Diviser pour régner

Algorithmique

Exercice 1 *QCM*

Pour chaque question, une seule réponse est correcte.

1. Le paradigme « diviser pour régner » repose sur trois étapes. Lesquelles ?
   - A. Lire, trier, afficher
   - B. Diviser le problème, résoudre les sous-problèmes, combiner les résultats
   - C. Comparer, échanger, recommencer
   - D. Initialiser, itérer, terminer
2. Quelle est la complexité de la recherche dichotomique dans un tableau trié de n éléments ?
   - A. $O(n)$
   - B. $O(n^2)$
   - C. $O(\log n)$
   - D. $O(n \log n)$
3. Quelle est la complexité du tri fusion dans le pire cas ?
   - A. $O(n^2)$
   - B. $O(n)$
   - C. $O(2^n)$
   - D. $O(n \log n)$
4. Quel est l'inconvénient principal du tri fusion par rapport au tri par insertion ?
   - A. Il nécessite de la mémoire supplémentaire proportionnelle à n
   - B. Sa complexité est moins bonne dans le pire cas
   - C. Il n'est pas stable
   - D. Il ne fonctionne que sur des entiers

Exercice 2 *Dérouler la recherche dichotomique (exercice guidé)*

On considère le tableau trié `tab = [3, 7, 11, 18, 25, 33, 42, 56, 68]` (neuf éléments, indices de 0 à 8).

On cherche la valeur 33.

a) Au départ, les bornes sont `gauche = 0` et `droite = 8`. Calculez l'indice du milieu : `milieu = (gauche + droite) // 2`. Quelle est la valeur de `tab[milieu]` ? Est-elle égale à 33, inférieure ou supérieure ?

b) En déduire les nouvelles bornes. Recalculez le milieu et comparez `tab[milieu]` à 33.

c) Continuez jusqu'à trouver 33. Combien d'étapes ont été nécessaires ?

d) Déroulez maintenant la recherche de la valeur 20 (absente du tableau). À quelle condition l'algorithme s'arrête-t-il en concluant que 20 n'est pas dans le tableau ?

e) Combien de comparaisons au maximum faut-il pour un tableau de neuf éléments ? Vérifiez avec la formule $\lfloor \log_2 n \rfloor + 1$.

Exercice 3 *Dérouler le tri fusion (exercice guidé)*

On souhaite trier le tableau `[38, 12, 45, 7, 21, 33]` par tri fusion.

a) Première étape : diviser. Coupez le tableau en deux moitiés. Quelles sont-elles ?

b) Continuez à diviser chaque moitié jusqu'à obtenir des tableaux d'un seul élément. Dessinez l'arbre de décomposition complet.

c) Deuxième étape : fusionner. On commence par le bas de l'arbre. Fusionnez `[12]` et `[45]` en un tableau trié, puis `[21]` et `[33]`. Que deviennent `[38]` et `[7]`, restés seuls à leur niveau ?

d) Remontez l'arbre en fusionnant les sous-tableaux triés deux à deux. Détaillez chaque fusion en indiquant les comparaisons effectuées.

e) Combien de comparaisons ont été effectuées au total ? Combien en aurait fait le tri par sélection sur ce même tableau de six éléments ?

Exercice 4 *Compléter la fonction `fusion` (exercice guidé)*

La fonction `fusion` prend deux listes **déjà triées** et renvoie une seule liste triée contenant tous les éléments. Complétez les parties manquantes (notées `...`).

```
def fusion(gauche, droite):
    resultat = []
    i, j = 0, 0
    while i < len(gauche) and j < len(droite):
        if gauche[i] <= droite[j]:
            resultat.append(...)  # (1)
            i += 1
        else:
            resultat.append(...)  # (2)
            ...                   # (3)
    # Ajouter les éléments restants
    resultat.extend(gauche[...])  # (4)
    resultat.extend(droite[...])  # (5)
    return resultat
```

a) Que faut-il écrire à la place de `(1)` et `(2)` ? Justifiez.

b) Que faut-il écrire à la place de `(3)` ?

c) Que faut-il écrire à la place de `(4)` et `(5)` ? Pourquoi faut-il ajouter les éléments restants ?

d) Testez votre fonction avec `fusion([3, 8, 15], [1, 9, 12])`. Détaillez les étapes.

