Dichotomie et tri rapide

Lors d'un TD précédent, vous avez pu découvrir les algorithmes de tri permettant d'ordonner les éléments d'un tableau. La complexité des trois tris présentés étant quadratique, ils fonctionnent sur un nombre relativement faible d'entrées. L'objectif de ce cours est de découvrir un nouvel algorithme de tri, plus efficace que les précédents, basé sur le principe de la dichotomie. Vous découvrirez également l'utilité d'un tableau trié pour la recherche rapide d'un élément.

1 Introduction à la dichotomie

Le principe de la dichotomie [1], ou diviser pour régner [2] (les termes sont équivalents) permet de résoudre plus rapidement un problème comportant N entrées en le séparant en deux sous-problèmes de taille $N/2$ environ. Le principe général est de découper la résolution globale du problème en trois parties :

- *diviser* le problème général en sous-problèmes ;
- *régner* dans chacun des sous-problèmes, c'est à dire les résoudre ;
- *combiner* les sous-problèmes afin de reformer la solution générale.

Si vous avez regardé « le juste prix [3] », ou si vous avez travaillé au semestre 1 en info, vous connaissez déjà le principe de la dichotomie. Le candidat cherche à mettre un prix sur un ensemble de cadeaux dont la valeur totale est entre 10000 [4] et 50000 Francs. [5]. Il dispose de 30 secondes pour trouver le montant exact de la vitrine. À chaque proposition, le présentateur répond au candidat « c'est plus » ou « c'est moins » en fonction de sa réponse et du montant de la vitrine.

Naturellement, peu de personnes tentent tous les montants en incrémentant le montant précédent de 1 Franc à cause du peu de temps alloué à cet exercice (recherche exhaustive). Étudions en détail cette méthode. Le principe de « couper en deux » divise la taille de l'espace de recherche par deux. Si par exemple la première tentative du candidat consiste à dire 30000 (donc pile entre le minimum et le maximum), sa prochaine tentative réduira le nombre de possibilités de moitié. De nouvelles bornes de recherche sont alors établies en fonction de la réponse du présentateur et le processus reprend jusqu'à trouver le résultat final.

Exercice 1. Recherche dichotomique pour le Juste Prix.

1) Rédigez un algorithme prenant en entrée un entier entre 0 et 50000 et affichant la recherche dichotomique de cet entier.
2) Codez cet algorithme de façon itérative.
3) Codez cet algorithme de façon récursive.

1. « Chotomie ! »
2. « Viser pour régner ! »
3. Le vrai, celui des années 90 avec Philippe Risoli, vous est malheureusement inconnu...
4. « Mille ! »
5. Ça non plus vous ne connaissez pas...

4) Quelle est la complexité d'une telle recherche ?

Nous pouvons généraliser l'exercice précédent à la recherche d'un élément dans un tableau trié. Les entrées de l'algorithme sont alors un tableau trié, sa taille et la valeur de l'élément à rechercher. L'algorithme compare la médiane du tableau et la valeur de l'élément à rechercher. Il réalise alors un appel récursif dans le sous-tableau contenant l'élément recherché (diviser et régner). L'algorithme retourne finalement l'indice de l'élément trouvé ou -1 s'il n'est pas présent dans le tableau (pas besoin de combiner).

2 Tri rapide

Le tri *rapide* porte le nom de *rapide* tout simplement parce qu'il est *rapide*. En effet, sa complexité moyenne est en $O(N\log(N))$ pour trier un tableau comportant N entrées. Cette complexité quasi-linéaire (c'est son petit nom) est à mettre en rapport avec la complexité quadratique des tris vus lors de la séance précédente. L'objectif de cette section est de comprendre le fonctionnement du tri, de le dérouler sur un exemple puis de le programmer.

Exercice 2. Vérification de la compréhension..

À partir de l'exemple présenté dans la vidéo donnée sur Moodle [6], vous devriez être capable de comprendre le fonctionnement général de l'algorithme.

1) Complétez le tableau présenté en annexe. La première ligne présente les cartes dans leur ordre initial. Écrivez l'état du tableau sur une nouvelle ligne dès qu'une permutation est effectuée ou qu'un nouveau pivot est choisi. Soulignez le pivot en cours d'utilisation. Entourez les éléments qui ne bougent plus à partir d'une certaine étape.
2) Résumez en quelques mots (pas plus de 5 lignes) le principe du tri. Si besoin, reformulez la description faite au début de la vidéo.

À partir de l'exemple précédent, nous allons maintenant étudier plus en détail l'algorithme de partitionnement et la réalisation de l'appel récursif.

Vous venez de le voir dans la vidéo, l'algorithme de tri du tableau est réalisé à l'aide d'un algorithme de partitionnement (itératif) et d'un algorithme récursif (le tri rapide) faisant appel à lui même ainsi qu'à l'algorithme de partitionnement. Il est donc nécessaire de détailler ces deux algorithmes avant de pouvoir coder le tri rapide.

Exercice 3. Algorithme général du tri rapide.

