Análisis de Algoritmos

Ejercicio 06: "Análisis de algoritmos no recursivos"

Nombre: Luis Fernando Ramírez Cotonieto

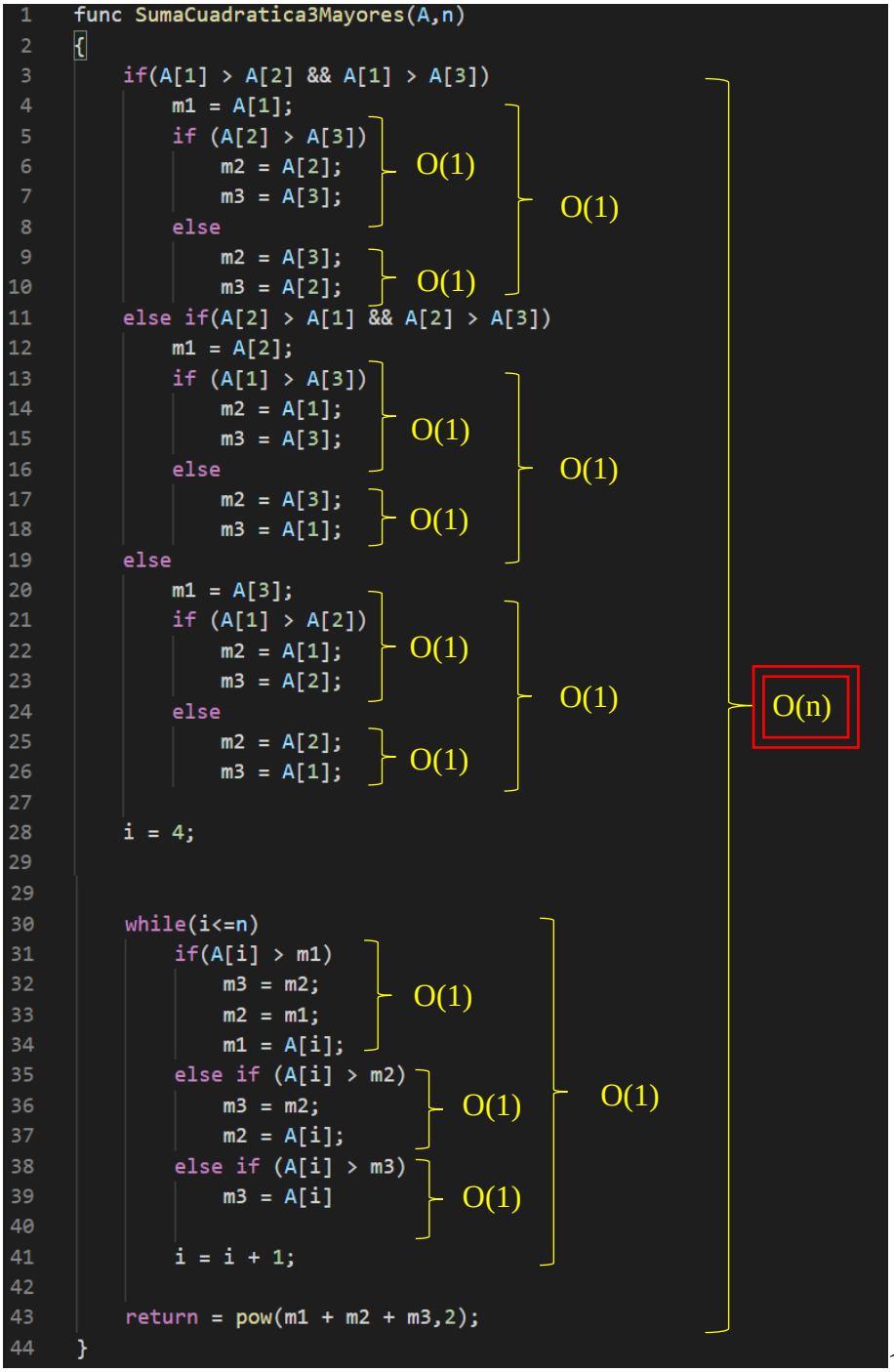
Fecha de entrega: 30 de Abril del 2021

Grupo: 3CM13

INSTITUTO POLITÉCNICO NACIONAL

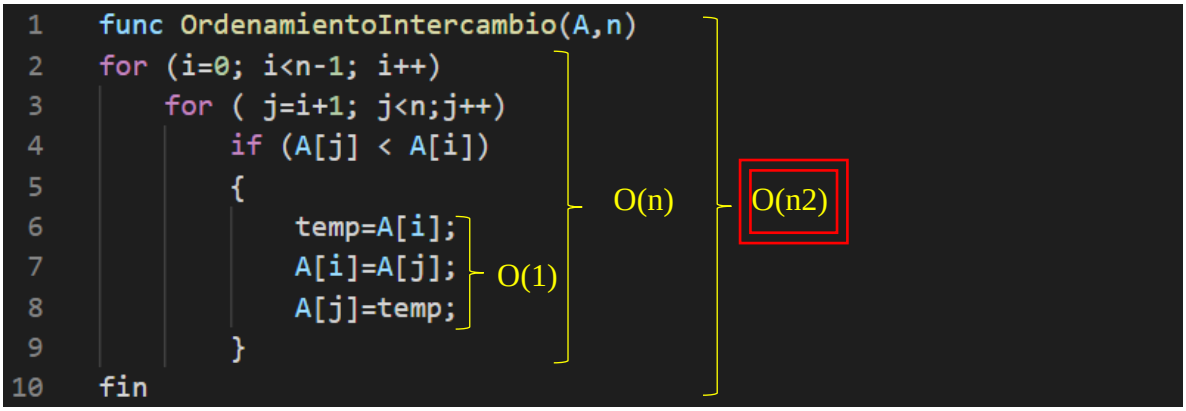
ESCOM

Código 1

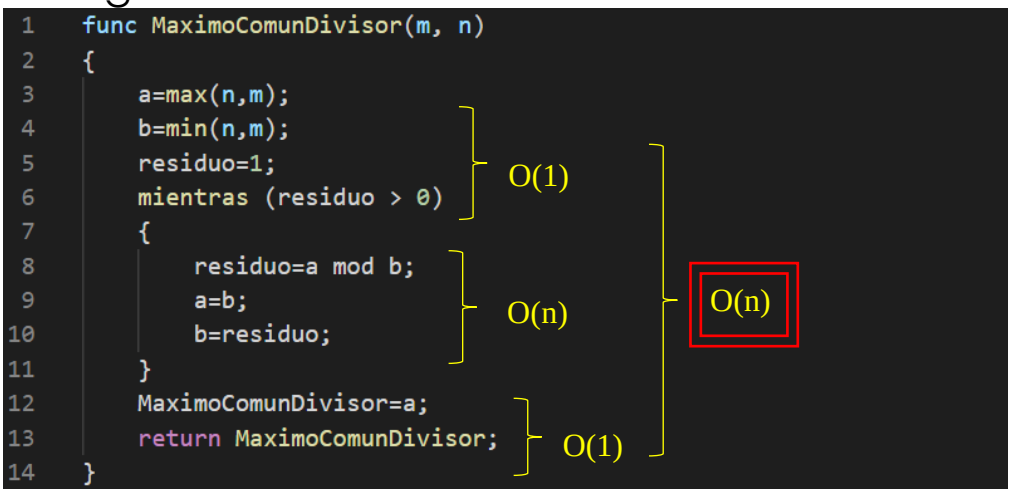

1 func SumaCuadratica3Mayores(A,n)
2 {
3 if(A[1] > A[2] && A[1] > A[3])
4 m1 = A[1];
5 if (A[2] > A[3])
6 m2 = A[2];
7 m3 = A[3];
8 else
9 m2 = A[3];
10 m3 = A[2];
11 else if(A[2] > A[1] && A[2] > A[3])
12 m1 = A[2];
13 if (A[1] > A[3])
14 m2 = A[1];
15 m3 = A[3];
16 else
17 m2 = A[3];
18 m3 = A[1];
19 else
20 m1 = A[3];
21 if (A[1] > A[2])
22 m2 = A[1];
23 m3 = A[2];
24 else
25 m2 = A[2];
26 m3 = A[1];
27
28 i = 4;
29
29
30 while(i<=n)
31 if(A[i] > m1)
32 m3 = m2;
33 m2 = m1;
34 m1 = A[i];
35 else if (A[i] > m2)
36 m3 = m2;
37 m2 = A[i];
38 else if (A[i] > m3)
39 m3 = A[i]
40
41 i = i + 1;
42
43 return = pow(m1 + m2 + m3,2);
44 }
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
O(n)

