

IE-0217 Estructuras abstractas de datos y algoritmos para ingeniería.

Lab 03: Tiempos y Ordenes de Duración

Belinda Brown Ramírez - B61254

I-2019

Tabla de contenidos

1. Enunciado:	3
1.1. Consideraciones	3
2. Solución:	4
2.1. Pseudocódigo:	4
2.1.1. Selección Sort:	4
2.1.2. Binary Search:	5
2.2. Tiempos de ejecución:	6
2.2.1. Python	6
2.2.2. C++	7
2.3. Gráficos:	8
2.3.1. Selection Sort	8
2.3.2. Binary Search	9
2.4. Conclusiones:	9
Referencias	10
3. Anexos:	11
3.1. Código Python:	11
3.1.1. Selection Sort:	11
3.1.2. Binary Search	12
3.2. Código C++:	14
3.2.1. Selection Sort:	14
3.2.2. Binary Search:	15

1. Enunciado:

1. Escriba en pseudocódigo los algoritmos de Selection sort y Binary search.
2. Calcule la función de tiempo de ejecución $T(N)$ y el orden de duración (complejidad temporal) $O(T(N))$ de dichos algoritmos.
3. Programe ambos algoritmos en Python y C++ de tal forma que reciban por línea de comandos la ruta de un archivo que contenga una lista de números a operar. Dicho archivo estará compuesto de dos líneas, la primera será la cantidad de números y la segunda, dichos números.
4. Averigüe como tomar tiempos de ejecución tanto en Python como en C++.
5. Mida cuanto tardan los algoritmos del punto 1. No tome en cuenta el tiempo de levantado del archivo en memoria ni la impresión del resultado.
6. Tome los tiempos de ejecución de ambos algoritmos, al menos 3 veces, para listas aleatorias de largo $N = \{ 10^1, 10^2, 10^3, 10^4, 10^5, 10^6 \}$.
7. Verifique que los tiempos medidos sean acordes con los tiempos y ordenes calculados. Grafique sus resultados.

1.1. Consideraciones

1. Equipos de 2 o 3 personas.
2. Genere un reporte en \LaTeX que incluya el enunciado, sus conclusiones y, como anexo incluya su código fuente
3. Suba su código y documentación (doxygen, README, INSTALL) al GitLab respectivo de su grupo y el directorio del laboratorio.
4. Cada estudiante debe subir el reporte a Schoology.
5. Recuerde que por cada día tardío de entrega se le rebajaran puntos de acuerdo con la formula: 3^d donde $d > 1$ es la cantidad de días tardíos

2. Solución:

2.1. Pseudocódigo:

2.1.1. Selecion Sort:

Pseudocodigos.txt

```
SelectionSort
{
    int i,j;
    int menor = i;
    array[n];

    for i = 1; i < n-1 ; i++
    {
        for j = i+1; j < n; j++
        {
            if array[j] < array[menor]
            {
                menor = j;
            }

        }

        ifmenor != i
        {
            swap array[i] and array[menor];
        }
    }
}

// n, número de entradas del array
```

2.1.2. Binary Search:

Pseudocodigos.txt

```
BinarySearch
{
    int n;
    int array[n];
    int objetivo;

    int min = 0;
    int max = n-1;
    int promedio = min+max/2

    whilemin<=max
    {

        ifarray[promedio] == objetivo
        {
            print "Fin";
        }
        else ifarray[promedio] < objetivo
        {
            min = promedio +1;
        }
        else ifarray[promedio] > objetivo
        {
            max = promedio - 1;
        }else{
            Print "Numero no está"
        }

    }

}

//n, número de elementos del arreglo
//array está ordenado
```

