Tri par insertion [tr05] - Exercice

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 21 mai 2018

Table des matières

C++ - Tri par insertion (TP)

Mots-Clés Algorithmes de tris et rangs, Tri par insertion ■
Requis Axiomatique impérative (sauf Fichiers) ■
Difficulté • ○ ○ (45 min) ■

Objectif
Cet exercice réalise le tri par insertion d'un tableau d'entiers. Dans le même ordre d'idées, l'exercice @[Tri bulle et associés] réalise le tri bulle et l'exercice @[Tri par sélection] celui par sélection.

1 Téléchargements – Utilitaires

Cet exercice utilise les opérations suivantes, toutes définies dans un bon nombre d'exercices de cet espace thématique :

- Fonction `saisirNombreElements`
- Procédure `afficherTri`
- Procédure `aleatoireTri`
- Procédure `permuterTab`

Elles ont été regroupées dans une bibliothèque.

Définitions C++

```
const int TMAX = ...;
using ITableau = int[TMAX];
```

Fixez la constante `TMAX=50` (nombre maximum d'éléments).

Téléchargez le fichier suivant et mettez-le dans votre dossier.

C++ @[UtilsTR.cpp]

Copiez/collez ensuite les lignes suivantes :

C++ **Au début** de votre programme :

```
#include "UtilsTR.cpp"
using namespace UtilsTR;
```

Soit la fonction `saisirNombreElements(nmax)` qui renvoie le nombre d'éléments, saisi par l'utilisateur, entier compris dans `[1..nmax]`. Elle affiche l'invite :

```
Nombre d'éléments dans [1..[nmax]]?
```

C++ @[saisirNombreElements] (dans UtilsTR.cpp)

Soit la procédure `afficherTri(t,n,g,h)` qui affiche, à la queue-leu-leu séparés par un espace le tout entre crochet, les `n` premières valeurs d'un `ITableau t`, les indices `g` et `h` indiquant le sous-intervalle du tri et représentés par une barre verticale. La barre de gauche est avant `g` et celle de droite est après `h`. Exemple :

```
afficherTri(t,10,4,9) ==> [1 2 3 |4 5 6 7 8 9 |10]
```

C++ @[afficherTri] (dans UtilsTR.cpp)

Soit la procédure `aleatoireTri(t,n,vmax)`H qui initialise les `n` premiers éléments d'un `ITableau t` en utilisant `vmax` comme valeur maximale pour la fonction de génération d'un entier pseudo-aléatoire.

C++ @[aleatoireTri] (dans UtilsTR.cpp)

Soit la procédure `permuterTab(t,j,k)` qui permute les éléments d'indice `j` et `k` d'un `ITableau t`. Les indices sont supposés valides.

C++ @[permuterTab] (dans UtilsTR.cpp)

Remarque
Si la fonction et les procédures n'ont pas été réalisées, il vous est conseillé de la(les) rédiger dans l'exercice @[Utilitaires Tris et Rangs].

...(suite page suivante)...

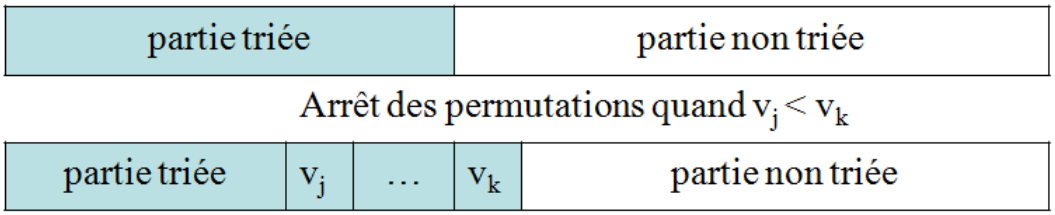
2 Tri par insertion / pginsertion

2.1 Tri par insertion basique

Principe du tri par insertion

Le **tri par insertion** d'un tableau `t[1..n]` de `n` éléments consiste, pour `j` variant de 1 à `n`, à insérer l'élément `t[j]` dans le sous-tableau trié `t[1..j-1]` de sorte que le tableau `t[1..j]` soit trié.