Nous nous intéressons dans un premier temps à la description générale de l'algorithme. Pour le moment, ne vous préoccupez pas de l'exactitude des indices, ils seront affinés plus tard si besoin.

1) Quel type de récursivité est utilisé pour le tri rapide ? Simple
2) Le tri rapide faisant appel à lui-même sur des sous-tableaux, quelles sont les entrées de l'algorithme général ? taille tableau, indice du pivot, index
3) Quelle est la sortie de l'algorithme ? un appel de fonction sur la nouvelle portion
4) Rédigez le pseudo-algorithme du tri rapide. Ne détaillez pas encore l'algorithme de partitionnement, vous devez simplement l'appeler. Utilisez la portion de vidéo entre 3:00 et 3:50 pour vous aider.

6. Vous pourrez remarquer le bon choix du marqueur pour les années à venir...

Exercice 4. Partitionnement : prototype.

Nous étudions maintenant uniquement l'algorithme de partitionnement d'un sous-tableau.

1) Quel élément est choisi en tant que pivot ?
2) Où se placent les indices en début d'algorithme de partitionnement ?
3) Déduisez-en les entrées, puis les sorties de l'algorithme de partitionnement.

Il est temps de compléter l'algorithme de partitionnement. Il est relativement simple à comprendre, moins à rédiger du fait de la présence de plusieurs indices toujours en mouvement. Répondez précisément aux questions suivantes afin de ne pas vous tromper dans la rédaction finale de l'algorithme.

Exercice 5. Partitionnement : rédaction de l'algorithme.

1) Quel évènement signe la fin du partitionnement ?
2) Traduisez cet évènement en termes de condition d'arrêt.
3) Quelles conditions doivent être réunies pour que l'indice gauche avance vers le droit ?
4) Même question pour que l'indice droit avance vers l'indice gauche.
5) Que doit-il se passer si les valeurs sous les indices sont égales ?
6) Que se passe-t-il lorsque le partitionnement est terminé ?
7) Déduisez-en l'algorithme complet du partitionnement.

Exercice 6.

Il est temps de programmer le tri rapide. Votre tri rapide doit fonctionner pour l'ensemble des cas suivants.

1) Les cartes présentées dans la vidéo : 4, 1, 2, 5, 6, 3.
2) Tableau déjà trié : 1, 2, 3, 4, 5, 6.
3) Tableau inversé (pire cas) : 6, 5, 4, 3, 2, 1.
4) Répétitions : 4, 1, 2, 4, 1, 3.
5) Élément unique : 3, 3, 3, 3, 3, 3.

3 Synthèse

Vous venez de découvrir deux algorithmes du type « diviser pour régner ». La recherche dichotomique permet de retrouver rapidement un élément dans un tableau trié. Le tri rapide permet de trier un tableau de façon plus efficace qu'un tri quadratique. Pour ces deux exemples, vous devez être capable de rédiger leur algorithme et de les implémenter en C. De plus, vous devez être capable de dérouler un exemple complet.

Exercice 7.

Répondez aux questions suivantes afin de résumer ce que vous venez d'apprendre sur ces deux méthodes.

1) Identifiez les trois étapes *diviser* - *régner* - *combiner* pour les algorithmes de recherche dichotomique et de tri rapide.
2) Donnez la complexité des deux algorithmes.
3) Résumez chacun des deux algorithmes en une phrase.

4 Pour aller plus loin

L'algorithme de partitionnement du tri rapide que vous pouvez lire dans les manuels ou sites web à caractère pédagogique diffère légèrement de celui qui vous est présenté ici. Nous avons souhaité introduire un algorithme simple à comprendre, de complexité asymptotique identique aux autres algorithmes (à constante près). Le principal problème de l'algorithme de partitionnement que vous venez de voir est qu'il effectue trop d'échanges.

Plusieurs autres algorithmes de partitionnement existent. Celui d'Hoare a le mérite d'être simple à implémenter tout en étant particulièrement efficace. Vous pouvez retrouver son pseudo-code sur la page Wikipédia du tri rapide en anglais (quicksort).

Exercice 8. Tri (encore plus) rapide.
Implémentez une version optimisée du tri rapide en utilisant l'algorithme de partitionnement d'Hoare.

Annexe 1

Vous trouverez ci-dessous le tableau à compléter pour le tri rapide. Celui-ci donne sur la première ligne l'état initial du tableau, puis sur les lignes 2 et 3 les deux premiers échanges effectués par le tri rapide. Le pivot est souligné, puis entouré lorsque la valeur ne sera plus échangée au cours de l'algorithme.

4	1	2	5	6	3
3	1	2	5	6	4
3	1	2	4	6	5
2	1	3	4	6	5
1	2	3	4	6	5
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6
1	2	3	4	5	6

2) Résumez en quelques mots (pas plus de 5 lignes) le principe du tri. Si besoin, reformulez la description faite au début de la vidéo.

Diviser le problème en plusieurs sous problèmes pour limiter la complexité de l'algorithme