Université de Montpellier 2019/2020
ULIN405 Durée: 20h

TER L3
TD – Séance n

Encadrant : rodolphe Giroudeau ; rgirou@lirmm.fr

Exercice 1 – Tri par insertion sur des petits tableaux pendant le tri par fusion

Bien que le tri par fusion s'exécute en $\theta(n \log n)$ dans le pire des cas et que le tri par insertion s'exécute en $\theta(n^2)$, les facteurs constants du tri par insertion le rendent plus rapide pour des n petits. Il est donc raisonnable d'utiliser le tri par insertion à l'intérieur du tri fusion lorsque que les sous-problèmes deviennent suffisamment petits. On considère une modification du tri par fusion pour laquelle n/k sous-listes de longueur k sont triées à l'aide du tri par insertion, puis fusionnées à l'aide du fusion classique, où k est une valeur à déterminer.

1. Montrer que les n/k sous-listes, chacune de longueur k, peut être triée grâce au tri par insertion avec un temps en $\theta(nk)$ dans le pire des cas.
2. Montrer que les sous-listes peuvent être fusionnées en $\theta(n \log(n/k))$ dans le pire des cas.
3. Sachant que l'algorithme modifié s'exécute en $\theta(nk + \lg(n/k))$ dans le pire des cas, quelle est la plus grande valeur asymptotique (en notation θ) de k en fonction de n pour laquelle l'algorithme modifié possède le même temps d'exécution asymptotique que le tri par fusion classique ?
4. Comment doit-on choisir k en pratique ?

Exercice 2 – Inversions

Soit $A[1 \dots n]$ un tableau de n nombres distincts. Si $i < j$ et $A[i] > A[j]$; on dit que le couple (i, j) est une **inversion** de A.

1. Donner les cinq inversions du tableau $2, 3, 8, 6, 1$.
2. Quel est le tableau dont les éléments appartiennent à l'ensemble $\{1, 2, \dots, n\}$ ayant le plus d'inversion ? Combien en possède-t-'il ?
3. Quelle est la relation entre les temps d'exécution du tri par insertion et le nombre d'inversion du tableau d'entrée ? Justifier votre réponse.
4. Donner un algorithme qui détermine le nombre des inversions présentes dans une permutation quelconque de n éléments en un temps $\theta(n \log n)$ dans le pire des cas (indication : modifier le tri fusion).

Exercice 3 – Le drapeau Hollandais et le tri rapide

Soit T un tableau d'éléments d'un ensemble E et une fonction couleur de $E \to \{blanc, rouge\}$. On peut déplacer les éléments de E et leur appliquer la fonction couleur. Le but est de mettre tous les éléments blanc dans les premières cases et les éléments rouge dans les dernières (initialement les éléments sont dans un ordre quelconque dans un tableau T).

1. Donner un algorithme (en vous inspirant de la partition du tri rapide) qui permette sans utiliser de tableau annexe de résoudre le problème. On ne regardera qu'une fois au plus la couleur de chaque élément pour effectuer cette opération.

2. En supposant qu'initialement la probabilité que $couleur(T[i])$ soit $rouge$ est égale à $1/2$ pour tout i, calculer le nombre moyen d'échanges effectués par l'algorithme. Programmer l'algorithme et vérifier le résultat précédent en l'exécutant sur des données générées aléatoirement.
3. Répondre aux mêmes questions en supposant que la couleur peut prendre trois valeurs $\{bleu, blanc, rouge\}$. Evidemment dans ce cas on supposera aussi que la probabilité que $couleur(T[i])$ soit $rouge$ (resp. $blanc$ ou $bleu$) est égale à $1/3$ pour tout i .
4. Vous avez vu que le tri rapide est en $O(n \log n)$ en moyenne. En fait ce résultat n'est valide que si tous les éléments sont différents. Que peut-on dire si de nombreux éléments peuvent être égaux ? (par exemple s'ils sont tous égaux).
5. En vous aidant de la question 3 montrez que l'on peut adapter le tri rapide pour obtenir à nouveau un tri efficace. Expliquer l'algorithme.
6. Comparer les deux versions de tris (à partir d'exécutions) lorsque de nombreux éléments sont égaux.

