ESTRUCTURA DE DATOS

Maria C. Torres M

Facultad de Minas

Departamento de Ciencias de la Computación y la Decisión

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Maria C. Torres

Ing. Electrónica (UNAL)
M.E. Ing. Eléctrica (UPRM)
Ph.D. Ciencias e Ingeniería de la Computación y la Información (UPRM)
Profesora Asociada – Departamento de Ciencias de la Computación y la Decisión

mctorresm@unal.edu.co

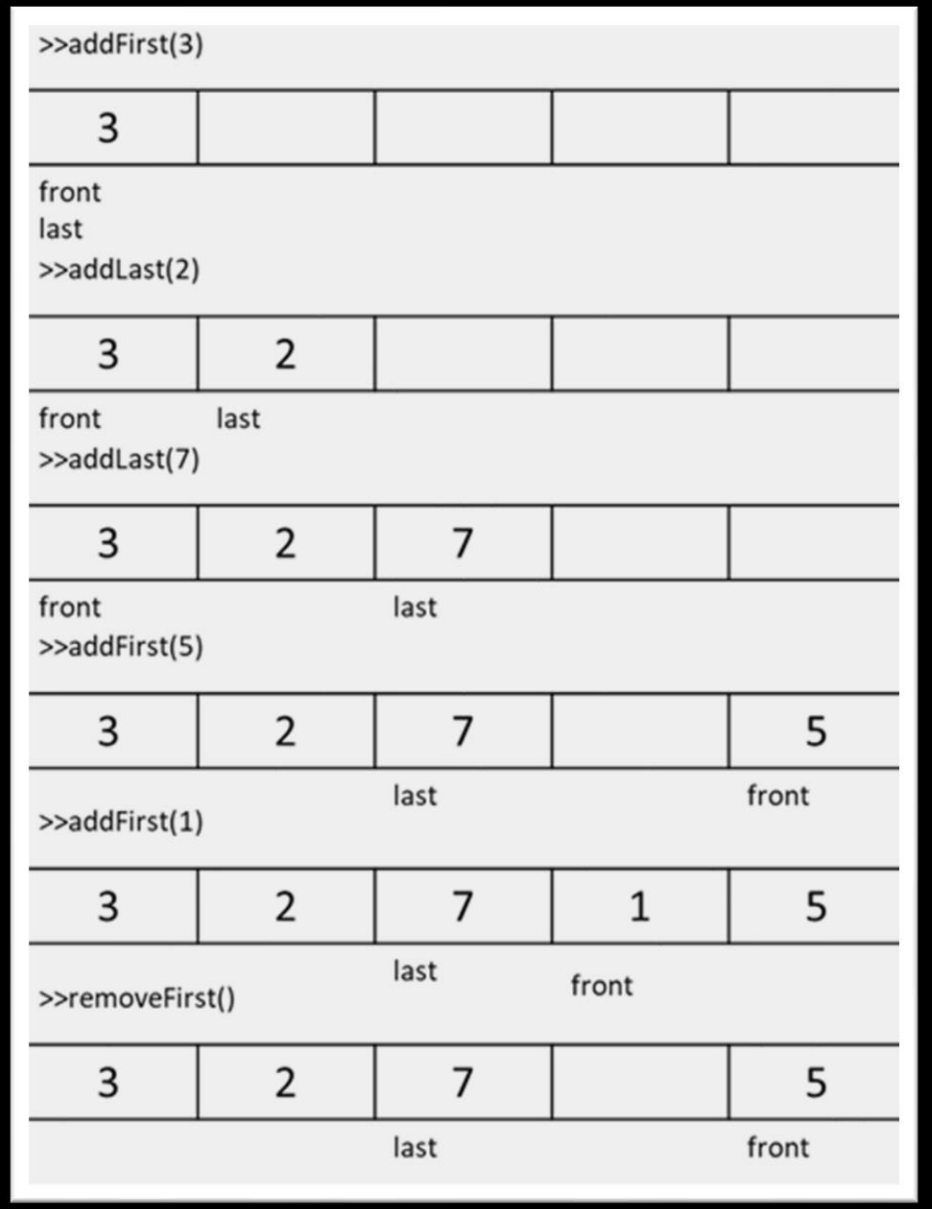
HORARIO DE ATENCIÓN: Martes -Viernes 10:00 am a 11:30 pm – Oficina 313 M8A

UNIVERSIDAD NACIONAL DE COLOMBIA

EXAMEN PARCIAL 2

Una cola de dos extremos (double-ended queue) se conoce como deque, permite agregar y eliminar un objeto en ambos extremos de la colección, es decir, permite las operaciones addFirst(Object e), addLast(Object e), removeFirst(), y removeLast(). Asuma que la estructura deque es implementada en un arreglo circular, conservando dos índices: front apunta al primer elemento de deque, y last apunta al último elemento de deque. Presente el algoritmo para addFirst(Object e).