Etape j :



Exemple

(Avec affichage des tableaux successifs)

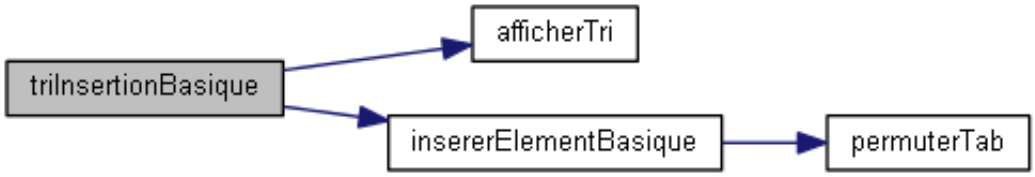
```
Nombre d'éléments dans [1..50]? 10
Tableau initial
[ 13 12  6  7 13 18 17  4 18  3  ]
Tri par insertion basique
[ 13|12  6  7 13 18 17  4 18  3 |]
[ 12 13| 6  7 13 18 17  4 18  3 |]
[  6 12 13| 7 13 18 17  4 18  3 |]
[  6  7 12 13|13 18 17  4 18  3 |]
[  6  7 12 13 13|18 17  4 18  3 |]
[  6  7 12 13 13 18|17  4 18  3 |]
[  6  7 12 13 13 17 18| 4 18  3 |]
[  4  6  7 12 13 13 17 18|18  3 |]
[  4  6  7 12 13 13 17 18 18| 3 |]
[  3  4  6  7 12 13 13 17 18 18||]
Tableau final
[  3  4  6  7 12 13 13 17 18 18  ]
```

Remarque

Le tri par insertion est similaire au tri du joueur de cartes qui insère les cartes une à une dans son jeu de manière à ce qu'à tout instant le jeu soit trié.

Écrivez une procédure `insererElementBasique(t,k)` qui insère l'élément `t[k]` dans le sous-tableau trié `t[1..k-1]`, `t` étant un `ITableau`. La procédure utilisera la procédure `permuterTab` pour réaliser l'échange de deux éléments de `t`.

Écrivez une procédure `triInsertionBasique(t,n)` qui effectue le tri par insertion de `n` éléments d'un `Tableau t`.



Écrivez un programme qui saisit le nombre d'éléments, déclare un `ITableau2` et l'initialise de façon aléatoire en prenant 20 pour valeur maximale puis effectue son tri et l'affiche.

Testez.

2.2 Tri par insertion, version 2

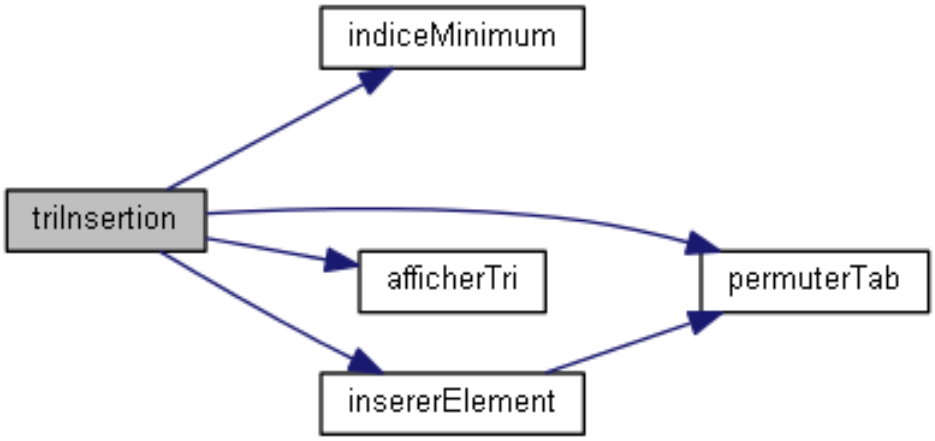
Principe de la sentinelle

Afin d'éviter le débordement vers la gauche lors de l'insertion de l'élément `t[k]` dans le sous-tableau trié `t[1..k-1]`, une autre solution est de mettre une sentinelle en `t[1]`, c.-à-d. un élément dont on est certain qu'il va arrêter la boucle. Il suffit pour cela qu'il soit inférieur à tout élément rencontré, ce qui est le cas si l'on recherche au préalable l'élément minimum et qu'on le met à cette place.

Écrivez une fonction `indiceMinTab(t,n)` qui calcule et renvoie l'**indice** de l'élément contenant la plus petite valeur parmi les `n` premières valeurs d'un `ITableau t`. En cas d'ex-aequo, c'est l'indice le plus petit qui sera renvoyé.

Copiez/collez la procédure `insererELementBasique` en `insererElement(t,k)`, puis modifiez la procédure de sorte qu'elle insère l'élément `t[k]` dans le sous-tableau trié `t[1..k-1]`, l'élément minimal du tableau étant en `t[1]`. La procédure utilisera `permuterTab` pour réaliser l'échange de deux éléments de `t`.

Copiez/collez la procédure `triInsertionBasique` en `triInsertion(t,n)`, puis modifiez la procédure de sorte qu'elle recherche tout d'abord l'élément minimum des `n` éléments de `t` et le place en `t[1]` afin qu'il joue le rôle de sentinelle puis effectue la procédure d'insertion à partir de l'élément 2.



Modifiez votre programme afin qu'il effectue le tri par insertion (version 2).

Testez.