J1MI2013: Algorithmes et Programmes: feuille 7

Tableaux(4)

Travaux pratiques

Récupérer depuis le site de l'UE l'archive `tp07.tar.gz` et désarchivez-le. Placez vous ensuite dans le répértoire `tp07`.

Exercice 1. Compléter le fichier `testFusionTableauxTries.c` avec la fonction `fusion` de l'exercice 2 de la feuille de TD 7. Compiler et exécuter.

Exercice 2. Dans cet exercice on veut écrire une fonction pour *fusionner* les éléments de deux tableaux `t` et `s` triés dans l'ordre croissant non pas dans un troisième tableau mais dans le tableau `t` lui même (on supposera que la taille de `t` est suffisante).

Exemple : si `t=[4,5,5,8,9]` et `s=[1,3,5]`, suite à l'appel de cette fonction on aura `t=[1,3,4,5,5,5,8,9]`.

La fonction n'utilisera pas de tableau auxiliaire et **ne devra pas déplacer plus d'une fois** chacun des éléments des deux tableaux.

- Si on adopte l'algorithme de fusion de l'exercice précédent on commence par comparer le premier élément de `t` avec le premier élément de `s`. Sur l'exemple ci-dessus il faudra placer le premier élément de `s` à sa place définitive dans `t`. Est-ce que ce choix d'algorithme permet de respecter la consigne de déplacer au maximum une seule fois chacun des éléments de `t` ou de `s` ?
- Si l'on veut minimiser le nombre de déplacements d'éléments de `t`, quel est le couple d'éléments (l'un dans `t` et l'autre dans `s`) qu'on a intérêt à comparer en premier ?
- À quel indice se trouvera à la fin le plus grand parmi les éléments de `t` et `s` ?
- Lorsqu'on ne pourra plus comparer des couples d'éléments, puisque tous les éléments de l'un des deux tableaux seront à leur place définitive, comment procéder avec les éléments qui n'auront pas encore été traités ?

Testez vos hypothèses sur le couple de tableaux donné en exemple ci-dessus, puis sur le couple `t=[1,2,5,8]` et `s=[4,5,7,10,13]` (rappel : la fusion se fait dans le tableau `t`).

Dans le fichier `testInsertionTabTrie.c` écrire une fonction :
`int insertionTableauTrieDansTableauTrie(int t[], int nt, int s[], int ns)` qui insère les `ns` éléments d'un tableau `s` trié en ordre croissant dans un autre tableau `t` contenant `nt` éléments triées en ordre croissant. La fonction retournera la nouvelle taille du tableau `t`

Exercice 3. Dans cet exercice on veut écrire une fonction qui calcule dans un tableau de taille $n+1$ la n-ième ligne du triangle de Pascal ($n \geq 0$).
Par exemple pour $n = 4$ le tableau devra contenir `1 4 6 4 1` .

On rappelle que le triangle de Pascal pour $0 \leq n \leq 4$ est :

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Les éléments du triangle sont les coéfficients binomiaux C_n^p où n représente la ligne et p la colonne ($0 \leq p \leq n$) dans le triangle.

Pour construire le triangle on pose $C_0^0 = 1$ et ensuite, pour $n > 0$ et $1 \leq p \leq n-1$, on a $C_n^p = C_{n-1}^p + C_{n-1}^{p-1}$, c'est à dire que l'élément en ligne n et colonne p se calcule à partir des éléments dans les colonnes p et $p-1$ de la ligne $n-1$.

La fonction à écrire **ne devra pas utiliser d'autres variables que le tableau** passé en paramètre, il faut donc trouver le moyen de calculer chaque élément de la `n`-ième ligne en utilisant les valeurs (qui représentent la `(n-1)`-ième ligne) déjà présentes dans le tableau (et calculées à partir des valeurs représentant la `(n-2)`-ième ligne....).
La fonction n'effectuera aucune multiplication (la seule opération arithmétique nécessaire est l'addition).

- Avec quelle valeur initialiser toutes les cases du tableau ?
- Quelle est la première case de la `n`-ième ligne que l'on peut calculer sans écraser des valeurs qui seront nécessaires aux calculs suivants ?

Dans le fichier `testLigneTrianglePascal.c` écrire la fonction :
`void ligneTrianglePascal(int t[], int n)` qui prend en paramètre un tableau `t` de taille `n+1` et calcule dans `t` la `n`-ième ligne du triangle de Pascal.