Exercice 4 – Arbres binaires : Application au tri par tas

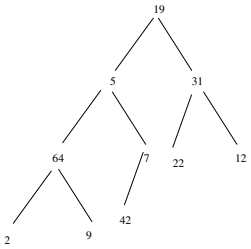

FIGURE 1 – Arbre binaire

1. Un arbre parfait est un arbre binaire dont tous les niveaux sont complétement remplis sauf éventuellement le dernier niveau, et dans ce cas les sommets (feuilles) du dernier niveau sont groupés le plus à gauche possible.
   (a) Comment peut-on représenter efficacement un arbre binaire parfait par un tableau et un entier (représentant la taille en nombre d'éléments du tas) ? Donner le tableau représentant l'arbre binaire proposé à la figure 1.
   (b) Donner les fonctions permettant de savoir si un sommet est une feuille, de connaître le père d'un sommet, le fils gauche, le fils droit.
   (c) Calculer la hauteur d'un arbre binaire parfait.
2. Un tas est un arbre binaire parfait tel que chaque élément d'un sommet est inférieur ou égal aux éléments de ses fils.
   (a) Donner les fonctions permettant d'obtenir le minimum d'un tas,
   (b) de supprimer le minimum d'un tas,
   (c) d'ajouter un élément à un tas.
   (d) Evaluer la complexité de chacune de ces fonctions
3. Donnez une fonction qui structure un tableau quelconque en tas en ajoutant successivement les éléments au tas déjà construit (initialement le tas est vide, la partie gauche du tableau est structurée en tas, les éléments du tableau sont ajoutés un à un de la gauche vers la droite). Evaluer la complexité de cette construction.

4. Même question mais en construisant le tas de la droite vers la gauche. On considère que les feuilles sont des tas et on fusionne les tas en ajoutant les sommets internes. Evaluer la complexité de cette construction.
5. En déduire un algorithme de tri par ordre décroissant d'un tableau (tri par tas). Evaluer sa complexité.
6. Adapter-le pour trouver le k^{ieme} plus petit élément d'un tableau. Evaluer la complexité de recherche de cet élément.
7. **Partie test**
   (a) Vous procéderez à une variation de la taille du tableau et à génération aléatoire des éléments. Les occurrences des éléments sont au départ supérieurs ou égaux à un.
   (b) Pour chaque tri (tri par tas et tri rapide) vous calculerez le nombre de comparaisons et le nombre d'affectations et également le temps d'exécution. Vous comparerez les résultats obtenus pour les deux tris. Vous devez produire des courbes et les commenterez.
   (c) Vous procéderez à des exécutions sur des tableaux pathologiques (tableaux quasi triés, sous-tableaux déjà triés, ...)
   (d) Question subsidiaire. Maintenant l'occurrence des éléments est égale à un. Refaire des tests et commenter.

Compétences souhaitées : un goût pour l'algorithmique, programmation

1 Objectifs du projet et travail à faire

1.1 Programme à écrire

1.1.1 Présentation des algorithmes, des structures de données, du programme et des fonctions

Le *cahier de programmation* que vous devrez rendre devra obligatoirement comporter les éléments suivants.

Algorithme en pseudo code
- Une description et une justification précise (par des arguments d'efficacité, de lisibilité, etc) des structures de données que vous utilisez.
- Une étude bibliographique des problèmes
- Une évaluation aussi précise que possible de la complexité en temps et en mémoire de votre algorithme.

Programme
- Un mode d'emploi de votre application.
- Le listing complet et commenté de votre programme.
- Une traduction en C des structures de données que vous utilisez.
- La liste complète de toutes vos fonctions avec, pour chacune d'elle, les éléments suivants.
  - L'entête complète de la fonction telle qu'elle apparaît dans le listing.
  - La description des paramètres d'entrée avec leurs noms et significations.
  - La description du *résultat* de cette fonction ainsi que la description des paramètres d'entrée qui peuvent être modifiés pendant son exécution.

