



INSTITUTO POLITECNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO (ESCOM)



ANÁLISIS DE ALGORITMOS

NOMBRE DEL ALUMNO:

- SANTOS MÉNDEZ ULISES JESÚS

EJERCICIO 07:

- ANÁLISIS DE ALGORITMOS NO RECURSIVOS

FECHA DE ENTREGA:

- 15/04/2022

GRUPO:

- 3CM14



Análisis de algoritmos no recursivos

Objetivo:

Para los siguientes algoritmos determine la cota de complejidad O (cota superior ajustada) bajo el principio del peor caso, de cada algoritmo, Indique a detalle el procedimiento de obtención de la cota.

Código 01:

$\in O(\log_2 n)$

```
1  Func SumaCubosAtresMayores(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 ]
```

Handwritten annotations on the code:

- For the first `if` block (lines 3-18), a large curly brace on the right indicates a complexity of $O(1)$.
- For the `while` loop (lines 30-41), a curly brace on the right indicates a complexity of $O(\log_2 n)$.
- Inside the `while` loop, the `if` statements are annotated with $O(1)$.

Código 02:

$\in O(n^2)$

```
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
```

$O(1)$ $O(n)$ $O(n^2)$

Código 03:

$\in O(\log_2 n)$

```
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 }
```

$O(1)$ $O(1)$ $O(\log_2 n)$ $O(1)$ $O(1)$

Código 04:

 $\in O(n^2)$

Código 04:

```

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 }

```

(A, n)

$\therefore \in O(n^2)$

$\{O(n)\} \cdot O(n) \cdot O(n) \cdot O(n)$

$[] : \begin{cases} O(n) \\ O(n) \end{cases}$

Código 05:

 $\in O(n \log n)$

```

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

```

$$\in O(n \log_2 n)$$

Código 06:

$\in O(n^2)$

```
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
```

$\left. \begin{matrix} O(1) \\ O(n) \end{matrix} \right\} O(n^2)$

Código 07:

$\in O(n)$

```
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
```

$\left. \begin{matrix} O(1) \\ O(n) \end{matrix} \right\} O(n)$

$\left. \begin{matrix} O(1) \\ O(1) \end{matrix} \right\} O(1)$

$\in O(n)$

Código 08:

 $\in O(3 \log n)$

```
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
```

Handwritten annotations in white on the black background:

- Lines 2-3: $O(1)$
- Lines 5-7: $O(\log n)$
- Line 10: $O(1)$
- Line 11: $O(1)$
- Line 13: $O(1)$
- Lines 14-16: $O(1)$
- Lines 17-21: $O(1)$
- Lines 18-19: $O(1)$
- Lines 20-21: $O(1)$
- Lines 22-23: $O(1)$
- Lines 24-26: $O(1)$
- Line 27: $O(1)$
- Line 28: $O(1)$
- Line 29: $O(1)$
- Line 30: $O(1)$
- Line 31: $O(1)$
- Line 32: $O(1)$
- A large bracket on the right side of the code, spanning from line 12 to line 32, is labeled $O(\log n)$.



Código 09:

```

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        {
12            if (txt[i + j] != pat[j])
13                break;
14        }
15        if (j == M)
16            printf("Pattern found at index %d \n", i);
17    }
18 }

```

Complexity Analysis:

- Line 3: $O(M)$
- Line 4: $O(N)$
- Line 6: $O(N - M + 1)$
- Line 8: $O(1)$
- Line 10: $O(M)$
- Line 11: $O(1)$
- Line 12: $O(1)$
- Line 13: $O(1)$
- Line 14: $O(1)$
- Line 15: $O(1)$
- Line 16: $O(1)$
- Line 17: $O(1)$

Overall Complexity: $O((N - M + 1) * M)$

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(); } O(1)
8          input.pop(); } O(1)
9
10         while (!tmpStack.empty() && tmpStack.top() > tmp) } O(n)
11         {
12             input.push(tmpStack.top()); } O(log n)
13             tmpStack.pop(); } O(1)
14         }
15
16         tmpStack.push(tmp); } O(1)
17     }
18     return tmpStack; } O(1)

```

$\in O(n \log n)$