Série10

Sauf si spécifié autrement, tous les algorithmes demandés sont à écrire sous forme d'une fonction Python.

Exercice 1

(a) Simulez à la main le parcours de `tri_par_selection` sur la liste ci-dessous en affichant le contenu de la liste à la fin de chaque itération de la boucle `for` extérieure. Vérifiez votre réponse en appelant `tri_par_selection` sur la liste en entrée dans un Jupyter Notebook (et en insérant une instruction `print` dans le code).
`[7, 2, 13, -5, -3, 17, 8, 24, -11, 1]`

(b) Même question pour `tri_par_insertion`.

(c) On considère l'algorithme de fusion vu en cours (slide 19). Qu'affichent les appels suivants à `fusion` ?

(i)
```
L = [1, 3, 5, 7, 2, 4, 6, 8]
fusion(L, 2, 3, 5)
print(L)
```

(ii)
```
L = [5, 1, 5, 6, 1, 2, 4, 3]
fusion(L, 2, 3, 6)
print(L)
```

(iii)
```
L = [1, 3, 5, 7, 1, 2, 3, 4]
fusion(L, 3, 3, 7)
print(L)
```

(d) Le fichier `tri_par_fusion.ipynb` contient le code de l'algorithme de tri par fusion vu en cours, avec des instructions `print` ajoutées de façon à montrer l'état de la liste à différents points de l'exécution. Réfléchissez à l'affichage produit par l'exécution des instructions suivantes :
`L = [4, 3, 2, 1]`
`tri_par_fusion(L, 0, len(L)-1)`
et vérifiez votre réponse en exécutant le code. Testez votre compréhension de l'algorithme de tri par fusion en exécutant le code donné pour divers choix de la liste `L`.

Exercice 2

Nous avons analysé en cours le temps de parcours de `tri_par_insertion` au pire des cas, qui correspond au cas où la liste est triée par ordre décroissant. A quoi correspond le meilleur des cas pour `tri_par_insertion` ? Donner l'ordre du temps de parcours en notation $\Theta(\cdot)$ dans ce cas.

Exercice 3

Récrivez l'algorithme `fusion` vu en cours sans utiliser les valeurs sentinelles `float(''inf'')`.

Exercice 4

Le **tri à bulles** (bubble sort) fonctionne de la manière suivante : Il prend en entrée une liste `L` de taille n. Il parcourt la liste n fois. A chaque parcours, dès qu'il rencontre deux éléments adjacents de la liste qui sont dans le mauvais ordre, c'est-à-dire `L[i], L[i+1]` tels que `L[i] > L[i+1]`, il les échange.
On appelle cet algorithme tri à bulles car à chaque parcours de la liste, le plus grand élément (restant) se déplace vers la fin de la liste comme une bulle qui remonte à la surface de l'eau.

(a) Observez une simulation du tri à bulles sur l'un des sites proposés en cours, par exemple https://visualgo.net/bn/sorting, pour comprendre exactement comment fonctionne l'algorithme.

(b) Donnez l'algorithme du tri à bulles en Python. N'oubliez pas de tester votre code sur de petites listes.

(c) Formulez et prouvez un invariant de boucle pour la boucle extérieure de l'algorithme.

(d) Quel est le temps de parcours du tri à bulles ?

Exercice 5

Etant donnés un nombre x et une liste `L` de n nombres tous distincts, on veut trouver un algorithme pour déterminer s'il existe deux éléments de `L` dont la somme vaut `x`. L'algorithme doit retourner des valeurs distinctes `L[i]` et `L[j]` telles que `L[i] + L[j] = x`. S'il n'existe pas de tels éléments, il retourne `None`.
Par exemple, pour l'entrée `10, [1, 2, 3, 5, 7, 8]`, l'algorithme peut retourner `(2, 8)` ou `(3, 7)`, mais pas `(5, 5)`.

(a) Donnez un algorithme naïf de temps de parcours quadratique qui résoud ce problème.

(b) Donnez un algorithme qui résoud ce problème en temps $\Theta(n \log_2(n))$. Vous pouvez faire appel à des algorithmes vus en cours.
Indice : [1]

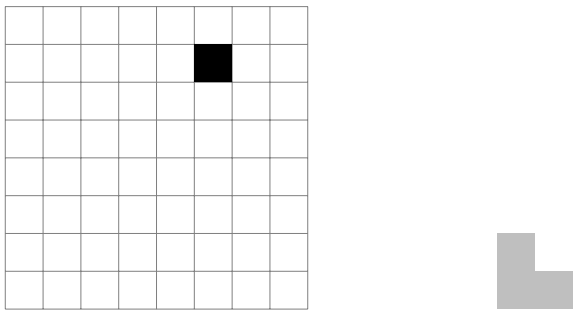
(c) On suppose maintenant que la liste `L` est déjà triée. Donner un algorithme qui résoud ce problème en temps linéaire en n.
Indice : [2]

Exercice 6 (facultatif)

Pour $n \geq 1$, on vous donne

— un tableau de dimensions $2^n \times 2^n$ avec une case manquante (qui peut être n'importe quelle case)

— et un nombre suffisant de **trominos** : ce sont des pièces qui couvrent trois cases d'un tableau formant un angle droit.



Un tableau 8×8 avec la case manquante indiquée en noir ; un tromino

Donnez un algorithme (en français) qui permet de paver le tableau complètement et exactement avec des trominos (les trominos ne se chevauchent pas, ne dépassent pas, la case manquante n'est pas couverte, et aucune autre case ne reste découverte).

Indice : [3]

1. Commencez par trier la liste. Puis répétez une certaine opération n fois.
2. Mettez un doigt au début de la liste et un doigt à la fin...
3. Divisez pour régner.