

Rappel - *instructions élémentaires : opérations mathématiques et booléennes, comparaisons, affectations, return, accès tableau*

Exercice 1 (Recherche dichotomique récursive).

L'objectif est de calculer la complexité en temps $T(N)$ de l'algorithme de recherche dichotomique ci-dessous, appelé avec $m = 1$ et $n = N$. Pour simplifier, on supposera que $N = 2^k$.

Première technique :

1. Dessiner l'arbre des appels récursifs de `rechercheDicho` pour *une exécution* de l'algorithme, avec un tableau de taille $N = 2^4$. Écrire $n - m + 1$ (taille du sous-tableau courant) sur chaque sommet.
2. Calculer le nombre d'opérations à chaque appel récursif.
3. Dans le cas général $N = 2^k$, quelle est la hauteur de l'arbre en fonction de N ? (*hauteur d'un arbre : nombre maximum de sommets sur un chemin de la racine à une feuille*)
4. En déduire la complexité en temps $T(N)$ dans le pire des cas de l'algorithme.

Deuxième technique en passant par des équations (on repart de zéro) :

5. (a) Combien vaut $T(1)$?
(b) Exprimer $T(N)$ en fonction de $T(N/2)$ quand $N > 1$.
6. (a) Remplacer N par $N/2$ dans la formule obtenue, pour exprimer $T(N/2)$ en fonction de $T(N/4)$.
(b) Substituer la valeur obtenue pour $T(N/2)$ dans la formule précédente, pour obtenir $T(N)$ seulement en fonction de $T(N/4)$. Simplifier.
7. Généraliser pour exprimer (sans preuve) $T(N)$ en fonction de $T(N/2^i)$ (i ème étape récursive). *Bonus : prouver la formule par récurrence.*
8. On rappelle que $N = 2^k$, donc si $i = k$ on a $T(N/2^i) = T(1)$. En déduire l'expression de $T(N)$ uniquement en fonction de N .

```
rechercheDicho(TAB, m, n, x): calcul d'une position de x dans TAB
Entrée : Un tableau d'entiers TAB[1...N] triés par ordre croissant, un entier x,
deux positions m et n
Sortie : 0 si x ∉ TAB, et une position de x dans TAB (entre m et n) sinon

• Si m == n :
  * Si TAB[m] == x :
    Retourner m
  * Sinon :
    Retourner 0

• Sinon :
  * k = ⌊(m+n)/2⌋
  * Si TAB[k] < x :
    Retourner rechercheDicho(TAB, k+1, n, x)
  * Sinon :
    Retourner rechercheDicho(TAB, m, k, x)
```

Exercice 2 (Diviser pour régner : le tri par fusion).

Pour simplifier, on supposera que $N = 2^k$ est une puissance de deux. La complexité de la sous-fonction `interClassement( $TAB_1, TAB_2$ )` est $13(N_1 + N_2) + 6$ dans le pire des cas.

Première technique :

1. Dessiner l'arbre des appels récursifs de `triFusion` pour un tableau de taille $N = 2^4$.
2. Quelle est la complexité approximative de l'algorithme sur *chaque niveau* de l'arbre, en fonction de N ? (*niveau d'un arbre : ensemble de sommets qui sont à la même distance de la racine*)
3. En déduire une estimation informelle de la complexité globale de l'algorithme en fonction de N , en utilisant la hauteur de l'arbre. (*hauteur d'un arbre : nombre maximum de sommets sur un chemin de la racine à une feuille*)

Deuxième technique :

4. On va maintenant calculer plus précisément cette complexité. Exprimer la complexité $T(N)$ de `triFusion` en fonction de $T(N/2)$. (On supposera que copier un tableau de taille N a une complexité de N .)
5. Appliquer la formule précédente à $T(N/2)$ pour substituer $T(N/2)$ dans la formule initiale. Ainsi on exprime $T(N)$ en fonction de $T(N/4)$.
6. Faire encore une étape en exprimant $T(N)$ en fonction de $T(N/8)$.
7. En déduire (sans preuve) $T(N)$ en fonction de $T(N/2^i)$ (i ème étape récursive). (*Bonus : prouver la formule par récurrence.*)
8. On rappelle que $N = 2^k$, donc si $i = k$ on a $T(N/2^i) = T(1)$. En déduire l'expression de $T(N)$ uniquement en fonction de N .

`triFusion( $TAB, m, n$ ):` tri du sous-tableau $TAB[m...n]$ d'entiers par ordre croissant

Entrée : Un tableau d'entiers $TAB[1...N]$ et deux positions m, n avec $m \leq n$

Sortie : Le sous-tableau $TAB[m...n]$ trié

- Si $m < n$:
 - * $k = \lfloor \frac{m+n}{2} \rfloor$
 - * $TAB_1 = \text{triFusion}(TAB, m, k)$
 - * $TAB_2 = \text{triFusion}(TAB, k+1, n)$
 - * $TAB[m...n] = \text{interClassement}(TAB_1, TAB_2)$
- Retourner $TAB[m...n]$

`interClassement( $TAB_1, TAB_2$ ):`

Entrée : Deux tableaux d'entiers $TAB_1[1...N_1]$, $TAB_2[1...N_2]$ triés par ordre croissant

Sortie : L'union TAB de TAB_1 et TAB_2 triée par ordre croissant

- $i = 1, j = 1, k = 1$
- TAB est un tableau vide à $N_1 + N_2$ éléments
- Tant que $i \leq N_1$ et $j \leq N_2$:
 - Si $TAB_1[i] < TAB_2[j]$:
 $TAB[k] = TAB_1[i]$
 $i = i + 1$
 - Sinon :
 $TAB[k] = TAB_2[j]$
 $j = j + 1$
 - $k = k + 1$
- Tant que $i \leq N_1$:
 $TAB[k] = TAB_1[i]$
 $i = i + 1$
 $k = k + 1$
- Tant que $j \leq N_2$:
 $TAB[k] = TAB_2[j]$
 $j = j + 1$
 $k = k + 1$
- Retourner TAB

Exercice 3 (Cherche algorithme pour formule récursive).

Écrire un algorithme récursif avec une complexité exprimée par la formule $T(n) = 3T(n/2) + 5n + 5$.