Auxiliatura INF-143 "A"
Complejidad Algorítmica

Univ. Miguel Angel Quispe Mamani
Universidad Mayor de San Andrés
Carrera de Informática

I/2022

1 Orden de Complejidades

- $O(1)$ Funciones constantes(Fórmulas,...)
- $O(log(n))$ Funciones logarítmicas(Divide y vencerás,...)
- $O(\sqrt{n})$ Funciones raiz cuadrada(Hallar los divisores de un número,...)
- $O(n)$ Funciones lineales(Búsqueda Completa,...)
- $O(nlog(n))$ Funciones lineales por funciones logarítmicas(QuickSort, Merge Sort,...)
- $O(n^2)$ Funciones cuadráticas
- $O(n^3)$ Funciones cúbicas
- . . .
- $O(n^k)$ Funciones polinómicas
- $O(2^n)$ Funciones exponenciales
- $O(n!)$ Funciones factorial
- $O(n^n)$ ...
- $O(n^{n^n})$ ...
- . . .

2 Tabla de Complejidades

Complejidad de tiempos comunes	
Valor máximo de n	Complejidad
∞	$O(1)$
$2^{50000000}$	$O(log(n))$
10^{15}	$O(\sqrt{n})$
50000000	$O(n)$
5000000	$O(nlog(n))$
5000	$O(n^2)$
500	$O(n^3)$
80	$O(n^4)$
20	$O(2^n)$
11	$O(n!)$

3 Reglas para medir la Complejidad

3.1 Regla de la Suma

Esta regla implica que al ejecutar dos algoritmos diferentes, uno después del otro, la complejidad de ejecutar ambos algoritmos será igual a la complejidad de ejecutar el algoritmo mas lento de ellos.

$$O(f(n) + g(n)) = max(f(n), g(n))$$

3.2 Producto por Constante

Si una función $f(n) = Kt(n)$ donde K es una constante, la complejidad de f es la misma que la de t, dicho de otra manera:

$$O(Kf(n)) = O(f(n))$$

3.3 Regla del Producto

Sea $g(x) = f(x)h(x)$, la complejidad de g es igual al producto de las complejidades de f y h.

4 Método Iterativo

4.1 Propiedades mas comunes a utilizar

$$\sum_{i=1}^{n} 1 = n \tag{1}$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{2}$$

$$\sum_{i=0}^{n-1} c^i = \frac{c^n - 1}{c - 1} \tag{3}$$

4.2 Ejemplo 1

Hallar el $T(n)$ y $O(n)$ del siguiente algoritmo:

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    //ordenar el vector ascendentemente
    //Algoritmo burbuja
        //mejor de los casos: que este ordenado ascendentemente
        //peor de los casos: que este ordenado descendentemente
        //caso promedio: que este desordenado
    int n;
    cin >> n;                                       //1
    int v[n];                                       //1
    for(int i = 0; i < n; i++)                      //n + 1
        cin >> v[i];
                                //n
    for(int i = 0; i < n - 1; i++)                  //n
        for(int j = i + 1; j < n; j++)              //(n - 1)(n - 1)
            if(v[j] < v[i]){                        //(n - 1)(n - 2)(1)
                int aux = v[i];                     //(n - 1)(n - 2)(1)
                v[i] = v[j];                        //(n - 1)(n - 2)(1)
                v[j] = aux;                         //(n - 1)(n - 2)(1)
            }
    for(int i = 0; i < n; i++){                     //n + 1
        if(i == n - 1)                              //n(1)
            cout << v[i] << "\n";                   //n(1)
        else
            cout << v[i] << " ";
    }


```

```
31      /*
32      1 + 1 + n + 1 + n + n^2 - n - n + 1 + 4(n^2 - 2n - n + 2) +  n + 1 + n
        + n
33      1 + 1 + n + 1 + n + n^2 - n - n + 1 + 4n^2 - 8n - 4n + 8 + n + 1 + n +
        n
34      T(n) = 5n^2 - 9n + 13
35      T(n) E O(n^2)
36
37      */
38      return 0;
39  }
```