Portail Descartes
Introduction à la science info : TD 5
Grand O, chaînes de caractères et tri

amU Faculté des sciences Aix Marseille Université

Exercice 1 Pour chacune des paires de termes généraux de suites ci-dessous, l'une est-elle un « grand O » de l'autre ? Justifier vos réponses.

1. $n^2 + 2n$ et n^3 ?
2. n et $n + 42$?
3. $n + 2n \log_2 n$ et $n \log_2 n$?
4. $8n^5 + 5n^4 - 3n + 2$ et n^6 ?

Exercice 2 Reprendre les exercices du TD 4 et donner la complexité des différentes fonctions en utilisant le « grand O » le plus simple possible.

1. dans l'exercice 2, on a trouvé une complexité de $3n+4$ opérations pour la fonction de recherche séquentielle dans un tableau trié ;
2. dans l'exercice 3, on a trouvé des complexités de $2n + 5$ opérations pour la moyenne et de $4n + 10$ pour la variance ;
3. dans l'exercice 4, on a trouvé une complexité de $2n\lfloor \log_2 n \rfloor + 3\lfloor \log_2 n \rfloor + 8$;
4. dans la dernière question de l'exercice 5, on a trouvé une complexité de $4n^2 + 2n - 1$.

Exercice 3 Écrire une fonction `flip` qui prend en paramètre une chaîne `s` de caractères et renvoie la chaîne obtenue en lisant `s` de droite à gauche.

Exercice 4 Écrire une fonction `swap_case` qui prend en paramètre une chaîne de caractère et renvoie la chaîne obtenue en remplaçant chaque lettre minuscule (non accentuée) par une lettre majuscule et inversement, et ne modifiant pas les caractères qui ne sont pas des lettres (non accentuées).

Exercice 5 Le tri par insertion étudié en cours peut s'écrire de la façon suivante en Python :

```
1  def insertionsort(t):
2      n = len(t)
3      for i in range(1, n):
4          x = t[i]
5          j = i
6          while j > 0 and x < t[j - 1]:
7              t[j] = t[j - 1]
8              j = j - 1
9          t[j] = x
10     return t
```

1. Que se passe-t-il si on remplace la ligne 3 par `for i in range(n)` ?
2. Montrer que cette fonction termine.

Exercice 6 Le tri par sélection est un autre tri classique de tableaux. On commence par chercher le minimum du tableau et on l'échange avec l'élément en première position. Ensuite, on cherche le minimum du tableau en ignorant le premier élément et on l'échange avec l'élément en deuxième position. Puis, on cherche le minimum du tableau en ignorant les deux premiers éléments et on l'échange avec l'élément en troisième position. On poursuit ainsi jusqu'à la fin du tableau.

1. Commencer par écrire une fonction `minimum_index` qui prend en paramètre un tableau `t` (de longueur n) et deux entiers `a` et `b` tels que $0 \leq$ `a` $<$ `b` $< n$ (on ne vérifiera pas ces hypothèses), et renvoie un indice j entre `a` et `b` qui contient dans `t` un élément minimum de la portion de tableau entre `a` et `b`. Par exemple, l'appel à `minimum_index([1, 5, 6, 2, 8, 0], 1, 4)` doit renvoyer 3 puisque c'est l'indice de l'élément 2 qui est minimal parmi les éléments $5, 6, 2, 8$ (ceux entre les indices 1 et 4).
2. Utiliser cette fonction pour écrire une fonction `selectionsort` qui prend en paramètre un tableau et le modifie pour le trier en utilisant le tri par sélection, puis le renvoie.

3. Quelle est la complexité dans le pire des cas du tri par sélection ? Exprimer cette complexité avec un « grand O » dépendant de la longueur du tableau à trier. *On commencera par étudier la complexité de la fonction* `minimum_index` *en fonction de ses deux derniers arguments.*

Exercice 7 Lorsqu'on trie des entiers positifs ou nuls, qu'on sait tous strictement inférieurs à une constante m, on peut utiliser un tri différent des précédents, appelé le tri par comptage. Il consiste à compter le nombre d'occurrences de chacun des entiers entre 0 et $m-1$ dans le tableau. On peut stocker ces comptes dans un tableau de longueur m. On reconstitue ensuite un tableau trié en parcourant le tableau de comptes.

1. Écrire une fonction `maximum` qui prend en paramètres un tableau d'entiers positifs ou nuls et renvoie l'élément maximum qui apparait dans le tableau.
2. Écrire une fonction `count` qui prend en paramètres un tableau `t` d'entiers positifs ou nuls et renvoie un tableau de longueur $m+1$ dont la case d'indice i contient le nombre d'occurrences de i dans `t`, en notant m le maximum du tableau en paramètre.
3. Utiliser cette fonction pour écrire une fonction `countsort` qui prend en paramètres un tableau `t` d'entiers positifs ou nuls et renvoie une copie triée du tableau `t` (on ne modifiera donc pas le tableau `t`).
4. Quelle est la complexité dans le pire des cas du tri par comptage en fonction de la longueur n du tableau à trier et de la borne m ? Exprimer cette complexité avec un « grand O » dépendant de la longueur du tableau à trier lorsqu'on a $m \leq \log_2 n$, $m \leq n$ et $m \leq 2^n$: quand est-il intéressant d'utiliser cette méthode de tri ?

Exercice 8 On a vu en cours que, dans le pire des cas, la complexité du tri par insertion est en $O(n^2)$. On peut raisonnablement se poser la question de savoir si on a surestimé le nombre d'opérations élémentaires. En fait, il n'en est rien. Trouvez donc une suite de tableaux $(t_n)_{n\in\mathbf{N}}$ avec t_n un tableau de longueur n telle que le nombre C_n d'opérations élémentaires effectuées lors du tri du tableau t_n est de la forme an^2+bn+c avec a, b, c des constantes et $a > 0$.