√

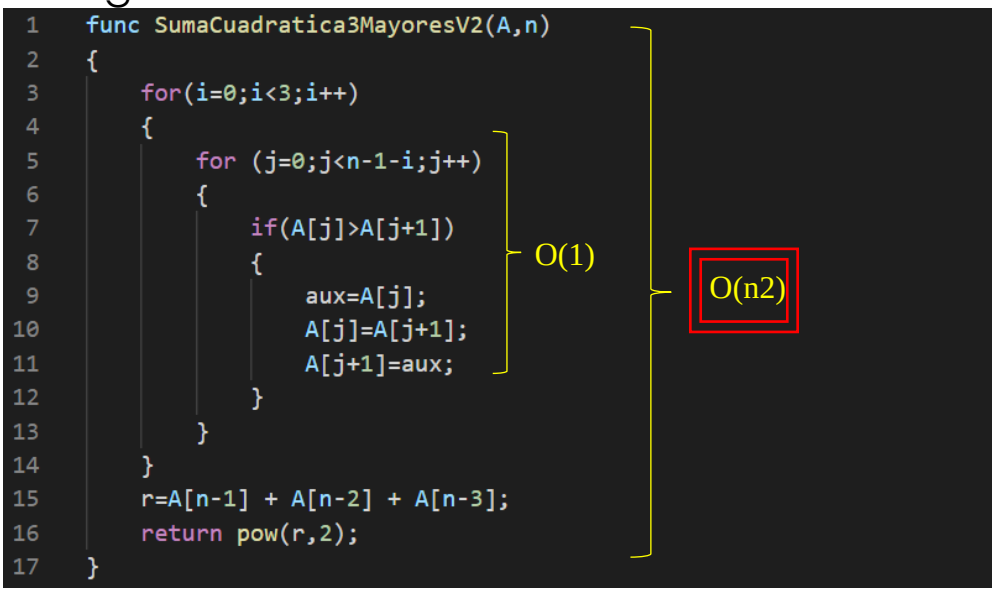
Código 2



Código 3



Código 4



Código 5

```
1   Procedimiento BurbujaOptimizada(A,n)
2       cambios = “Si”
3       i=0
4       Mientras i< n-1 && cambios != “No” hacer
5           cambios = “No”
6           Para j=0 hasta (n-2)-i hacer
7               Si(A[j] < A[j]) hacer
8                   aux = A[j]
9                   A[j] = A[j+1]
10                  A[j+1] = aux
11                  cambios = “Si”
12              FinSi
13          FinPara
14           i= i+1
15      FinMientras
16  fin Procedimiento
```

O(1) O(n) O(n2)

Código 6

```
1   Procedimiento BurbujaSimple(A,n)
2       para i=0 hasta n-2 hacer
3           para j=0 hasta (n-2)-i hacer
4               si (A[j]>A[j+1]) entonces
5                   aux = A[j]
6                   A[j] = A[j+1]
7                   A[j+1] = aux
8               fin si
9           fin para
10      fin para
11  fin Procedimiento
```

O(1) O(n) O(n2)

Código 7

```
1   Proceso ValidaPrimo
2       Leer n
3       divisores<-0
4       si n>0 Entonces
5           Para i<-1 Hasta n Hacer
6               si (n%i=0) Entonces
7                   divisores=divisores+1
8               FinSi
9           FinPara
10      FinSi
11
12      si divisores=2
13          Escribir 'S'
14      SiNo
15          Escribir 'N'
16      FinSi
17  FinProceso
```

O(1) O(1) O(n)

Código 8

```
1   Algoritmo FrecuenciaMinNumeros
2       Leer n
3       Dimension A[n]
4       i=1
5       Mientras i<=n
6           Leer A[i]
7           i=i+1
8       FinMientras
9
10      f=0
11      i=1
12      Mientras i<=n
13          ntemp=A[i]
14          j=1
15          ftemp=0
16          Mientras j<=n
17
18              si ntemp=A[j]
19                  ftemp=ftemp+1
20              FinSi
21              j=j+1
22          FinMientras
23
24          si f<ftemp
25              f=ftemp
26              num=ntemp
27          FinSi
28
29          i=i+1
30      FinMientras
31      Escribir num
32  FinAlgoritmo
```

O(n)

O(n)

O(n)

O(n2)

Código 9

```
1   void search(char* pat, char* txt)
2   {
3       int M = strlen(pat);
4       int N = strlen(txt);
5
6       for (int i = 0; i <= N - M; i++)
7       {
8           int j;
9
10          for (j = 0; j < M; j++)
11              if (txt[i + j] != pat[j])
12                  break;
13
14          if (j == M)
15              printf("Pattern found at index %d \n", i);
16      }
17  }
```

O(1)

O(1)

O(1)

O(n2)

Código 10

```
1   stack<int> sortStack(stack<int> &input)
2   {
3       stack<int> tmpStack;
4
5       while (!input.empty())
6       {
7           int tmp = input.top();
8           input.pop();
9
10          while (!tmpStack.empty() && tmpStack.top() > tmp)
11          {
12              input.push(tmpStack.top());
13              tmpStack.pop();
14          }
15
16          tmpStack.push(tmp);
17      }
18      return tmpStack;
```

O(n)

O(n2)