2.2. Tiempos de ejecución:

Teóricamente, el tiempo de ejecución para Binary Search viene dado por:

$$\log_2(n) + 1 \quad (1)$$

Además, para Selection Sort:

$$c(n) = \frac{n^2 - n}{2} \quad (2)$$

2.2.1. Python

```
caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 BS.py
Digite nombre del archivo (con su .txt): archivo.txt
Cantidad de datos 10
Objetivo: 75
Objetivo no está
Iteraciones: 1
Tiempo de ejecución: 1.3113021850585938e-05
caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 BS.py
Digite nombre del archivo (con su .txt): a1.txt
Cantidad de datos 100
Objetivo: 75
Objetivo no está
Iteraciones: 1
Tiempo de ejecución: 1.4066696166992188e-05
caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 BS.py
Digite nombre del archivo (con su .txt): a2.txt
Cantidad de datos 1000
Objetivo: 75
Iteraciones: 9
Tiempo de ejecución: 1.5497207641601562e-05
caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 BS.py
Digite nombre del archivo (con su .txt): a3.txt
Cantidad de datos 10000
Objetivo: 75
Objetivo no está
Iteraciones: 11
Tiempo de ejecución: 4.00543212890625e-05
caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 BS.py
Digite nombre del archivo (con su .txt): a5.txt
Cantidad de datos 40000
Objetivo: 75
Objetivo no está
Iteraciones: 11
Tiempo de ejecución: 0.00013399124145507812

caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 SS.py
Digite nombre del archivo (con su .txt): archivo.txt
Cantidad de datos: 10
Tiempo de ejecución: 4.267692565917969e-05
caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 SS.py
Digite nombre del archivo (con su .txt): a1.txt
Cantidad de datos: 100
Tiempo de ejecución: 0.002638578414916992
caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 SS.py
Digite nombre del archivo (con su .txt): a2.txt
Cantidad de datos: 1000
Tiempo de ejecución: 0.22375035285949707
caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 SS.py
Digite nombre del archivo (con su .txt): a3.txt
Cantidad de datos: 10000
Tiempo de ejecución: 21.188767910003662
caro@deathstar:~/Desktop/Estructuras/Lab03$ python3 SS.py
Digite nombre del archivo (con su .txt): a5.txt
Cantidad de datos: 40000
Tiempo de ejecución: 337.90108942985535
```

2.2.2. C++

```
rvenegas@debian:~/Escritorio/Lab03$ g++ BS.cpp -o a.out
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo1.txt
Cant datos: 10
objetivo: 100000
Numero esta en la posición: 9
Tiempo de ejecución : 1.3e-05 s
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo2.txt
Cant datos: 100
objetivo: 100000
Numero esta en la posición: 99
Tiempo de ejecución : 2.6e-05 s
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo3.txt
Cant datos: 1000
objetivo: 100000
Numero esta en la posición: 999
Tiempo de ejecución : 3e-06 s
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo4.txt
Cant datos: 10000
objetivo: 100000
Numero esta en la posición: 9999
Tiempo de ejecución : 3e-06 s
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo5.txt
Cant datos: 100000
objetivo: 100000
Numero esta en la posición: 99999
Tiempo de ejecución : 3e-06 s
rvenegas@debian:~/Escritorio/Lab03$ █
```

```
rvenegas@debian:~/Escritorio/Lab03$ g++ SS.cpp -o a.out
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo1.txt
Cant. datos: 10
Tiempo de ejecución : 6.4e-05 s
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo2.txt
Cant. datos: 100
Tiempo de ejecución : 0.000154 s
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo3.txt
Cant. datos: 1000
Tiempo de ejecución : 0.002631 s
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo4.txt
Cant. datos: 10000
Tiempo de ejecución : 0.167669 s
rvenegas@debian:~/Escritorio/Lab03$ ./a.out archivo5.txt
Cant. datos: 100000
Tiempo de ejecución : 16.6956 s
rvenegas@debian:~/Escritorio/Lab03$ █
```

Orden de duración (complejidad temporal) para ambos algoritmos, Binary Search viene dado por:

