ESTRUCTURA DE DATOS 1
Código ST0245

Laboratorio Nro. X
Escribir el tema del laboratorio

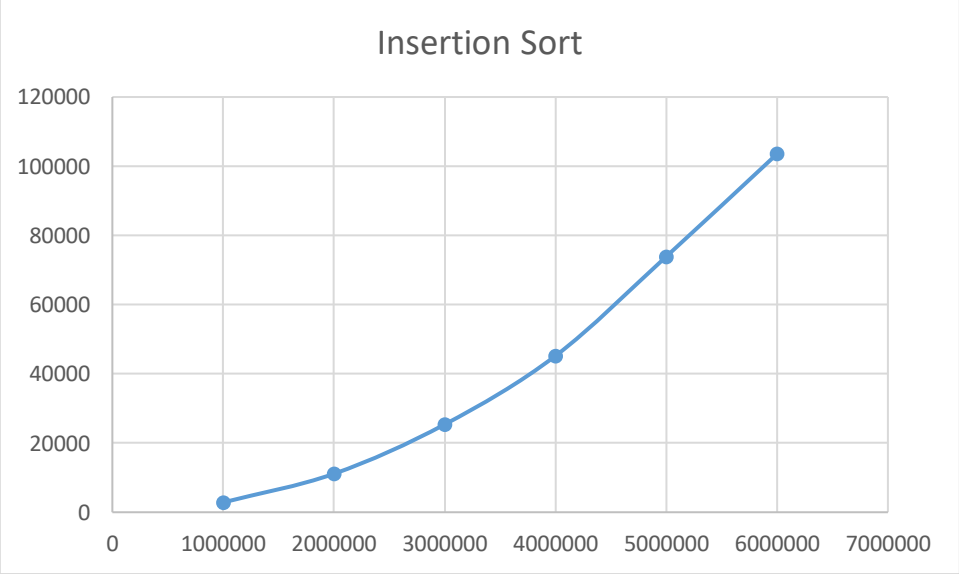
Juan Andres Henao Diaz
Universidad Eafit
Medellín, Colombia
jahenaod@eafit.edu.co

Carlos Andres Mosquera
Universidad Eafit
Medellín, Colombia
camosquerp@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

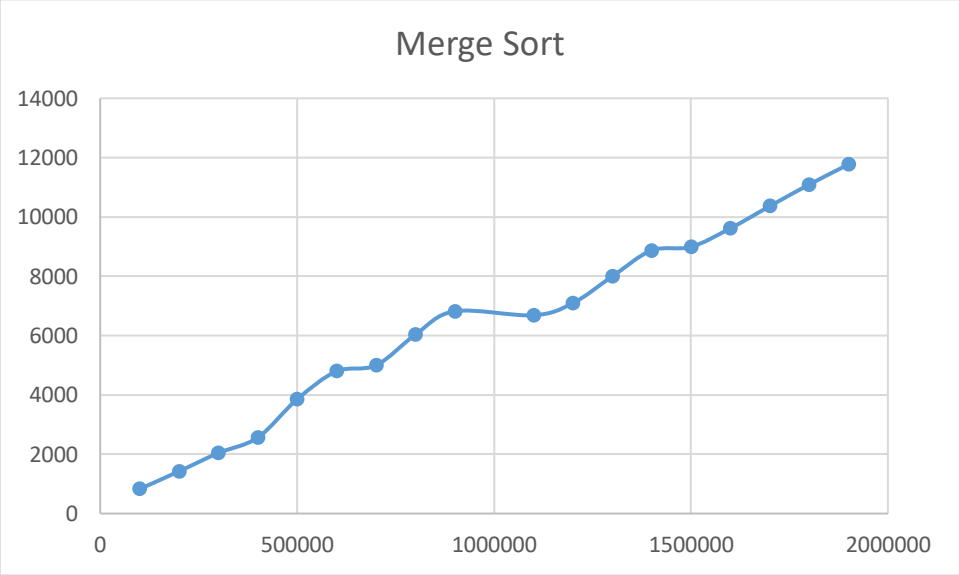
3.2 Grafiquen los tiempos que tomó en cada algoritmo para los 20 tamaños del problema. Grafiquen el Tamaño de la Entrada Vs. Tiempo de Ejecución

Insertion Sort



PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Merge sort



3.3 ¿Es apropiado usar insertion sort para un videojuego con millones de elementos en una escena y demandas de tiempo real en la renderización?

- El algoritmo insertion sort es un método de ordenamiento bastante eficiente y simple de hacer para los programadores principiantes pero no significa que sea el mejor para todos los casos de la vida cotidiana. Para pequeños conjuntos de datos, el algoritmo es bastante eficiente y rápido en su procesamiento, pero a los grandes conjuntos de datos este se hace bastante demorado y no es eficiente para la renderización de un videojuego. En conclusión, en grandes elementos el algoritmo tiende a tardarse mucho ya que su complejidad es O(n^2)
-

3.4 ¿Por qué aparece un logaritmo en la complejidad asintótica, para el peor de los casos, de merge sort o insertion sort?