Exercice 5 *Programmer le tri fusion*

On dispose de la fonction `fusion` de l'exercice précédent.

a) Quel est le cas de base du tri fusion ? C'est-à-dire : pour quel type de tableau n'a-t-on rien à trier ?

b) Écrivez la fonction `tri_fusion(tab)` qui :

— renvoie `tab` si c'est le cas de base ;

— sinon, coupe `tab` en deux moitiés, trie récursivement chaque moitié, puis les fusionne.

c) Vérifiez que `tri_fusion([38, 27, 43, 3, 9, 82, 10])` renvoie `[3, 9, 10, 27, 38, 43, 82]`.

d) En quoi cet algorithme relève-t-il du paradigme diviser pour régner ? Identifiez chacune des trois étapes (diviser, régner, combiner).

Exercice 6 *Recherche dichotomique récursive*

a) Écrire une fonction récursive `recherche_dicho(x, tab, gauche, droite)` qui renvoie l'indice de `x` dans le tableau trié `tab` (entre les indices `gauche` et `droite`), ou -1 si `x` est absent.

b) Écrire une fonction enveloppe `chercher(x, tab)` qui appelle `recherche_dicho` avec les bornes initiales.

c) En quoi cet algorithme relève-t-il du paradigme diviser pour régner ? Quelle particularité a-t-il par rapport au tri fusion ?

Exercice 7 *Rotation d'un quart de tour d'une image*

Une image carrée en niveaux de gris est représentée par une liste de listes (matrice carrée) de dimension $n \times n$, où n est une puissance de deux.

Par exemple, l'image 4×4 suivante :

```
image = [
    [1,  2,  3,  4],
    [5,  6,  7,  8],
    [9,  10, 11, 12],
    [13, 14, 15, 16]
]
```

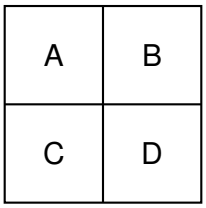
Après rotation d'un quart de tour dans le sens horaire, elle devient :

```
[
    [13, 9,  5, 1],
    [14, 10, 6, 2],
    [15, 11, 7, 3],
    [16, 12, 8, 4]
]
```

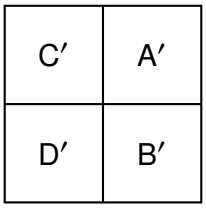
a) Vérifiez sur l'exemple : où se retrouve le pixel en position `(0, 0)` (valeur 1) après rotation ? Et le pixel en position `(3, 0)` (valeur 13) ? Quelle est la règle générale : le pixel `(i, j)` se retrouve en position ... ?

b) Écrire une fonction `rotation_directe(image)` qui effectue la rotation sans DPR, en appliquant la règle trouvée en a).

c) Voici l'approche diviser pour régner. On découpe l'image $n \times n$ en quatre blocs de taille $\frac{n}{2} \times \frac{n}{2}$:



Après rotation d'un quart de tour horaire, les blocs se réarrangent ainsi :



où chaque bloc est lui-même tourné d'un quart de tour. Expliquez pourquoi.

d) Écrire les fonctions auxiliaires suivantes :

— `extraire_bloc(image, ligne_debut, col_debut, taille)` : renvoie le sous-bloc de `taille` × `taille` pixels ;

— `assembler(haut_gauche, haut_droite, bas_gauche, bas_droite)` : assemble quatre blocs en une seule image.

e) Écrire la fonction récursive `rotation_dpr(image)` utilisant l'approche DPR. Quel est le cas de base ?

f) Vérifier que les deux fonctions donnent le même résultat sur l'image 4×4.

Exercice 8 *Synthèse – comparer les algorithmes*

a) Complétez le tableau suivant :

Algorithme	**Paradigme**	**Complexité (pire cas)**	**En place ?**
Recherche séquentielle	...	...	Oui
Recherche dichotomique	...	...	Oui
Tri par sélection	Itératif	...	...
Tri par insertion	Itératif	...	...
Tri fusion	...	...	...

b) Pour un tableau d'un million d'éléments, estimez le nombre d'opérations pour la recherche séquentielle et la recherche dichotomique. Quel est l'ordre de grandeur du facteur de gain ?

c) Un élève affirme : « le tri fusion est toujours le meilleur choix pour trier ». Donnez un argument en sa faveur et un contre-argument.

d) La recherche dichotomique nécessite un tableau trié. Si l'on dispose d'un tableau non trié de n éléments et qu'on doit y faire k recherches, quel est le coût total si l'on trie d'abord avec le tri fusion puis on fait k recherches dichotomiques ? À partir de combien de recherches cette stratégie devient-elle plus avantageuse que k recherches séquentielles ?