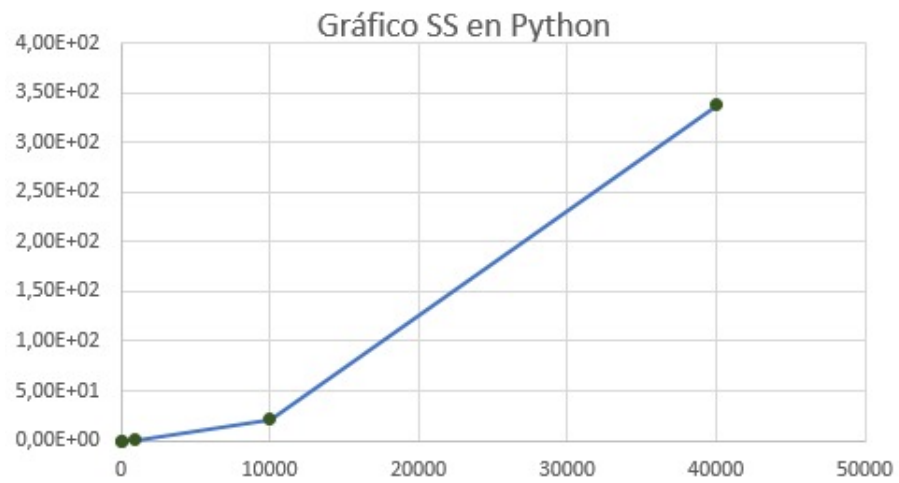
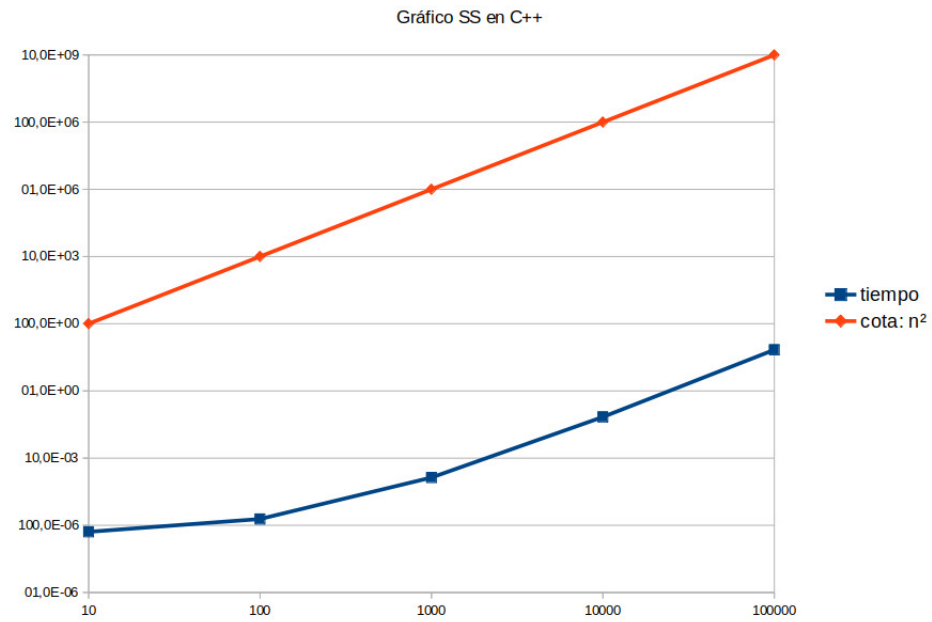
$$O(k + \log_{10}(n)) \quad (3)$$

Para Selection Sort:

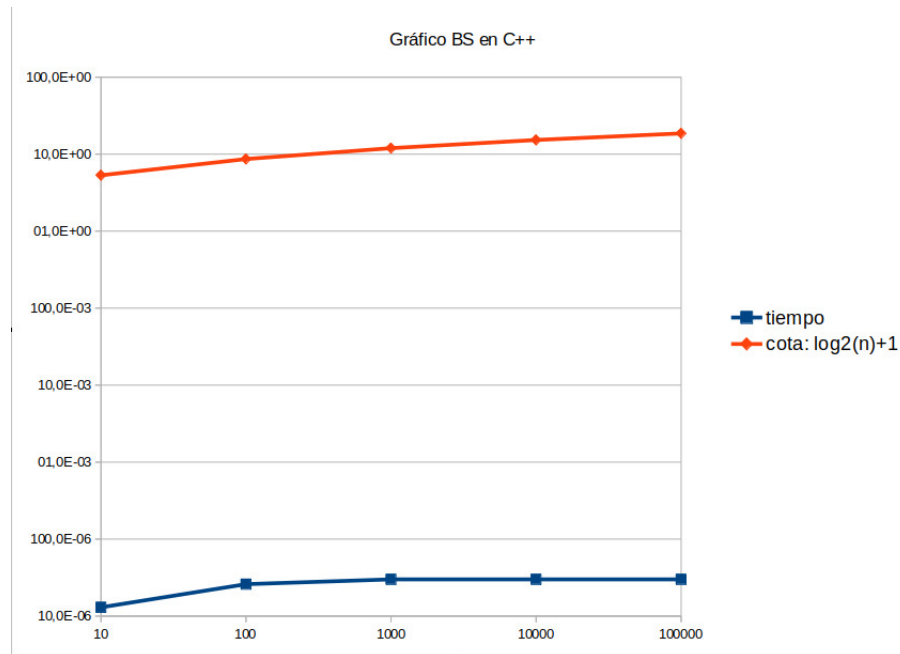
$$O((n)) = n^2 \quad (4)$$

2.3. Gráficos:

2.3.1. Selection Sort



2.3.2. Binary Search



2.4. Conclusiones:

Gracias a las gráficas, podemos observar como varían los tiempos de ejecución, al utilizar C++ y Python. En cuanto a las diferencias, se pueden deber a varios aspectos, como el uso de diferentes computadoras para realizar las pruebas y el que Python es un lenguaje interpretado, y por ende, más lento en comparación con C++. También, los tiempos de ejecución pueden verse afectados por la cantidad de procesos que este realizando la computadora a la hora de correr el código y tomar el tiempo.

Además, se observa como para el código de Selecion Sort, tanto en Python como C++ los comportamientos se parecen y se respeta la cota superior. Para el código de Binary Search, en C++, el comportamiento se parece al esperado teóricamente pero no en Python.

Referencias

- [1] Wikipedia.(2019) *Búsqueda Binaria*
<https://es.m.wikipedia.org/wiki/Busquedabinaria>
- [2] Stack Overflow(2018) *Varios*
<https://es.stackoverflow.com/>
- [3] EcuRed(2018) *Lenguaje de programación Python*
<https://www.ecured.cu/>