Ejemplo: suponga un arreglo circular de longitud 5



UNIVERSIDAD NACIONAL DE COLOMBIA

EXAMEN PARCIAL 2

Queue
- front: int - last: int - size: int - data: Object[ ]
+ Queue(int capactity) + size():int + isEmpty():Boolean + addFirst(Object e) + addLast(Object e) + removeFirst(): Object + removeLast(): Object

```
addFirst(Object e)
if size()<data.length()
   // existe espacio disponible
   if first==0
      first = data.length-1
  else
     first = first-1
  data[first]=e
  size++
else
   print("No hay espacio disponible")
```

Tiempo de computo O(1)

UNIVERSIDAD NACIONAL DE COLOMBIA

EXAMEN PARCIAL 2

Una ArrayList emplea una lista simple para simular el comportamiento de un arreglo, pero preservando las ventajas de la memoria dinámica. Uno de los métodos que se puede incluir en el ArrayList es una getting que permita obtener los datos del nodo i. Presente un algoritmo que reciba una lista simple y un indice i; el método debe retornar el dato almacenado en el respectivo nodo i (getInx(List L, int i):Object).

EXAMEN PARCIAL 2

```
getInx(List L, int i):
 if i<L.size()
     temp = L.first()
     for (i=1,i<i,i++)
           temp = temp.getNext()
     return temp.getData()
 else
   return null
```

Tiempo de computo: O(n)

UNIVERSIDAD NACIONAL DE COLOMBIA

EXAMEN PARCIAL 2

Lista de canciones favoritas: suponga que se mantiene una colección de canciones favoritas en una lista doble. Cada canción además del nombre se almacena el numero de veces que es reproducida. Cada vez que una canción se reproduce aumenta el numero de reproducciones, y por tanto, puede cambiar de posición dentro de la lista de canciones favoritas. Presente un algoritmo que permita reproducir una canción identificada por su título, incrementar el número de reproducciones y reorganizar la lista doble. Para este problema no haga uso de algoritmos externos de ordenamiento (listaFavorita(DoubleList d, String titulo):Object).

Canciones
- titulo: String - song: Object - norep: int
+ Canciones(String t, Object s) + gettings… + settings..

UNIVERSIDAD NACIONAL DE COLOMBIA

EXAMEN PARCIAL 2

```
listaFavorita(DoubleList d, String titulo)
    //Primer paso: Buscar la canción
    temp = d.First()
    while(temp!=null & temp.getData().getTitulo()!= titulo)
              temp=temp.getNext()
    if temp==null
       return null  // No se encontró la canción
    else
       //Segundo paso: Se incrementa numero de reproducciones
       temp.getData().setNoRep(temp.getData().getNoRep()+1)
       //Tercer paso: se reorganiza la lista si es necesario
       actual = temp;
       anterior = temp.getPrev();
       while(anterior!=null & anterior.getData().getNoRep()<actual.getData().getNoRep())
            actual.setData(anterior.getData())
            anterior.setData(temp)
            actual = anterior
            anterior = actual.getPrev()
       return temp.getData()
```

Tiempo de computo: O(n)

UNIVERSIDAD NACIONAL DE COLOMBIA

EXAMEN PARCIAL 2

Una de las aplicaciones de un Stack es el balanceo de paréntesis. Dada una expresión matemática, suponga que se cuenta con un método que permite extraer de la expresión los símbolos de agrupamiento: (,),[,],{,} en un arreglo de caracteres S. Presente el algoritmo que permite determinar si una expresión con (,),[,],{,} se encuentra balanceada. Use los siguientes ejemplos para verificar su algoritmo (balanceo(S[]):Boolean):

S=( ) (( )) { ([( )]) } TRUE

S=) (( )) { ([( )]) } FALSE

S = ({ [ ]) } FALSE

UNIVERSIDAD NACIONAL DE COLOMBIA

EXAMEN PARCIAL 2

```
balanceo(S[])
    A = new Stack()
    for(i=0,i<S.length,i++)
        if (S[i]=='(' || S[i]=='{' || S[i]=='[')
           A.push(S[i])
        elseif S[i]==')' & !A.isEmpty() & A.top()=='('
           A.pop()
        elseif S[i]=='}' & !A.isEmpty() & A.top()=='{'
           A.pop()
        elseif S[i]==']' & !A.isEmpty() & A.top()=='['
           A.pop()
        else
            return FALSE
     if A.isEmpty
        return TRUE
     else
        return FALSE
```

