TRIER VITE POUR CHERCHER VITE

par
Timothy R. Walsh, Département d'Informatique, UQAM

Quand vous cherchez un mot dans un dictionnaire de 1 000 000 mots, vous ne regardez pas chaque mot du début du dictionnaire à la fin. Vous pouvez trouver votre mot en ayant regardé une vingtaine de mots seulement parce que les mots sont triés en ordre alphabétique. Bien sûr, la première lettre de votre mot vous aide à commencer votre recherche - par exemple, puisque le mot 'épouvantable' commence par la cinquième lettre de l'alphabet de 26 lettres, vous pouvez supposer que ce mot se trouve environ un cinquième de la distance du début du dictionnaire à la fin. Mais cette supposition ne tiendrait pas si le dictionnaire consistait en les mots (admirable, bon, convenable, déplorable, épouvantable)! En général, et surtout quand on travaille avec des nombres au lieu des mots, on ne peut rien supposer sur la distribution des objets dans le tableau. Tout ce qu'on peut supposer, c'est que le tableau est trié en ordre croissant, comme

(22,22,33,33,33,45,45,**57**,57,57,61,73,73,87,99).

La meilleure stratégie, c'est de comparer le nombre que vous cherchez avec le nombre au milieu du tableau, qui est 57 (caractères gras). Si vous cherchez 45, par exemple, puisque 45<57 vous savez que 45 doit venir avant le milieu du tableau. Si vous cherchez 61, puisque 61>57 vous savez que 61 doit venir après le milieu du tableau. En tout cas, avec une seule comparaison vous avez réduit par un facteur de 2 la taille de la portion du tableau dans laquelle vous devez chercher votre nombre. Puis vous regardez l'entrée au milieu de cette portion du tableau, et ainsi de suite.

Pour illustrer ce processus, qu'on appelle 'fouille dichotomique' ou 'recherche binaire', je vais écrire le tableau avec la position de chaque entrée:

```
position  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
entrée   22 22 33 33 33 45 45 57 57 57 61 73 73 87 99
```

Maintenant je donne la position la plus à gauche et la position la plus à droite de la portion du tableau dans laquelle il faut chercher après chaque comparaison pendant la recherche du nombre 61. Chaque comparaison sera faite avec l'entrée dans la position au milieu de cette portion du tableau, c'est-à-dire (gauche+droite)/2.

gauche	droite	milieu	entrée	commentaire
1	15	8	57	61>57, d'où 61 doit être après position 8.
9	15	12	73	61<73, d'où 61 doit être avant position 12.
9	11	10	57	61>57, d'où 61 doit être après position 10.
11	11	11	61	61=61, d'où 61 est en fait à la position 11.

Ce dernier test est nécéssaire. Essayez de trouver 62 par la fouille dichotomique. Après 3 comparaisons, vous allez conclure que 62 doit être entre les positions 11 et 11 *si 62 était dans la*

tableau. Mais puisque la onzième entrée du tableau n'est pas 62, vous pouvez conclure correctement que 62 n'est pas dans le tableau.

Combien de comparaisons sont nécessaires pour chercher un objet dans un tableau de taille n? Prenons le cas où n=15. Au début on cherche dans un tableau de taille 15, puis dans un tableau de taille 7 (de position 9 à position 15 inclusivement), puis de taille 3, puis de taille 1. À chaque étape la taille est divisée par 2 avec arrondissement vers le bas. Quand la taille est réduite à 1, vous devez faire encore une comparaison pour décider si l'objet que vous cherchez est vraiment dans le tableau. Le nombre de comparaisons nécessaires est le nombre de fois qu'il faut diviser n par 2 pour le réduire à 1, plus encore 1 comparaison pour le dernier test. Quand n = 15, la division successive donne 15, 7, 3, 1: 3 divisions, et donc 3 comparaisons, plus le dernier test, pour un total de 4. Quand n = 1 000 000, il faut 19 divisions pour le réduire à 1 (essayez-le!), et donc le nombre total de comparaisons nécessaires est 20. Je note par $d(n)$ le nombre de fois qu'il faut diviser n par 2 pour le réduire à 1 (si vous connaissez les logarithmes, vous pouvez vérifier que $d(n)$ est le plus grand entier $\leq \log_2 n$). Le nombre total de comparaisons nécessaires est donc $d(n)+1$. Avez-vous remarqué que j'ai choisi n pour que la division soit toujours avec un reste? Bon, je commence avec un tableau de 8 entrées (22,33,45,57,61,73,87,99). La première comparaison sera avec 57. Si je cherche 87, j'ai le sous-tableau (61,73,87,99) avec 4 entrées. Si je cherche 33, j'ai le sous-tableau (22,33,45) avec 3 entrées. Si je cherche 57 j'ai fini. Je veux trouver le nombre maximum possible de comparaisons, ce qui m'oblige de considérer le cas où j'ai le plus grand tableau - celui avec 4 entrées - et en divisant 8 par 2 j'obtiens 4. La formule tient.

