Cálculo II
Examen II

Rubén Pérez Palacios
Profesor: Dr. Fernando Nuñez Medina

30 Abril 2020

Problemas

1. Esturcturas de datos, selecciona la adecuadda.

   a) Cola, requieres de una escturctura donde puedas a acceder a ellos en la forma en que fueron agregados desde el primero hasta el último es decir $FIFO$ la definición de cola.

   b) Arreglo no ordenado, ya que permite el acceso aleatorio de sus elementos en $O(1)$-

   c) Pila, requieres de una escturctura donde puedas a acceder a ellos en la forma en que fueron agregados desde el último hasta el primero es decir $FILO$ la definición de pila.

   d) Set, ya que este como nos permitira acceder, borrar e insertar en $O(logN)$. De las opciones la mejor es la lista doblemente ligada ya que nos permite una manejo de memoria flexible, pero el acceder, agregar y eleminar sus elementos es en $O(N)$ por lo que sería lento.

   e) Lista doblemente ligada *dequeue*, esta te permite flexibilidad en el tamaño y memeria de los datos, y permite acceder e insertar al inicio y al final de esta.

2. Pila

   a) Creamos un arreglo del tamaño máximo de la pila, y guardamos la posición del último elemento insertado en el pila (o ínidicamos que no hay ninguno). Al insertar un elemento lo guardamos en la siguiente posición del úlitmo elemento insertado, y al eliminar borramos el último elemento insertado y para ambos actualizamos la posición del último elemento insertado.

   b) Creamos una lista simplemente ligada y una apuntador que indicara el primer elemento. Al agregar un elemento lo hacemos al final en nuestra lista, y al quitar solo movemos nuestro apuntador al nodo que apunta el nodo al que estabamos apuntando

   c) Creamos dos colas, una para guardar los elementos de la pila (digamos stack), y otra para efectuar (digamos tmp). Al agregar un elemento, lo agregamos a la cola tmp, y despues movemos uno por uno los elemntos de stack a tmp y por último regresamos todos los elementos a tmp; para quitar un elemento, moevemos todos los elementos de stack a tmp, hacemos pop en tmp y regresamos todos los elementos a stack.

3. Cola

   a) Creamos un arreglo del tamaño máximo de la pila, y guardamos la posición del último elemento insertado en el pila (o ínidicamos que no hay ninguno), a diferencia de la cola en el arreglo los iremos agregando de final a inicio. Al insertar un elemento lo guardamos en la anterior posición del úlitmo elemento insertado, y al eliminar borramos el último elemento insertado y para ambos actualizamos la posición del último elemento insertado.

   b) Creamos una lista simplemente ligada. Al agregar y al quitar un elemento lo hacemos al final en nuestra lista.

   c) Tenemos dos pilas. La principal, y la ayuda. En la principal guardamos la cola. Para agregar un numero, solo lo agregamos a la pila principal. Para quitar un numero, pasamos todos los numeros de la pila principal, a la de ayuda, exceptuando el ultimo numero. Y ahora regresamos los numeros de la pila principal a la de ayuda.

4. POO

   a) Código corregido

```
#include"iostream"
using namespace std;
class Vector3
{
        private :
                float x,y,z;
        public :
                Vector3 ( float x0 , float y0 , float z0 )
                {
                        x=x0;
                        y=y0;
                        z=z0;
                }
                void imprime ()
                {
                        cout<<x<<"␣"<<y<<"␣"<<z<<'$\setminus$n';
                }
                float getX(){ return x;}
                float getY(){ return y;}
                float getZ(){ return z;}
                float setX(float tmp){ return x = tmp;}
                float setY(float tmp){ return y = tmp;}
                float setZ(float tmp){ return z = tmp;}
};
Vector3* sumaVectores ( Vector3 *a, Vector3 *b)
{
        return *ret = new Vector3(a->getX()+b->getX(),a->getY()+b->getY
            (),a->getZ()+b->getZ()) ;
}

int main () {
        Vector3 v1(1 ,2 ,3) ;
        Vector3 v2(4 ,5 ,6) ;
        Vector3 *suma=sumaVectores (&v1,&v2) ;
        suma->imprime();
        delete suma;
        return 0;
}
```

b) Separar la definicion de la clase en una librería aparte y sobrecargar el operador +.

5. Heap

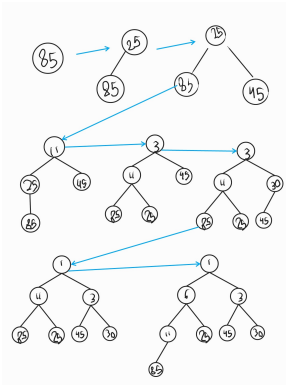

Figura 1: Construcción Árbol

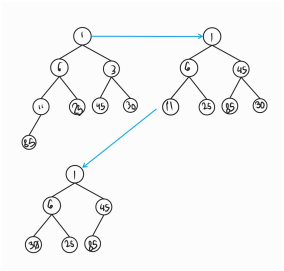

Figura 2: Eliminación de nodos

6. Algoritmos de Ordenamiento

a) Es cuando un conjunto de datos no importa el orden de estos en la entrada siempre tomara el mismo número de accciones ordenarlo.
b) InsertionSort,MergeSort,Heap-sort
c) MergeSort,Heap-Sort
d) InsertionSort,Heap-Sort
e) MergeSort,QuickSort

7. Tipos de Algoritmos de Ordenamiento

a) Merge sort
b) Selection sort
c) Quicksort
d) Insertion sort
e) Heapsort