- para el peor de los casos, el método de ordenamiento merge sort es mucho mejor que insertion sort ya que el primero mencionado tiene una complejidad de O(nlog(n)). Esto se debe a que ordena gracias a que divide el arreglo en dos mitades y así hasta ordenar todos los datos, lo que lo hace bastante eficiente al enfrentarse con millones de datos, algo mas acercado a la vida cotidiana

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

3.5 Calculen la complejidad de los Ejercicios en Línea de los numerales 2.1 y agréguenla al informe PDF

- **Array 2**

- **CountEvens:**

```
public int countEvens(int[] nums) {
    int count = 0;          // C1
    for (int n : nums)      // n
        if (n % 2 == 0)     // n
            count++;        // n
    return count;           //C2

}
```

Complejidad O(n)

- **Only14:**

```
public boolean only14(int[] nums) {
    for(int i=0;i<nums.length;i++)       //n
        if (nums[i] != 1 && nums[i]!= 4)    //n
            return false;                    //n
    return true;                          //c

}
```

complejidad O(n)

- **CentredAverage:**

```
public int centeredAverage(int[] nums) {
    int min = nums[0];  //C1
    int max = nums[0];  //C2
    int sum = 0;

    for (int i = 0; i < nums.length; i++){ //C3
        sum += nums[i];                  //C4*n
        min = Math.min(min,nums[i]);       //C5*n
        max = Math.max(max,nums[i]);       //C6*n
    }
    sum = sum - max - min;                //C8
    sum = sum / (nums.length-2);
    return sum;

}
```

//Complejidad de O(n^2)

- **Lucky13:**

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
public boolean lucky13(int[] nums) {
    boolean result = true;              //C1
    for (int i =0;i<nums.length ;i++)       //C2
        if ( nums[i] == 1 || nums[i] == 3)      //C3*n
            result = false;
    return result;                          //C4
}
```

//Complejidad de O(n)

- **BigDiff:**

```
public int bigDiff(int[] nums) {
    int cont1=nums[0]; //c1
    int cont2=nums[0]; //c2
    for(int i=0;i<nums.length;i++){ //n
        if(nums[i]>=cont1)
            cont1=nums[i]; //c3*n
        if(nums[i]<=cont2)
            cont2=nums[i]; //c4*n
    }
    return cont1-cont2; //c5
}
```

Complejidad O(n)

- **Array 3:**

- **MaxSpan**

```
public int maxSpan(int[] nums) {
    int max = 0; //C
    for (int i = 0; i < nums.length; i++) {//n*C
        int j = nums.length - 1; //n*C
        for (j = nums.length - 1; nums[i] != nums[j]; j--) { //n*m*C
            if (nums[j] == nums[i]) { //n*m*C
                break; //n*m*C
            } }
        int span = j - i + 1; //n*C
        if (span > max) { //n*C
            max = span; //n*C
        }
    }
    return max; //C
```

//Complejidad O(n*m)

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
}
```

- **CanBalance**

```
public boolean canBalance(int[] nums) {
    int suma1=0; //c1
    for (int i = 0; i < nums.length; i++){ //c2*n
        suma1=suma1+ nums[j]; //c3*n
        int suma2=0; //c5
        for (int j = nums.length
            -1; j > i; j--) { //c6*n*
            m
            suma2=suma2+ nums[j]; //c7*n

        }
        if(suma1==suma2){ //C8
            return true;

        }

    }
    return false;
}
```

Complejidad de O(n*m)

- **SeriesUp**

```
public int[] seriesUp(int n) {
    int[] arr = new int[n * (n + 1) / 2];
    int cont = 0;
    for (int i=1; i<=n;i++){ //c1*n
        for (int j=1;j<=i;j++){ //c2*n*m
            arr[cont++]=j;
        }
    }
    return arr;
}
```

//Complejidad O(n*m)

- **Fix34**

```
public int[] fix34(int[] nums) {
    for (int i = 0; i < nums.length; i++) //c1*n
        if (nums[i] == 3) {
            int temp = nums[i + 1];
            nums[i + 1] = 4;
```

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

```
        for (int j = i + 2; j < nums.length; j++) //c2*n*
            m
        if (nums[j] == 4)
            nums[j] = temp; //c3*n

    }
  return nums; //c4
  //Complejidad O(n*m)
}
```

- **LinearIn:**

```
public boolean linearIn(int[] outer, int[] inner) {
  int pos=0; //c1
  for(int i=0;i<outer.length;i++){ //c2
    *
    n
    if(pos<inner.length) //c3
      *
    n
    if(outer[i]==inner[pos]){ //c4
      *
      n*m
      pos++; //c5
      *
      n

    }

  }
  if(pos==inner.length){
    //c6
    return true;

  }
  else {
    return false;

  }
  //Complejidad O(n*m)
```

3.6

N = N representa el tamaño del arreglo de cada problema
M = M representa el ciclo, se repite en función de N

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

4) Simulacro de Parcial

4.1 *C*
4.2 *B*
4.3 *B*
4.4 *B*
4.5 D , A
4.6 10 segs
4.7 A, B y C
4.8 A
4.9 A
4.10 C
4.11 C
4.12 B
4.13 C
4.14 C

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473