— Une description sommaire de ce que fait cette fonction.
— Un graphe de dépendance des fonctions entre elles. Chaque fonction devra être représentée dans ce graphe. Vous placerez un arc d'une fonction f_1 vers une fonction f_2 si f_1 fait appel à f_2 dans son code.
— Des jeux de tests de votre programme.

1.2 Document à rendre : contenu forme et date

Vous devrez rendre un compte rendu de votre travail sous la forme d'un rapport. Celui-ci devra obligatoirement contenir les éléments suivants.

— Une introduction présentant le problème.
— Une présentation des algorithmes
— Les divers documents demandés à la section 1.1.1.
— Une conclusion synthétisant le travail fait ainsi que les difficultés rencontrées et les extensions possibles.

Toute journée de retard entraîne un point (sur 20) de pénalité sur la note globale de projet. Notez qu'une soutenance orale des projets sera organisée plus tard. Chaque membre du groupe devra *obligatoirement participer* à cette soutenance.

2 Conseils pour le rapport

1. Un code est lisible, ce qui prend plusieurs aspects : il est commenté, il est structuré, il est autant que possible simple (peu d'astuces, des choix d'identificateurs explicites, pas trop de niveaux d'imbrications, ...). En particulier toute astuce, tout choix algorithmique sophistiqué doit être justifié. Un code doit être auto-suffisant, et doit pouvoir être maintenu, remanié par une autre personne que la personne l'ayant rédigé (en particulier, par l'étudiant en binôme). Qui plus est, il doit être lisible par une personne non experte du langage de programmation.
2. Un jeu de tests (commenté) met en lumière les cas nominaux où cela marche, les cas particuliers, les cas aux limites, et éventuellement les cas de figure pour lesquels une solution ad hoc a été adoptée, pour ne pas dire, un traitement d'exception expéditif. Un jeu de tests est autant que possible complet : cela signifie que toutes les fonctionnalités importantes sont testées, qu'une stratégie de test a été appliquée pour choisir les tests, ...
3. Un rapport synthétise et justifie les choix de conception effectués lors du projet. Il peut s'agir des choix de structures de données, des choix algorithmiques, des choix d'organisation logicielle (si le projet est découpé en plusieurs grandes fonctionnalités), des choix de répartition de travail dans le binôme, et surtout des choix relatifs à l'élargissement ou à la restriction du sujet initial (ou tout au moins relatifs aux contours du sujet).
4. Autant pour avoir une note correcte, il convient de traiter le sujet comme il a été demandé dans le sujet, autant pour bénéficier d'un petit plus, il peut être bénéfique de compléter le projet soit par une étude de performances comparées, soit par l'ajout de remarques pertinentes quant à l'intérêt du sujet, aux extensions possibles du projet, à la remise en cause d'une solution mise en œuvre, ... Ce point souligne en fait qu'une certaine originalité est souvent appréciée, dès lors que l'essentiel du contrat est rempli.
5. Le rapport doit être synthétique (et donc ne pas dépasser en pratique 20 pages) et dans la mesure du possible écrit en bon français. De plus, il doit être lisible sans faire référence

sans cesse au code : on peut donc y insérer un passage du code que l'on considère comme clé pour étayer le commentaire.

6. Un projet peut recéler des difficultés cachées. Le commencer deux jours avant la date limite est donc un jeu dangereux, même si on est sûr de ses compétences techniques.
7. Il est bien sûr important que le projet réponde au sujet et pas à côté et que bien sûr, cela marche ! ! Ne pas hésitez à demander des compléments d'informations aux encadrants sur les objectifs du projet. Le moment le plus favorable est à la fin d'un cours et sinon sur rendez-vous (venez en groupe pour éviter de demander plusieurs fois la même chose).