3. Anexos:

3.1. Código Python:

3.1.1. Selection Sort:

```
1 from time import time
2 import re
3 import numpy
4 import os
5 import sys
6 import string
7
8 def bs (objetivo):
9
10
11     fname = input("Digite nombre del archivo (con su .txt): ")
12
13     #lectura de la primera linea
14     f = open(fname, 'r')
15     l = f.readline()
16     nuevool = int(l)
17
18     #Lectura de la segunda linea, se obtiene una lista
19     for index, line in enumerate(open(fname, 'r').readlines()):
20         w = line.split(' ')
21         ll = w[0:nuevol]
22
23     #Se convierte cada entrada de la lista en numeros en un array
24     array = [int(i) for i in ll]
25
26     #Se ordenan los n meros del array
27     array.sort()
28
29     print("Cantidad de datos", nuevool)
30     print("Objetivo:", objetivo)
31
32     tiempoinicial = time()
33
34     min = 0
35     max = len(array)-1
36     iteracion = 0
37     fin = False
38     while min <= max and not fin:
39         iteracion = iteracion + 1
40         promedio = (min+max)//2
41
42         if array[promedio] == objetivo:
43             print("Objetivo encontrado, igual al promedio")
44             fin = True
45
46         if array[promedio] < objetivo:
47             #print("Objetivo no encontrado, buscando m s a la derecha")
48             min = promedio + 1
49             #promedio = (min+max)//2
50             fin=False
51
52         if array[promedio] > objetivo:
```

```

53         #print("Objetivo no encontrado, buscando m s a la izquierda")
54         max = promedio -1
55         fin=False
56         #promedio = (min+max)//2
57
58     else:
59         print("Objetivo no est ")
60         fin = True
61
62
63     tiempofinal = time()
64     tiempototal = tiempofinal - tiempoinicial
65
66     print("Iteraciones:", iteracion)
67     print("Tiempo de ejecuci n:", tiempototal)
68     return fin
69
70
71
72
73 objetivo = 75
74
75 bs(objetivo)

```

3.1.2. Binary Search

```

1 import os, sys
2 import stat
3 import numpy as np
4 from time import time
5
6 def ss():
7
8     fname = input("Digite nombre del archivo (con su .txt): ")
9
10    #lectura de la primera linea
11    f = open(fname, 'r')
12    l = f.readline()
13    nuevool = int(l)
14
15    #print(nuevol)
16
17    #Lectura de la segunda linea, se obtiene una lista
18    for index, line in enumerate(open(fname, 'r').readlines()):
19        w = line.split(' ')
20        l1 = w[0:nuevol]
21
22    #Se convierte cada entrada de la lista en numeros en un array
23    array = [int(i) for i in l1]
24
25    print("Cantidad de datos:", nuevool)
26
27    tiempoinicial = time()
28
29    #Procedimiento del SS
30    for i in range(nuevol):
31        menor = i
32
33        for j in range(i+1, nuevool):

```

```
34         if array[j] < array[menor]:
35             menor = j
36
37         swap = array[i]
38         array[i] = array[menor]
39         array[menor] = swap
40
41     tiempofinal = time()
42     tiempototal = tiempofinal - tiempoinicial
43
44
45     #print("Nuevo arreglo ordenado:", array)
46     print("Tiempo de ejecuci n:" ,tiempototal)
47
48     return array
49
50
51
52 ss()
```

3.2. Código C++:

3.2.1. Selection Sort:

```
1 #include<iostream>
2 #include <fstream>
3 #include <cstring>
4 #include <sstream>
5 #include <string>
6 #include <ctime>
7
8 using namespace std;
9
10 void ss(string filePath)
11 {
12
13     ifstream archivo;
14     archivo.open(filePath , ios::in);
15     string texto;
16     int linea =0;
17     int max;
18
19     while(!archivo.eof()){
20         getline(archivo , texto);
21         istringstream isstream(texto);
22         while(!isstream.eof()){ //divide cada linea en palabras
23             int tempStr;
24             isstream >> tempStr;
25             max=tempStr;
26         }
27         break;
28     }
29     archivo.close();
30
31     ifstream archivo2;
32     archivo2.open(filePath , ios::in);
33     string texto2;
34     int array[max] ={};
35     linea =0;
36     while(!archivo2.eof()){
37         getline(archivo2 , texto2);
38         istringstream isstream(texto2);
39         if (linea==1) {
40             int iter=0;
41             while(!isstream.eof()){ //divide cada linea en palabras
42                 if (iter<max){
43                     int tempStr;
44                     isstream >> tempStr;
45                     array[iter]=tempStr;
46                     iter=iter+1;
47                 }else{
48                     break;
49                 }
50             }
51         }
52         linea=1;
53     }
54     archivo2.close();
55 }
```

```

56 //Comienza el algoritmo de selection sort
57 unsigned t0, t1;
58
59 t0=clock();
60 int min;
61 int aux;
62 for (int i=0; i<max-1; i++){
63     min=i;
64     for(int j=i+1; j<max; j++){
65         if (array[min]>array[j]){
66             min=j;
67         }
68     }
69     aux=array[i];
70     array[i]=array[min];
71     array[min]=aux;
72 }
73 cout<<"Cant. datos: "<<max<<endl;
74 t1 = clock();
75
76 double time = (double(t1-t0)/CLOCKS_PER_SEC);
77 cout << "Tiempo de ejecuci n : " << time << " s"<< endl;
78
79 return;
80 }
81
82
83
84 int main(int argc , char* argv[]){
85
86 ss(argv[1]);
87
88
89 return 0;
90
91 }

