando es eliminar la cabeza y la cola, por lo cual, la referencia del siguiente y al anterior de la nueva cabeza es None

Head

|  |  |  |
| --- | --- | --- |
| None | C | None |

Después se agregará en la cabeza el luchador que estaba en la cola y el que estaba en la cabeza pasa a ser la cola, se cambian las referencias del luchador del medio.

Tail

|  |  |  |
| --- | --- | --- |
| C | B | None |

Head

|  |  |  |
| --- | --- | --- |
| None | A | C |

|  |  |  |
| --- | --- | --- |
| A | C | B |

**3.**

Para este caso de usará la estructura de datos Cola, ya que el primero en luchar es el que lleva mas tiempo en la lista.

Los nombres de los jugadores serán A, B, C, D

Se crea una variable llamada anfitrión en la cual se encuentra A

En un inicio la sala está vacía, después llega el primer jugador (B), por lo tanto, la sala queda de la siguiente manera:

Anfitrión = A

|  |  |
| --- | --- |
| B | None  Head |

***Entra el jugador C***

Anfitrión = A

Head

|  |  |
| --- | --- |
| B | C |

|  |  |
| --- | --- |
| C | None |

***Entra el jugador D***

Anfitrión = A

|  |  |
| --- | --- |
| D | None |

|  |  |
| --- | --- |
| C | D |

|  |  |
| --- | --- |
| B | C  Head |

***Primera partida:***

El anfitrión pierde contra B:

Se le asigna a la variable anfitrión B, se elimina la cabeza.

Anfitrión = B

|  |  |
| --- | --- |
| C | D |

|  |  |
| --- | --- |
| D | None |

Se ingresa a la sala el jugador que antes era anfitrión (A)

Anfitrión = B

|  |  |
| --- | --- |
| A | None |

|  |  |
| --- | --- |
| D | A |

|  |  |
| --- | --- |
| C | D  Head |

***Segunda partida:***

El ganador es el anfitrión:

Se elimina la cabeza.

Anfitrión = B

|  |  |
| --- | --- |
| D | A |

|  |  |
| --- | --- |
| A | None |

Se ingresa a la sala el jugador que perdió (C)

Anfitrión = B

|  |  |
| --- | --- |
| C | None |

|  |  |
| --- | --- |
| A | C |

|  |  |
| --- | --- |
| D | A  Head |

**4.**

Para el nuevo feature se usa la estructura de datos árbol binario de búsqueda, el cual como nos piden se representa como un árbol, además, los valores menores se ubican en los subárboles izquierdos y por ende los mayores en los derechos.

A continuación, se presenta un árbol con los registros históricos, donde en cada nodo se puede ver la *afinidad*, los nodos además tienen valores almacenados, que son la fecha exacta del encuentro y los personajes seleccionados para ese combate.

**20**

**5560**

**3260**

**1260**

**6660**

**2560**

Personajes

Fecha

23/10/2018 12:02

“Kakarott, VegitaSSj4, Gran saiyamen”

Si un nuevo combate ocurre, por ejemplo: (24/10/2018 14:05, “Tronks, Carnage, Gon”, 60). Entonces el árbol queda de la siguiente manera:

24/10/2018 14:05

Fecha

“Tronks, Carnage, Gron”

Personajes

**20**

**5560**

**3260**

**1260**

**6660**

**2560**

**6060**

**6060**

¿Qué pasa si un jugador gana?

Se asigna donde corresponda en el árbol, Esto depende de la afinidad conseguida, la cual se obtiene del tiempo restante en el contador de batalla y la vida de cada uno de los combatientes.

¿Qué pasa si un jugador pierde?

No se hace nada, debido a que si pierde, significa que ese *Roster* no es bueno*.*

¿Qué pasa cuando la partida no es válida?

Si no se vale la partida, no se tendría en cuenta, ya que por la razón que no se tiene cuenta puede estar ingresando al árbol una *Roster* que no merece la posición que se le asigne o por el contrario se deja de lado un buen *Roster*.

**5.**

(10 puntos) [Bono] Para las funciones resultantes usando la técnica de memorización del punto 1, ¿Cuál sería la estructura de datos que acompañaría a las implementaciones Top-down ?, ¿Cuál sería el estado de la memoria para N=10? Sustente su respuesta explicando con una simulación el llenado de la memoria

Si N = 10, entonces para esto se llega por recursión a que primero se debe calcular

*f\_*memo(0, M), N al ser igual a 0 entonces: *f\_*memo(0, M) = 1, se guarda que M[0] = 1.

Después se calcula para arriba de la siguiente manera:

*f\_*memo(1, M) = 1 *f*\_memo(0, M)

Como se debe calcular *f*\_memo(0, M), pero 0 ya está en M entonces:

*f\_*memo(1, M) = 1 M[0]

*f\_*memo(1, M) = 1 1

*f\_*memo(1, M) = 1

Se hace esto sucesivamente hasta llegar al 10:

*f\_*memo(10, M) = 10 M[9]

*f\_*memo(10, M) = 10 362880

*f\_*memo(10, M) = 3628800

El estado de la memoria para N=10 es 362880.

La estructura de datos que acompañaría a las implementaciones Top-down es una lista enlazada donde la cabeza es el estado de memora del número N, cada nodo tiene un valor almacenado que representa el estado de memoria.

*1*

*3628800*

El llenado de memoria comienza desde el 0 hasta el 10.