



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



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
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 }

```

Diagram illustrating the complexity analysis of the code:

- The initial selection of the three largest elements (lines 3-26) is analyzed as $O(1)$ for each of the three possible cases, leading to an overall $O(1)$ complexity for this section.
- The loop (lines 30-41) iterates from $i = 4$ to n . Inside the loop, the complexity of the conditional logic is analyzed as $O(1)$ for each iteration.
- The overall complexity of the function is $O(n)$, indicated by a red box around the $O(n)$ label.

Código 2

```

1  func OrdenamientoIntercambio(A,n)
2  for (i=0; i<n-1; i++)
3      for ( j=i+1; j<n;j++)
4          if (A[j] < A[i])
5              {
6                  temp=A[i];
7                  A[i]=A[j];
8                  A[j]=temp;
9              }
10 fin

```

Diagrama de complejidad:

- El bucle interno (líneas 3-9) tiene una complejidad de $O(1)$.
- El bucle externo (línea 2) tiene una complejidad de $O(n)$.
- La complejidad total del algoritmo es $O(n^2)$.

Código 3

```

1  func MaximoComunDivisor(m, n)
2  {
3      a=max(n,m);
4      b=min(n,m);
5      residuo=1;
6      mientras (residuo > 0)
7          {
8              residuo=a mod b;
9              a=b;
10             b=residuo;
11         }
12     MaximoComunDivisor=a;
13     return MaximoComunDivisor;
14 }

```

Diagrama de complejidad:

- Las líneas 3-5 tienen una complejidad de $O(1)$.
- El bucle "mientras" (líneas 6-11) tiene una complejidad de $O(n)$.
- Las líneas 12-13 tienen una complejidad de $O(1)$.
- La complejidad total del algoritmo es $O(n)$.

Código 4

```

1  func SumaCuadratica3MayoresV2(A,n)
2  {
3      for(i=0;i<3;i++)
4          {
5              for (j=0;j<n-1-i;j++)
6                  {
7                      if(A[j]>A[j+1])
8                          {
9                              aux=A[j];
10                             A[j]=A[j+1];
11                             A[j+1]=aux;
12                         }
13                  }
14          }
15     r=A[n-1] + A[n-2] + A[n-3];
16     return pow(r,2);
17 }

```

Diagrama de complejidad:

- El bucle interno (líneas 5-13) tiene una complejidad de $O(1)$.
- El bucle externo (línea 3) tiene una complejidad de $O(3)$.
- La complejidad total del algoritmo es $O(n^2)$.

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+1]) 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

```

Diagrama de complejidad para Código 5:

- El bucle interno (líneas 6-11) tiene una complejidad de $O(1)$.
- Este bucle se repite $O(n)$ veces.
- Por lo tanto, la complejidad total es $O(n^2)$.

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

```

Diagrama de complejidad para Código 6:

- El bucle interno (líneas 3-7) tiene una complejidad de $O(1)$.
- Este bucle se repite $O(n)$ veces.
- Por lo tanto, la complejidad total es $O(n^2)$.

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

```

Diagrama de complejidad para Código 7:

- El bucle interno (líneas 5-8) tiene una complejidad de $O(1)$.
- Este bucle se repite $O(n)$ veces.
- Por lo tanto, la complejidad total es $O(n)$.

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             si ntemp=A[j]
18                 ftemp=ftemp+1
19             FinSi
20             j=j+1
21         FinMientras
22
23         si f<ftemp
24             f=ftemp
25             num=ntemp
26         FinSi
27
28         i=i+1
29     FinMientras
30     Escribir num
31
32 FinAlgoritmo

```

Diagrama de complejidad para Código 8:

- El bucle `Mientras i<=n` (líneas 5-8) tiene una complejidad de $O(n)$.
- El bucle interno `Mientras j<=n` (líneas 16-21) tiene una complejidad de $O(n)$.
- El bucle `si f<ftemp` (líneas 24-26) tiene una complejidad de $O(1)$.
- El bucle `Mientras j<=n` (líneas 16-21) y el bucle `si f<ftemp` (líneas 24-26) están dentro del bucle `Mientras i<=n` (líneas 12-29), lo que resulta en una complejidad total de $O(n^2)$.

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 }

```

Diagrama de complejidad para Código 9:

- Las líneas 3-4 (`int M = strlen(pat);` y `int N = strlen(txt);`) tienen una complejidad de $O(1)$.
- El bucle `for (int i = 0; i <= N - M; i++)` (línea 6) tiene una complejidad de $O(N)$.
- El bucle interno `for (j = 0; j < M; j++)` (línea 10) tiene una complejidad de $O(M)$.
- El bucle `for (j = 0; j < M; j++)` (línea 10) y el bucle `for (int i = 0; i <= N - M; i++)` (línea 6) están dentro del bucle `for (int i = 0; i <= N - M; i++)`, lo que resulta en una complejidad total de $O(n^2)$.

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;
```

Complexity analysis annotations:

- A yellow bracket on the right side of the code, spanning from line 5 to line 17, is labeled $O(n^2)$ in a red box.
- A yellow bracket on the right side of the code, spanning from line 12 to line 14, is labeled $O(n)$.