TP : Tris de tableaux

Dans ce TP vous devez utiliser le paquetage `paq_permutation` écrit lors des précédents TP. Vous pouvez également charger et utiliser le paquetage compilé fourni si le votre ne fonctionne pas bien. Dans ce cas il vous faudra supprimer les droits en écriture sur les fichiers `paq_permutation.*` par exemple avec la commande `chmod u-w paq_permutation.*`.

L'objectif de ce TP est de (re)programmer quelques algorithmes de tri classiques et d'étudier leur complexité analytiquement et expérimentalement.

Tous les algorithmes de tri seront placés dans le paquetage `paq_tri`.

1 Préparation

Pour simplifier l'étude expérimentale de la complexité des tris vous allez encapsuler les deux opérations élémentaires de comparaison d'éléments et d'échange de position d'éléments dans deux procédures que vous utiliserez lors de la programmation des tris.

Q1 . Programmez la fonction `comp` prenant en paramètre un tableau `t` de type `PERMUTATION` et deux indices `x` et `y` de ce tableau qui retourne `true` si `t(x)` $\leq$ `t(y)` et `false` sinon.

Q2 . Programmez la fonction `comp_qs` (à utiliser pour le tri rapide ou Quick Sort) prenant en paramètre un tableau `t` de type `PERMUTATION` et deux indices `x` et `y` de ce tableau qui retourne `true` si `t(x)` $\leq$ `y` et `false` sinon.

Q3 . Programmez la procédure `echanger` prenant en paramètre un tableau `t` de type `PERMUTATION` et deux indices `x` et `y` de ce tableau qui échange les positions des valeurs `t(x)` et `t(y)`.

2 Tri à bulles

Le tri à bulles est le tri le plus naturel, son principe est très simple : on parcours le tableau du début jusqu'à la fin et chaque fois qu'on rencontre un indice i pour lequel $t(i)$ est supérieur à $t(i+1)$ on intervertit ces deux éléments. On recommence le parcours tant que le tableau n'est pas trié.

Q4 . Programmez l'algorithme du tri à bulle sur des tableaux de type `PERMUTATION`.

Q5 . Dans un algorithme, un invariant est une propriété qui, comme son nom l'indique, reste vraie du début jusqu'à la fin. Par exemple pour le traditionnel algorithme itératif de calcul de la somme des n premiers entiers, à chaque début d'itération k l'équation $\sum_{i=1..n} = \sum_{i=k..n} + S$, où S est la somme partielle déjà calculée par l'algorithme. Un tel invariant est un élément central des preuves formelles de correction d'algorithme.

En observant le tableau à chaque itération donnez un invariant pour l'algorithme du tri à bulles et en déduire une preuve (assez formelle) qu'il s'agit bien d'un algorithme de tri.

Q6 . Identifiez un pire et un meilleur cas pour votre tri à bulles pour le nombre de comparaisons effectuées.

Q7 . D'après le meilleur cas trouvé pour votre algorithme est-il possible de l'améliorer ?

Q8 . Combien de comparaisons entre éléments du tableau sont effectuées dans le pire et le meilleur cas pour trier un tableau de taille n ?

Q9 . Donnez l'ordre de grandeur asymptotique du nombre de comparaisons effectuées dans le pire et le meilleur cas.

3 Tri par sélection

Le principe de l'algorithme du tri par sélection consiste à construire petit à petit une partie de tableau triée de taille croissante en sélectionant à chaque étape le plus petit élément de la partie non triée puis en l'échangeant avec le premier élément de la partie non triée. On peut également choisir de s'intéresser plutôt au plus grand élément et de construire une partie triée en fin de tableau.

Q10 . Représentez par un schéma le tableau au début d'une itération en y faisant figurer la partie triée, la partie non triée et le passage à l'itération suivante par l'ajout d'un élément de la partie non triée à la partie triée.

Q11 . Programmez l'algorithme du tri par sélection sur des tableaux de type `PERMUTATION`.

Q12 . Prouvez que cet algorithme est bien un algorithme de tri. Vous pouvez commencer par identifier un invariant de l'algorithme.

Q13 . Identifiez un pire et un meilleur cas pour le nombre de comparaisons effectuées par le tri par sélection.

Q14 . Combien de comparaisons entre éléments du tableau sont effectuées dans le pire et le meilleur cas pour trier un tableau de taille n ?

Q15 . Donnez l'ordre de grandeur asymptotique du nombre de comparaisons effectuées dans le pire et le meilleur cas.

4 Tri par insertion

Comme pour le tri par sélection, au cours du tri par insertion une partie du tableau est triée et les éléments non encore triés sont ajoutés un par un. La différence entre les deux réside dans la manière d'ajouter un nouvel élément à la partie triée. Dans le tri par insertion c'est l'élément non trié de plus petit indice et pas celui de valeur minimum qui est ajouté. Il est inséré à sa place dans la partie triée ce qui peut nécessiter de décaler une partie des éléments triés pour lui faire une place. Ils existe plusieurs manières de procéder tant pour la recherche de la bonne place que pour procéder au décalage, dans ce TP vous échangerez l'élément x à inserer avec son voisin de gauche, et cela autant de fois que nécessaire jusqu'à ce que ce x soit "à sa place".

