Bartlomiej Maraj

**PROBLEMS UNIT 1**

**3) Analyze the efficiency of the following code:**

We assume the cost of Mínimo(i,j) = 1

T(n) -> O(n ^log(n))

$$T(n) = 1 + \sum_{i=1}^{n} \left(1 + \sum_{i=1}^{n} \left( \sum_{i=1}^{n} (1+1)\right) + 1 + t(n/2)\right)$$

$$= 1 + \sum_{i=1}^{n} \left(1 + \sum_{i=1}^{n} 2n + 1 + t(n/2)\right)$$

$$= 1 + \sum_{i=1}^{n} \left(2 + 2n^2 + t(n/2)\right)$$

$$= 1 + 2n + 2n^3 + n\,t(n/2)$$

$n = 2^k$

$$T(2^k) = 1 + 2\cdot 2^k + 2\cdot(2^k)^3 + 2^k \cdot t(2^k/2)$$

$$T(2^k) = x^k$$

$$x^k = 1 + 2\cdot 2^k + 2\cdot 8^k + 2^k \cdot x^{k-1}$$

$$x^k - 2^k \cdot x^{k-1} = 1 + 2\cdot 2^k + 2\cdot 8^k$$

Homogenous: $x^{k-1}\cdot(x - 2^k) = 0$

roots: $x = 2^k$

$x^{(H)} = A(2^k)^k$

Particular: $x^{k-1}\cdot(x - 2^k) = 1 + 2\cdot 2^k + 2\cdot 8^k$

roots: $x = 1,\ x = 2,\ x = 8$

$x^{(P)} = B + C2^k + D8^k$

$$x = x^{(H)} + x^{(P)} = A(2^k)^k + B + C2^k + D8^k \underset{n=2^k}{=} A\,n^k + B + Cn + Dn^3 \underset{k=\log_2(n)}{=} A\,n^{\log_2 n} + B + Cn + Dn^3$$

$T(n) \rightarrow O(n^{\log n})$

**6) Program a function to determine if a number received as parameter is prime. Analyze the efficiency and complexity.**

```java
public static boolean isPrime(int num) {

    int temp;
    boolean isPrime = true;
    if (num == 0 || num == 1)            // Both 0 and  arent prime numbers
    {
        isPrime = false;
    }
    else
    {
        for(int i = 2; i <= num/2; i++)  // num/2 because we are going through the possible factors,
        {                                // so we dont want to repeat them.
            temp = num % i;
            if(temp == 0)
            {
                isPrime = false;         // If we find a factor different from 0 or num then
                break;                   // num is not prime.
            }
        }
    }
    return isPrime;
}
```

T(n) = 1 + E 1+ 1 + 1

= 1 + 3 * n/2

= 1 + 1,5n

= O(n)

**8) Program a recursive procedure to obtain the inverse number of a given one. Example : 627 -> 726. Analyze the efficiency and complexity.**

```java
public static void reverseNumber(int num, int n, int aux) {
    if (num < 10)
    {
        aux = aux * 10 + num;
        System.out.println(aux);
    }
    else
    {
        aux = aux * 10 + (num % 10 );
        reverseNumber(num/10,n++,aux);
    }
}
```

T(n) = 1 + 1 + t(n/10)

N = 10^k

X^k = 2 + x^k-1

X^k-1(x - 1) = 0    x(H) = A

X^k-1(x - 1) = 2    x(P) = Bk

X = A + Bk = A + log(n)B

= O(log(n))