```

3.2.2. Binary Search:

Funciones.h:

```

1 #include<iostream>
2 #include <fstream>
3 #include <cstring>
4 #include <sstream>
5 #include <string>
6 #include <ctime>
7
8 using namespace std;
9
10 void bs(string filePath, int objetivo)
11 {
12
13     ifstream archivo;
14     archivo.open(filePath, ios::in);
15     string texto;
16     int linea =0;
17     int max;
18     while(!archivo.eof()){

```

```

19     getline(archivo , texto);
20     istringstream isstream(texto);
21     while(!isstream.eof()){ //divide cada linea en palabras
22         int tempStr;
23         isstream >> tempStr;
24         max=tempStr;
25     }
26     break;
27 }
28 archivo.close();
29 ifstream archivo2;
30 archivo2.open(filePath , ios::in);
31 string texto2;
32 int array[max]={};
33 linea =0;
34 while(!archivo2.eof()){
35     getline(archivo2 , texto2);
36     istringstream isstream(texto2);
37     if (linea==1) {
38         int iter=0;
39         while(!isstream.eof()){ //divide cada linea en palabras
40             if (iter<max){
41                 int tempStr;
42                 isstream >> tempStr;
43                 array[iter]=tempStr;
44                 iter=iter+1;
45             }else{
46                 break;
47             }
48         }
49     }
50     linea=1;
51 }
52 archivo2.close();
53
54 //Codigo selection sort para ordenar el arreglo
55 int min;
56 int aux;
57 for (int i=0; i<max-1; i++ ){
58     min=i;
59     for(int j=i+1; j<max; j++){
60         if (array[min]>array[j]){
61             min=j;
62         }
63     }
64     aux=array[i];
65     array[i]=array[min];
66     array[min]=aux;
67 }
68
69
70
71 cout<<"Cant datos: "<<max<<endl;
72 cout<<"objetivo: "<<objetivo<<endl;
73 min=0;
74 int result=0;
75 bool cond =false;
76
77 //Algoritmo binary search

```

```

78
79     unsigned t0, t1;
80     t0=clock();
81
82     while (min<=max)
83     {
84         int promedio = (min + max)/2;
85
86         if (array[promedio]==objetivo){
87             result=promedio;
88             cond=true;
89             break;
90         }
91         if (array[promedio]<objetivo){
92             min=promedio+1;
93             promedio=(min+max)/2;
94         }
95         if (array[promedio]>objetivo){
96             max=promedio-1;
97             promedio=(min+max)/2;
98         }
99     }
100     if (cond==false){
101         cout<<"Numero no esta en el arreglo"<<endl;
102     }
103     else{
104         cout<<"Numero esta en la posici n: "<< result<< endl;
105     }
106     }
107     t1 = clock();
108
109     double time = (double(t1-t0)/CLOCKS_PER_SEC);
110     cout << "Tiempo de ejecuci n : " << time << " s"<< endl;
111
112     return;
113 }
114
115
116
117
118 int main(int argc , char* argv[]){
119
120
121
122     int objetivo = 100000;
123     bs(argv[1], objetivo);
124
125
126
127
128
129     return 0;
130
131 }

```