Q16 . Programmez l'algorithme du tri par insertion sur des tableaux de type `PERMUTATION`.

Q17 . Identifiez un pire et un meilleur cas en nombre de comparaisons entre éléments du tableau du tri par insertion.

Q18 . Combien de comparaisons entre éléments du tableau sont effectuées dans le pire et le meilleur cas pour trier un tableau de taille n avec chacune des versions ?

Q19 . Donnez l'ordre de grandeur asymptotique du nombre de comparaisons effectuées dans le pire et le meilleur cas.

5 Tri par fusion

Le tri par fusion est une bonne illustration du paradigme diviser pour régner. Il s'agit d'un algorithme récursif qui procède en divisant le problème en plusieurs sous problèmes indépendants et plus simples à traiter, souvent car leurs données sont de tailles réduites. Chacun des sous problèmes est ensuite résolu. Enfin leurs solutions sont recombinées pour obtenir la solution du problème initial.

Pour le tri par fusion les trois grandes étapes sont les suivantes :

Diviser Il s'agit de couper le tableau en deux moitiés de tailles à peu près égales, ce qui revient simplement à identifier l'indice du milieu du tableau `indice`.

Régner Les deux sous-tableaux `t(t'first..indice)` et `t(indice+1..t'last)` sont triés par appel récursif à la procédure de tri par fusion.

Combiner C'est la partie la plus importante du tri par fusion, en effet les deux sous-tableaux `t(t'first..indice)` et `t(indice+1..t'last)` sont triés, mais celà ne veut pas dire que le tableau `t` l'est également. Le tri fait appel à une sous procédure `fusion` qui dans un premier temps recopie au moins un des deux sous tableaux dans un nouveau tableau (ou les deux), puis re-rempli `t` en prenant dans les copies des sous tableaux les éléments par ordre croissant.

Q20 . Programmez l'algorithme du tri par fusion sur des tableaux de type `PERMUTATION`.

Q21 . Identifiez le meilleur et le pire des cas pour le tri par fusion en nombre de comparaisons entre éléments effectuées.

Q22 . Quelle est la complexité en nombre de comparaisons pour les pires et meilleurs cas ?

6 Tri rapide

Le tri rapide (ou encore quicksort) est fondé comme le tri fusion sur le paradigme diviser pour régner. C'est également un algorithme récursif qui procède en divisant le tableau en deux sous-tableaux. La différence est la définition des sous-tableaux.

Voici un descriptif des trois étapes employées pour le tri d'un tableau `t` :

Diviser On choisit arbitrairement un élément de `t` que l'on appelle *pivot*. On partitionne le tableau `t` en deux : à gauche on dispose tous les éléments de `t` plus petits que le pivot et à droite tous les éléments de `t` plus grands que le pivot. Ainsi, si `indice` est l'indice de séparation, tous les éléments de `t(t'first..indice)` sont plus petits que ceux de `t(indice+1..t'last)`, et le pivot doit être placé à l'indice `indice`.

Régner Les deux sous-tableaux `t(t'first..indice)` et `t(indice+1..t'last)` sont triés par appel récursif à la procédure de tri rapide.

Combiner Comme les sous-tableaux `t(t'first..indice)` et `t(indice+1..t'last)` sont triés sur place, il n'y a aucun travail de fusion à faire, le tableau `t` est trié.

Q23 . Intégrez le code à compléter de la procédure `tri_rapide` à votre paquetage contenant déjà les autres procédures de tri.

La procédure `tri_rapide` comporte une fonction `pivot` qui choisit un pivot et une procédure `partionner` (à écrire) qui effectue le travail de partitionnement d'un tableau t en deux et retourne l'indice qui sépare le tableau par rapport au pivot. Celle-ci réorganise le tableau `t` de manière telle que tous les éléments jusqu'à la position `indice` dans `t` sont plus petits que ceux de position supérieure à `indice`.

Attention, le pivot retourné par la fonction `pivot` est une valeur du tableau et pas un indice du tableau ! ! !

Vous devez ensuite compléter le code de la procédure `tri_rapide` afin que celle-ci procède au tri en suivant la méthode de tri donnée ci-dessus.

Q24 . Identifiez le meilleur et le pire des cas pour le tri rapide en nombre de comparaisons entre éléments effectuées.

Q25 . Quelle est la complexité en nombre de comparaisons dans chacun de ces deux cas ? Donnez une intuition pour la complexité en moyenne.

7 Etude expérimentale de la complexité des tris

Pour mesurer expérimentalement la complexité des tris vous allez ajouter à toutes les procédures de tri des compteurs donnant le nombre de comparaisons entre éléments ainsi que des compteurs donnant le nombre d'échanges entre éléments qui sera évalué aussi.