L'importance de cette formule est que la taille de $d(n) + 1$ reste modérée même quand n devient très grand. Il faut 10 comparaisons pour chercher un mot dans un dictionnaire de taille 1 000, 20 comparaisons dans un dictionnaire de taille 1 000 000, 30 comparaisons si la taille est 1 000 000 000, etc. La fouille dichotomique est un processus vraiment rapide par rapport à la fouille linéaire: commencez du début et continuez jusqu'à la fin, ou jusqu'à ce que vous trouviez la chose que vous cherchez.

Mais supposons que vous cherchez dans un bottin téléphonique le nom de quelqu'un dont vous ne connaissez que le numéro de téléphone. Vous devez chercher du début à la fin du bottin puisque les numéros de téléphone ne sont pas triés. Pour permettre aux gens de profiter de la rapidité de la fouille dichotomique il faut d'abord trier le tableau. La façon la plus facile à comprendre pour trier un tableau s'appelle *tri par sélection*. Cherchez tout le tableau pour trouver l'entrée qui doit aller à la première position et échangez-la avec la première entrée. Puis cherchez

tout le tableau sauf la première entrée pour trouver l'entrée qui doit aller à la deuxième position et échangez-la avec la deuxième entrée, et ainsi de suite.

J'illustre le processus avec le tableau (61,87,73,57,22,45,99,33). Pour trouver la plus petite entrée du tableau, supposez d'abord que c'est la première entrée 61. Comparez votre candidat 61 avec la deuxième entrée 87. Puisque 87>61, votre candidat est toujours 61. Comparez-le avec la troisième entrée 73: il reste 61. Comparez-le avec la quatrième entrée 57: 57<61, et donc votre nouveau candidat est 57. Comparez-le avec le cinquième entrée 22: 22<57, et donc votre nouveau candidat est 22. Comparez-le avec la sixième, septième et huitième entrée, et le candidat reste 22. La plus petite entrée est donc 22. Vous avez comparé votre candidat avec toutes les entrées sauf la première - ça fait 7 comparaisons. Le processus complet est illustré dessous. Les entrées du tableau qui ont été mises dans leurs propres positions sont à gauche d'une barre verticale. La plus petite entrée dans le reste du tableau se trouve dans la colonne avec l'entête m, sa position dans la colonne p, et le nombre de comparaisons nécessaires pour le trouver dans la colonne c.

position	1	2	3	4	5	6	7	8	m	p	c	échange à faire.
entrée	|61	87	73	57	22	45	99	33	22	5	7	22 avec 61
	22|	87	73	57	61	45	99	33	33	8	6	33 avec 87
	22	33|	73	57	61	45	99	87	45	6	5	45 avec 73
	22	33	45|	57	61	73	99	87	57	4	4	57 avec 57
	22	33	45	57|	61	73	99	87	61	5	3	61 avec 61
	22	33	45	57	61|	73	99	87	73	6	2	73 avec 73
	22	33	45	57	61	73|	99	87	87	8	1	87 avec 99
	22	33	45	57	61	73	87|	99				

Le nombre total de comparaisons est 1+2+3+4+5+6+7.
La même somme peut être écrite de droite à gauche: 7+6+5+4+3+2+1.
Si on additionne colonne par colonne obtient 8+8+8+8+8+8+8 = 7*8, ce qui est deux fois la somme. La somme est donc (7*8)/2=28. De la même façon on peut montrer que le nombre de comparaisons nécessaires pour trier un tableau de taille n est $1+2+...+(n-1) = n(n-1)/2$. Quand n = 1 000 000, ce nombre est énorme: 499 999 500 000. Un ordinateur qui exécute 1 000 000 comparaisons par seconde prendrait *une semaine* pour trier un tableau de taille 1 000 000.

Heureusement il existe des façons plus rapides pour trier un tableau. Parmi elles, la plus facile à comprendre est *tri par fusion*. Pour illustrer ce processus, supposons que vous avez deux tableaux qui sont déjà triés: (57,61,73,87) et (22,33,45,99) et que vous voulez en faire un seul tableau trié. Vous comparez la première entrée de chacun des deux tableaux, et vous mettez en première position dans un grand tableau la plus petite parmi elles en l'enlevant du tableau duquel elle appartenait. Puis vous comparez la première entrée de qui reste des deux tableaux, et vous en mettez la plus petite en deuxième position dans le grand tableau, et ainsi de suite:
premier tableau deuxième tableau grand tableau

```
57 61 73 87     22 33 45 99
57 61 73 87        33 45 99     22
57 61 73 87           45 99     22 33
57 61 73 87              99     22 33 45
   61 73 87              99     22 33 45 57
      73 87              99     22 33 45 57 61
         87              99     22 33 45 57 61 73
                         99     22 33 45 57 61 73 87
                                22 33 45 57 61 73 87 99
```

Le nombre de transferts d'un des petits tableaux au grand tableau est la taille finale du grand tableau, et le nombre de comparaisons est au plus la taille finale du grand tableau moins 1 puisqu'au moins le dernier transfert se fait sans aucune comparaison. On peut donc considerer que le coût d'une fusion est la taille finale du grand tableau.

