TD4 Listes

VAN DE MERGHEL Robin

2023

Table des matières

Exercice 4.1

On a les codes suivants : Quelle est la complexité de chacun de ces codes ?

Algorithme 1

```
s = 0;
for (i=0;i<=n;i++) {
    s = s + i;
}
```

On a $n+1$ itérations, dont chaque itération prend 3 unités de temps (incrémentation, addition, et comparaison). De plus il y a une affectation à l'initialisation qui coûte 3 unités de temps. Donc la complexité est $O(3(n+1)+3) = O(n)$.

Algorithme 2

```
s = 0;
for (i=0;i<=n;i++) {
    for (j=i;j<=n;j++) {
        s = s + i;
    }
}
```

On a $n+1$ itérations, dont chaque itération prend 3 unités de temps (incrémentation, addition, et comparaison). De plus, la boucle `for` interne a $n-i+1$ itérations, donc $n(n+1)/2$ itérations au total. Donc la complexité est $O(3(n+1) + 3(n(n+1)/2)) = O(n^2)$.

Algorithme 3

```
for (k=2; k<=n;k++) {
    x = a[k];
    j = k - 1;
    while (j!=0) {
```

```
        if (x < a[j]) {
            j = j - 1;
        } else {
            j = 0;
        }
    }
}
```

On a une boucle `for` qui fait $n-2$ itérations. La boucle while elle a un nombre d'itérations qui dépend de la valeur de j.

Dans le pire cas, on a $j = 0$ à la fin de la boucle `while` grâce à la condition `j!=0`. Donc on a $n-2$ itérations dans la boucle `for` et k itérations dans la boucle `while` pour $k = 2, \dots, n$. Donc la complexité est $\sum_{k=2}^{n} O(k) = O(n^2)$.

Algorithme 4

```
c = n;
while (c > 1) {
    c = c / 5;
}
```

On peut traduire cet algorithme par une suite récurrente :

$$\begin{cases} c_0 = n \\ c_{k+1} = \frac{c_k}{5} \end{cases}$$

C'est à dire :

$$c_k = \frac{n}{5^k}$$

On a donc $c_k \leq 1$ quand $n \geq \log_5(n)$. Donc la complexité est au pire $O(\log_5(n)) = O(\log(n))$.

Exercice 4.2

Notations O

Proposition 1

Montrez que $(n+a)^b = O(n^b)$ pour tout a et b

On a :

$$\begin{aligned} (n+a)^b &= \sum_{k=0}^{b} \binom{b}{k} a^{b-k} n^k \\ &= n^k + \alpha_1 n^{k-1} + \alpha_2 n^{k-2} + \dots + \alpha_{k-1} n + \alpha_k \end{aligned}$$

Avec $\alpha_i = \binom{b}{i}$.

On a donc :

$$\begin{aligned} O((n+a)^b) &= O(n^b + \alpha_1 n^{b-1} + \alpha_2 n^{b-2} + \dots + \alpha_{b-1} n + \alpha_b) \\ &= O(n^b) + O(\alpha_1 n^{b-1}) + O(\alpha_2 n^{b-2}) + \dots + O(\alpha_{b-1} n) + O(\alpha_b) \\ &= O(n^b) + O(n^{b-1}) + O(n^{b-2}) + \dots + O(n) + O(1) \end{aligned}$$

On a $O(1)$ négligeable par rapport à $O(n)$, puis $O(n)$ négligeable par rapport à $O(n^2)$, etc. Jusqu'à $O(n^{b-1})$ qui est négligeable par rapport à $O(n^b)$. Donc $O((n+a)^b) = O(n^b)$.

Proposition 2

Est-ce que la phrase suivante a un sens : "Le temps d'exécution de l'algorithme est $O(n^2)$" ? Et pourquoi ?