Tiempo de computo: O(n)

UNIVERSIDAD NACIONAL DE COLOMBIA

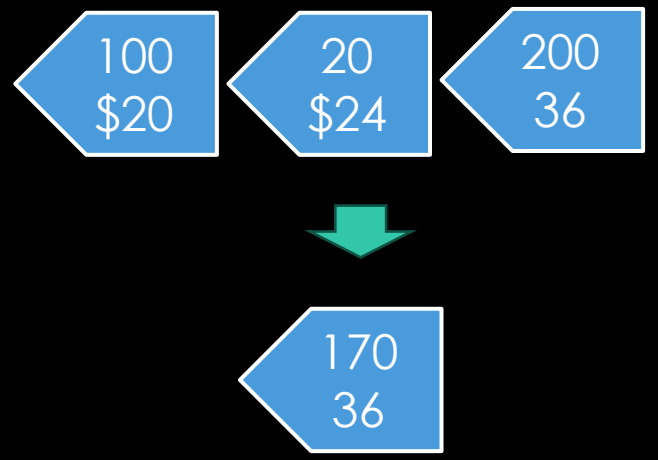
EXAMEN PARCIAL 2

Portafolio de inversión: cuando se vende una acción que hace parte de un portafolio de inversión, se calcula la ganancia o la perdida, la cual resulta de la diferencia entre el precio de venta y el precio inicialmente pagado por la acción cuando se compró. Esta regla es sencilla de aplicar cuando solo se venden acciones adquiridas al mismo tiempo; sin embargo, en los portafolios de inversión es común encontrar acciones de la misma compañía compradas en diferentes periodos de tiempo; para lo cual es necesario mantener para cada compra el precio de la acción. Una forma sencilla de calcular la ganancia es emplear un protocolo FIFO (first-to-in first-to-out) en el cual se venden las acciones que llevan mas tiempo en el portafolio. Por ejemplo, suponga que se compraron 100 acciones a $20 cada una el día 1, 20 acciones a $24 en el día 2, 200 acciones a $36 en el día 3, y se venden 150 acciones en el día 4 a $30 cada una. El protocolo FIFO indica que para vender las 150 acciones se venderá 100 del día 1, 20 del día 2, y 30 del día 3; por tanto la ganancia por la venta de las acciones será 100*10+20*6+30*(-6)=$940. Describa un algoritmo que dado una cola Q, donde se almacenan el número de acciones compradas y el valor de cada uno por día, el total de acciones a vender y el precio de venta de cada acción, retorne las ganancias producto de la venta (gananciaVenta(Queue Q, int numAcciones, int precioVenta):int).

UNIVERSIDAD
NACIONAL
DE COLOMBIA

EXAMEN PARCIAL 2

Por ejemplo, suponga que se compraron 100 acciones a $20 cada una el día 1, 20 acciones a $24 en el día 2, 200 acciones a $36 en el día 3, y se venden 150 acciones en el día 4 a $30 cada una. El protocolo FIFO indica que para vender las 150 acciones se venderá 100 del día 1, 20 del día 2, y 30 del día 3; por tanto la ganancia por la venta de las acciones será 100*10+20*6+30*(-6)=$940. Describa un algoritmo que dado una cola Q, donde se almacenan el número de acciones compradas y el valor de cada uno por día, el total de acciones a vender y el precio de venta de cada acción, retorne las ganancias producto de la venta (gananciaVenta(Queue Q, int numAcciones, int precioVenta):int).



UNIVERSIDAD NACIONAL DE COLOMBIA

EXAMEN PARCIAL 2

```
gananciaVenta(Queue Q, int numAcciones, int precioVenta)
      if !Q.isEmpty()
         ganancia = 0
         numTemp = numAcciones
         while(!Q.isEmpty() & Q.first().getNoAcciones()<numTemp)
                  temp = Q.dequeue()
                  ganancia + = temp.getNoAcciones()*(previoVenta-temp.getPrecio())
                  numTemp = numTemp – temp.getNoAcciones()
    if Q.isEmpty()
       print(“No se tenían suficientes acciones”)
    elseif numTemp>0
       temp = Q.dequeue()
       ganancia += numTemp*(previoVenta-temp.getPrecio())
       temp.setNoAcciones(temp.getNoAcciones()-numTemp())
       Q2 = new Queue()
       Q2.enqueue(temp)
       while(!Q.isEmpty())
            Q2.enqueue(Q.dequeue())
       Q = Q2;
   return ganancia
```

Tiempo de computo: O(n)

UNIVERSIDAD NACIONAL DE COLOMBIA