Comment utiliser ce processus de fusion pour trier un tableau? Considerez chaque entrée du tableau comme un sous-tableau de taille 1. Un tableau de taille 1 est forcément trié. Vous pouvez donc fusionner le premier sous-tableau avec le deuxième, le troisième avec le quatrième, et ainsi de suite, et vous aurez maintenant des sous-tableaux triés de taille 2. Puis répétez le processus pour avoir des sous-tableaux triés de taille 4, et ainsi de suite. J'illustre le process dessous; les barres verticales séparent les sous-tableaux triés.

```
                                        coût  (somme des tailles des grands sous-tableaux)
|61|87|73|57|22|45|99|33|
|61 87|57 73|22 45|33 99|     2+2+2+2 = 8
|57 61 73 87|22 33 45 99|       4 + 4 = 8
|22 33 45 57 61 73 87 99|             8
```

Remarquez que le coût de chaque étape de ce processus est la taille du tableau. Combien y a-t-il d'étapes? Chaque étape réduit par un facteur de 2 le nombre de sous-tableaux, et le processus se termine quand il y a un seul sous-tableau. Donc le nombre d'étapes est le nombre de fois qu'il faut diviser la taille du tableau par 2 pour la réduire à 1. Si le tableau est de taille n, c'est $d(n)$. Le coût total est donc le nombre d'étapes multiplié par le coût de chaque étape, c'est-à-dire $nd(n)$. Quand n = 1 000 000, $d(n)$=19 et $nd(n)$ = 19 000 000. Le même ordinateur qui exécute 1 000 000 opérations par seconde - soit comparaisons, soit transferts - va prendre 19 secondes pour faire les transferts et au plus 19 secondes pour faire les comparaisons pour un temps total de 38 secondes, ce qui est beaucoup moins que la semaine nécessaire pour trier le même tableau à l'aide du tri par sélection!

Avez-vous remarqué que j'ai triché encore une fois - que j'ai utilisé un tableau de taille 8, qui peut être divisé successivement par 2 sans jamais avoir un reste? Bon - je vais ajouter une entrée au

tableau pour illustrer le cas général. Et cette fois je vais compter le nombre de sous-tableaux - vous pouvez ignorer le nombre de comparaisons pour le moment.

```
                                                  nombre de sous-tableaux  nombre de comparaisons
|61|87|73|57|22|45|99|33|50|                      9
|61 87|73 57|22 45|99 33|50|                       5                    1+1+1+1 = 4
|57 61 73 87|22 33 45 99|50|                       3                      3 + 3 = 6
|22 33 45 57 61 73 87 99|50|                       2                              7
|22 33 45 50 57 61 73 87 99|                       1                              8
                                                                 nombre total de comparaisons =25
```

Le nombre de sous-tableaux est toujours divisé par 2, mais cette fois avec arrondissement vers le haut. Le nombre de fois qu'il faut diviser n par 2 en arrondissant vers le haut est $d(n)$ si toutes les divisions se font sans reste et $d(n) + 1$ sinon (en fait, c'est le plus petit entier $\geq \log_2 n$) - essayez-le avec 8 et 9 pour vous en convaincre. Donc, au lieu de $nd(n)$ j'aurais dû dire $n(d(n)+1)$ pour le coût de tri par fusion sur un tableau de taille n. Le pauvre ordinateur doit prendre 40 secondes au lieu de 38 sur le tableau de taille 1 000 000!

Eh bien, si vouz tenez à une telle précision, considérez l'astuce suivante. Au lieu de fusionner le premier sous-tableau avec le deuxième, je fusionne le premier sous-tableau avec le dernier, en mettant le grand sous-tableau au début, et puis je fusionne le deuxième avec le troisième, le quatrième avec le cinquième et ainsi de suite. J'illustre ce processus dessous et je compte les comparaisons. Je vous rappelle que le nombre de comparaisons nécessaires pour fusionner deux tableaux est la taille du tableau fusionné moins 1, et un sous-tableau qui est copié sans fusion, comme (45,99) de la deuxième ligne à la troisième, ne compte pas.

```
                                                                  nombre de comparaisons
              |61|87|73|57|22|45|99|33|50|
          |50 61|73 87|22 57|45 99|33|                  1+1+1+1 = 4
       |33 50 61|22 57 73 87|45 99|                        2 + 3 = 5
|33 45 50 61 99|22 57 73 87|                                      4
|22 33 45 50 57 61 73 87 99|                                      8
                                                 nombre total de comparaisons = 21
```

Regardez bien le nombre total de comparaisons avec et sans cette astuce. Expliquez, si vous pouvez, comment je suis arrivé à économiser 4 comparaisons!