Pour cela vous devez simplement initialiser ces compteurs *au bon moment* puis relever leurs valeurs également *au bon moment*, sans oublier de les incrémenter dans les fonction et procédure `comp` et `echanger`.

Vous allez ensuite utiliser le programme `gnuplot` pour tracer les complexités des différents algorithmes en fonction des tailles des données.

Q26 . Ajoutez les compteurs nécessaires en variables globales du paquetage de tri et modifiez le code de `comp` et `echanger` pour les y incrémenter.

7.1 Représentations graphiques

En guise de petit exemple pour démarrer, recopiez ces lignes dans un fichier texte `fichier.txt`. Si vous souhaitez ajouter des commentaires, ils débutent par le caractère `#`.

```
1   1  10
2   5  25
3  25  30
```

Chaque ligne représente par exemple (au hasard) une taille de tableau suivie par un nombre de comparaisons et un nombre d'échanges.

Lancez `gnuplot` puis tapez :

```
plot 'fichier.txt' using 1:2 title 'comparaisons' with lines
```

La commande `plot` est la commande de production de graphique, `fichier.txt` est le fichier où se trouvent les données à afficher, `using 1:2` indique que les points servant à construire le graphique ont leur abscisse dans la première colonne et leur ordonnée dans la deuxième et enfin, `with lines` précise que le tracé doit être continu.

Pour superposer deux tracés sur un même graphique il suffit de séparer par une virgule leurs descriptions respectives directement dans la commande comme suit :

```
plot 'fichier.txt' using 1:2 title 'comparaisons' with lines, \
'fichier.txt' using 1:3 title 'echanges' with lines
```

Notez que le caractère \ vous permet de poursuivre la commande à la ligne suivante.

Pour sauvegarder les schémas il suffit de préciser avant de lancer la commande `plot` le format de sauvegarde de l'image (png, eps, etc) et le nom du fichier à créer, ce qui donne :

```
>set term png
>set output 'graphique.png'
>plot 'fichier.txt' using 1:2 title 'comparaisons' with lines, \
'fichier.txt' using 1:3 title 'echanges' with lines
```

Vous pouvez ensuite admirer les résultats de vos mesures dans le fichier `graphique.png` qui vient d'être crée.

Question 4 Reprenez les questions du TP précédent afin de dessiner les courbes de complexité moyenne pour des tableaux dont la taille varie de 1 à 10. Dans un deuxième temps, pour des tailles de tableau $n = 11, 12, 13, 14, 15, 25, 50, 100, 250, 500, 1000, 5000$ réaliser $p = n$ tirages aléatoires [1].

1. Il n'est pas possible de calculer dans un temps raisonnable pour toutes les permutations pour n grand.

7.2 Complexité en moyenne

Des valeurs exactes

Pour établir expérimentalement la valeur exacte du nombre moyen d'opérations, on peut engendrer les $n!$ tableaux possibles contenant n éléments distincts, évaluer le nombre d'opérations effectuées lors du tri de chacun, et faire la moyenne arithmétique.

Q27 . Programmez une procédure par tri qui traitera tous les tableaux de taille n et calculera les nombres moyens de comparaisons et d'échanges de manière exacte et l'affichera sous la forme `<taille de donnée> <comparaisons> <échanges>`.

Q28 . Pour chaque tri, afficher le nombre moyen d'opérations nécessaires au tri d'une permutation de taille $n = 2, 3, \ldots, 10$. Vous redirigerez l'affichage dans un fichier de données qui sera ensuite utilisé par gnuplot.

Des estimations

Malheureusement $n! = \Omega(e^n)$: en pratique, trier tous les tableaux de taille n devient rapidement impossible. Pour des valeurs de n plus grandes, il faut se contenter de valeurs établies au vu d'échantillons. Vous calculerez donc des valeurs approchées du nombre moyen d'opérations en considérant n tableaux de taille n choisis aléatoirement.

Q29 . Programmez une procédure par tri qui traitera tous les tableaux de taille n et calculera les nombres moyens de comparaisons et d'échanges de manière approchée.

Q30 . Pour chaque tri, afficher le nombre moyen d'opérations nécessaires au tri d'une permutation de taille $n < 15$, $n = 25, 50, 100, 250, 500, 1000, 5000$. Vous redirigerez l'affichage dans un fichier de données qui sera ensuite utilisé par gnuplot.

Tracé des courbes

Q31 . Tracez les courbes donnant le nombre moyen de comparaisons d'une part et le nombre moyen d'échanges d'autres part sur deux graphiques rassemblant tous les tris.

Q32 . Que pouvez vous dire à propos du tri rapide et du tri par fusion l'un par rapport à l'autre ?

7.3 Pire et meilleurs cas

Vous pouvez vérifiez les complexités pire et meilleur cas que vous avez calculées dans les parties précédentes en traçant les courbes correspondantes. Pour cela vous ne devez donner en entrée des algorithmes que des permutations représentant les pires ou les meilleurs cas pour chaque taille de tableau.