Exercice de programmation en C

Saut de valeur maximale

On considère un tableau de `double` de taille `n`.
Dans un tableau `tab`, on appelle **saut** un couple d'indices (i, j) tel que

$$0 \leq i \leq j < n.$$

La **valeur** du saut (i, j) est définie par

`tab[j] - tab[i]`.

Le but de cet exercice est de programmer plusieurs méthodes permettant de trouver un saut de valeur maximale dans un tableau de réels.

Par exemple, dans le tableau :

```
t={2.0, 0.2, 3.0, 5.3, 2.0}
```

un saut de valeur maximale est $(1, 3)$, avec

$$t[3] - t[1] = 5.3 - 0.2 = 5.1.$$

Représentation en C

On utilisera la structure suivante pour représenter un saut :

```c
typedef struct {
    int i;
    int j;
} Saut;
```

Partie A — Fonctions de base

A.1 Écrire une fonction

```c
double valeur(double tab[], Saut s);
```

qui renvoie la valeur du saut `s` dans le tableau `tab`.

A.2 Donner un exemple de tableau de `double` possédant exactement deux sauts de valeur maximale, et préciser ces deux sauts.

A.3 Montrer, à l'aide d'un contre-exemple, qu'on ne peut pas se contenter de chercher le minimum global et le maximum global du tableau pour trouver un saut de valeur maximale.

Partie B — Méthode naïve

B.1 Écrire une fonction

```c
Saut saut_max_naif(double tab[], int n);
```

qui teste tous les couples (i, j) tels que $0 \leq i \leq j < n$, puis renvoie un saut de valeur maximale.

B.2 Déterminer la complexité de cette méthode dans le pire cas.

Partie C — Programmation dynamique

On souhaite maintenant concevoir un algorithme plus efficace.
Pour chaque entier k compris entre 1 et n, on note :

— m_k : l'indice d'un minimum dans le sous-tableau `tab[0..k-1]` ;

— (i_k, j_k) : un saut de valeur maximale dans le sous-tableau `tab[0..k-1]`.

Pour $k = 1$, on a nécessairement :

$$m_1 = 0, \qquad i_1 = 0, \qquad j_1 = 0.$$

C.1 Pour $k < n$, expliquer comment calculer efficacement m_{k+1} à partir de m_k et de la valeur `tab[k]`.

C.2 Justifier que, pour calculer un saut maximal sur `tab[0..k]`, il suffit de comparer :

— le meilleur saut déjà connu sur `tab[0..k-1]` ;

— le saut (m_k, k).

C.3 Écrire une fonction

```
Saut saut_max_dynamique(double tab[], int n);
```

qui renvoie un saut de valeur maximale en utilisant cette méthode.

C.4 Déterminer la complexité de cette méthode dans le pire cas, puis la comparer à celle de la méthode naïve.

Partie D — Méthode « diviser pour régner »

On s'intéresse enfin à une méthode de type diviser pour régner.
On considère un tableau de `double tab` de taille $n \geq 2$.
On note $\lfloor x \rfloor$ la partie entière d'un réel x.
Pour un tableau de taille n, on souhaite déterminer un saut de valeur maximale en distinguant les trois cas suivants :

— (i_g, j_g) un saut de valeur maximale tel que $j_g \leq \lfloor n/2 \rfloor$;

— (i_d, j_d) un saut de valeur maximale tel que $i_d > \lfloor n/2 \rfloor$;

— (i_m, j_m) un saut de valeur maximale tel que $i_m \leq \lfloor n/2 \rfloor < j_m$.

D.1 Justifier qu'un saut de valeur maximale du tableau est nécessairement l'un des trois cas ci-dessus.

D.2 Montrer que, pour un saut maximal (i_m, j_m) tel que $i_m \leq \lfloor n/2 \rfloor < j_m$, l'indice i_m est nécessairement l'indice d'une valeur minimale dans la sous-partie gauche `tab[0...`$\lfloor n/2 \rfloor$`]`.
On admettra de même que j_m est nécessairement l'indice d'une valeur maximale dans la sous-partie droite `tab[`$\lfloor n/2 \rfloor + 1$`...n]`.

D.3 On considère maintenant un sous-tableau `tab[a...b]` avec $0 \leq a \leq b < n$.
Écrire une fonction récursive

```
Resultat saut_max_aux(double tab[], int a, int b);
```

qui renvoie un quadruplet constitué de :

— un saut de valeur maximale (i, j) dans `tab[a...b]` ;

— l'indice d'un minimum dans `tab[a...b]` ;

— l'indice d'un maximum dans `tab[a...b]`.

On attend du candidat qu'il propose une structure `Resultat`. La fonction devra utiliser une stratégie de type « diviser pour régner », en découpant le tableau en deux sous-tableaux de tailles aussi proches que possible.

D.4 En déduire une fonction

```
Saut saut_max(double tab[], int n);
```

qui renvoie un saut de valeur maximale du tableau `tab`.