

TD 1 et 2 Diviser pour régner

1 Ordres de grandeur et complexité

Exercice 1. Ordonnez les complexités suivantes de la plus petite à la plus grande :

1. $n!$
2. 2^n
3. n^n
4. $n \log n$
5. $n^2 \log n$
6. $n^{1/3}$
7. n^3
8. $n \log^4 n$
9. $n \log_4 n$
10. $\log n$
11. $n^{1.5}$
12. \sqrt{n}

Exercice 2. Donnez l'ordre de grandeur des expressions suivantes :

1. $5 + 0,001n^3 + 0,024n$
2. $100 \log n + 0.1n$
3. $5n \ln n + 100n$
4. $n \log_3 n + n$
5. $100n + 0,001n^2$
6. $0,01n \log_2 n + n (\log_2 n)^2$

Exercice 3. Donnez la complexité asymptotique d'algorithmes récursifs dont la complexité peut être exprimée avec les récurrences suivantes :

1. $T(n) = T(n/2) + n$
2. $T(n) = 4T(n/2) + n^2$
3. $T(n) = 3T(n/4) + n \log n$
4. $T(n) = 4T(n/2) + \log n$
5. $T(n) = T(n-1) + n$
6. $T(n) = 6T(n/2) + n^2 \log n$
7. $T(n) = 8T(n/3) + 2^n$

2 Algorithmique

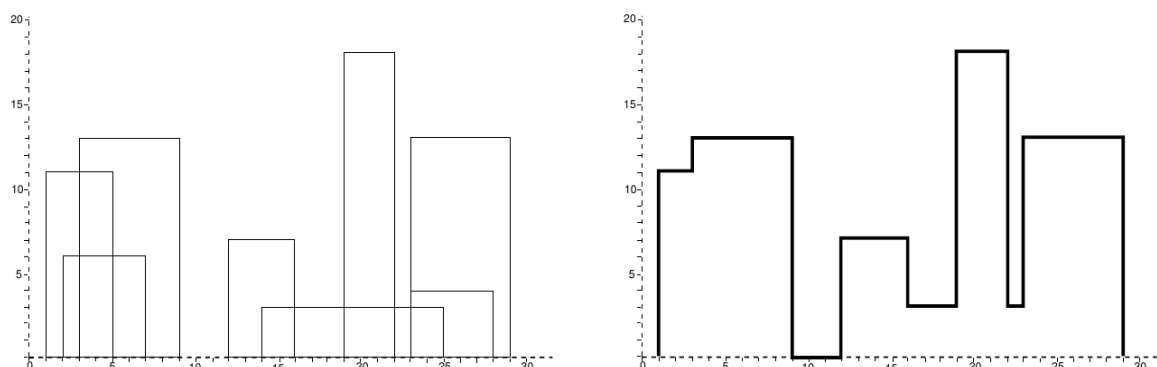
Exercice 4. Écrivez un algorithme qui prend en entrée un nombre x et un entier n et qui calcule x^n . Quelle est sa complexité en nombre de multiplications?

Exercice 5. Écrivez un algorithme qui prend en entrée un tableau T d'entiers trié par ordre croissant et tel que tous les éléments sont distincts et qui retourne un indice i tel que $T[i] = i$ si celui-ci existe. La complexité de l'algorithme devra être en $\mathcal{O}(\log n)$.

Exercice 6. Un élément est *majoritaire* dans un tableau de taille n s'il y occupe au moins $\lfloor n/2 \rfloor + 1$ cases.

1. Donner un algorithme simple en $\Theta(n^2)$ qui retourne l'élément majoritaire s'il existe et Faux sinon.
2. Même question en $\mathcal{O}(n \log n)$ en utilisant l'approche diviser pour régner.
3. Même questions en $\mathcal{O}(n \log n)$ sans utiliser diviser pour régner.
4. Même question en $\mathcal{O}(n)$. *Indication* : lorsque deux éléments du tableau sont différents et que l'on les enlève, s'il y avait un élément majoritaire il le reste dans le tableau restant.

Exercice 7. On s'intéresse ici au problème de la *skyline* d'immeubles. Un immeuble B_i est représenté par un triplet (L_i, H_i, R_i) où L_i et R_i sont les coordonnées en x du début et de la fin de l'immeuble et H_i sa hauteur. La *skyline* est la courbe des hauteurs maximales d'immeubles pour chaque coordonnée. Par exemple, avec les immeubles à gauche, la skyline est représentée à droite.



Une skyline pour un ensemble de n immeubles sera représentée par une liste de couples $[(x_1, h_1) \dots (x_n, h_n)]$ où x_i représente la coordonnée d'un changement de hauteur et h_i la nouvelle hauteur. Ainsi, h_n sera toujours égale à 0 par exemple et h_i est la hauteur de la skyline entre x_i et x_{i+1} .

Si on reprend la liste d'immeubles de l'exemple précédent :

$$\{(3, 13, 9), (1, 11, 5), (12, 7, 16), (14, 3, 25), (19, 18, 22), (2, 6, 7), (23, 13, 29), (23, 4, 28)\}$$

sa skyline est $[(1, 11) (3, 13) (9, 0) (12, 7) (16, 3) (19, 18) (22, 3) (23, 13) (29, 0)]$.

Donnez un algorithme pour trouver la skyline de n immeubles qui fonctionne en $\mathcal{O}(n \log n)$.

Exercice 8. Soit A est un arbre binaire de recherche (ABR) : pour tout noeud x , sa valeur est inférieure strictement à toute valeur qui est dans son sous-arbre droit et supérieure ou égale à toute valeur qui est dans son sous-arbre gauche.

1. Écrire un algorithme pour trouver le nombre d'apparitions d'une valeur x dans un arbre de binaire de recherche. Les paramètres d'entrée de votre algorithme sont A , un ABR, et x , une valeur. L'algorithme doit retourner un entier qui est 0 si $x \notin A$ et le nombre d'apparitions sinon.
2. Calculer, en notation asymptotique Θ , le nombre de comparaisons dans le meilleur des cas
3. Calculer, en notation asymptotique Θ , le nombre de comparaisons dans le pire des cas.
4. Vous devez commenter votre algorithme et justifier les complexités.