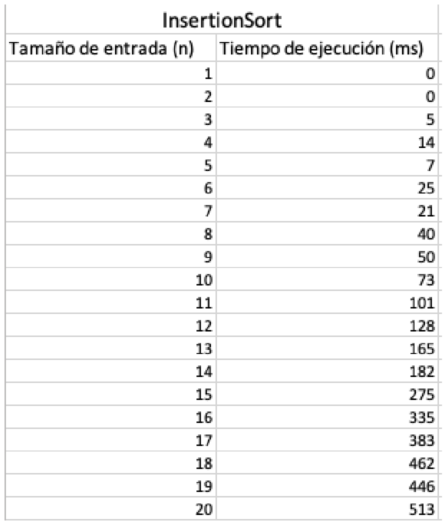
Laboratory practice No. 2: Algorithm complexity

María Paulina Ocampo Duque
Universidad Eafit
Medellín, Colombia
mpocampod@eafit.edu.co

María José Gutiérrez Estrada
Universidad Eafit
Medellín, Colombia
mjgutierre@eafit.edu.co

3) Practice for final project defense presentation

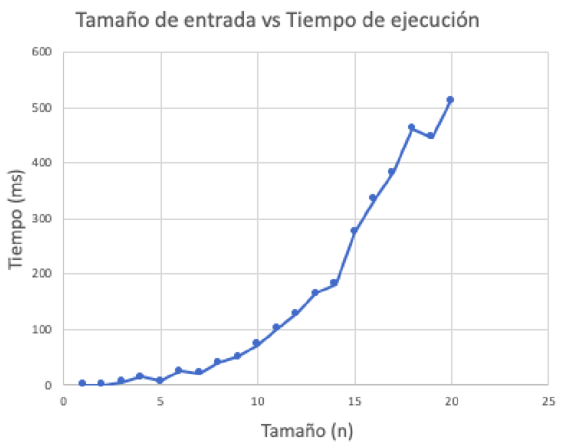
3.1



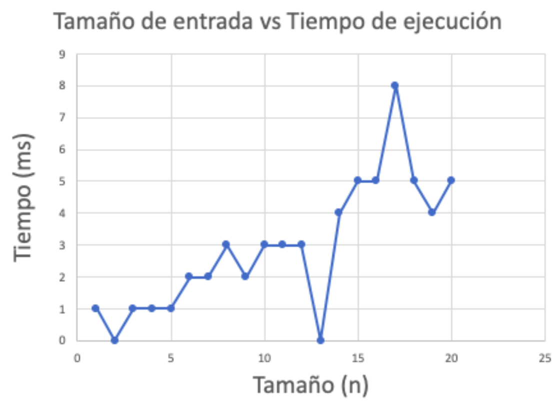
InsertionSort	
Tamaño de entrada (n)	Tiempo de ejecución (ms)
1	0
2	0
3	5
4	14
5	7
6	25
7	21
8	40
9	50
10	73
11	101
12	128
13	165
14	182
15	275
16	335
17	383
18	462
19	446
20	513

MergeSort	
Tamaño de entrada (n)	Tiempo de ejecución (ms)
1	1
2	0
3	1
4	1
5	1
6	2
7	2
8	3
9	2
10	3
11	3
12	3
13	0
14	4
15	5
16	5
17	8
18	5
19	4
20	5

3.2 Insertion sort



Merge sort



PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

3.3 It´s not appropriate to use Insertion sort code for a videogame with millions of elements in the scene because their complexity O(n^2) is going to double in every single value. So it would take many time to execute the program, making it less efficient.

3.4 Merge sort is the one with the logarithmic complexity in the worst case, it appears because while the data increase, the operations almost all the time increase too but not in a proporcional way, making it complexity faster than others sorting algorithms.

3.5 When is insertion sort faster than merge sort?
When the array is already organized the algorithm insertion sort should be faster because it checks the elements once.

3.6 Array2

- **countEvens**
```
public int countEvens(int[ ] nums) {
        int count = 0;          // C1
        for (int n : nums)      // C2
            if (n % 2 == 0)     // n
                count++;        // n
        return count;           //C3*n
    }
```
  Complexity O(n)

- **Only14**
```
public boolean only14(int[] nums) {
      for(int i=0;i<nums.length;i++)       //C1
       if (nums[i] != 1 && nums[i]!= 4)    //n
          return false;                    //n
       return true;                        //c
}
```
  Complexity O (n)

- **More14**
```
public boolean more14(int[] nums) {
    int cont1=0;                              //C1
    int cont2=0;                              //C2
    for(int i=0 ; i<nums.length ; i++){       //n
      if(nums[i]==1)
        cont1++;                              //c3*n
      if(nums[i]==4)
        cont2++;                              //c4*n
    }
     if(cont1>cont2)                          //c5
     return true; return false;               //c6
```
  Complexity O(n)

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

- **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
}
```

Complexity O(n)

- **fizzArray3**

```
public int[ ] fizzArray3(int start, int end) {
  int[] array = new int[end - start];             //c1

   for(int i = start; i < end; i++)
        array[i - start] = i;                     //c2*n

       return array;                              //c3
}
```

Complexity O(n)

Array3

- **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
```

Complexity O(n*m)

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

- **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;
           for (int j = i + 2; j < nums.length; j++) //c2*n*m
                if (nums[j] == 4)
                 nums[j] = temp;                        //c3*n
        }
    return nums;                                       //c4
}
Complexity O(n*m)
```

- **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;
}
Complexity 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;
 }
```

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

```
    }
```

Complexity 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;
}
```

Complexity O(n*m)

3.7

- **countEvens= nums** represents the size of the array
- **Only14= nums** represents the size of the array
- **More14= nums** represents the size of the array
- **bigDiff= nums** represents the size of the array
- **fizzArray3= array** represents the size of the array
- **maxSpan= nums** represents the size of the array and **m** represents a cycle, it repeats the value created by nums
- **fix34= nums** represents the size of the array and **m** represents a cycle, it repeats the value created by nums
- **canBalance= nums** respresents de size of the array and **m** represents a cycle, it repeats the value created by nums
- **linearIn=inner** is the first array and **outer** is the second array
- **seriesUp= n** represents de size of the array and **m** represents a cycle, it repeats the value created by n

4) Practice for midterms

4.2 b
4.5 1 d
2 a
4.6 The algorithm in 100s processes 10,000 data
4.7 1) O(f+g)=O(max(f,g))
2) O(f×g)=O(f)×O(g)
4) *O*(*c*.*f*)=*O*(*f*), where *c is a constant*
4.8 a
4.9 a
4.10 d
4.11 c

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

4.12 b
4.13 c
4.14 a

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473