TD5 : Algorithmique

MP2I Lycée Pierre de Fermat

Vous pouvez télécharger sur Cahier de Prépa une archive pour ce TD. Dans ce TD, on étudie plusieurs problèmes algorithmiques. Pour chaque problème, le but est de trouver un algorithme le plus efficace possible. Vous pouvez donc coder et tester vos algorithmes en C, et ensuite passer à l'étude théorique : terminaison, correction et complexité.

Un bon réflexe à prendre lorsque l'on étudie un problème est d'avoir des fonctions qui génèrent des données, pour pouvoir tester les algorithmes rapidement, ainsi que des fonctions qui affichent les données du problème. Pour ce TD, l'archive contient du code où tout cela est déjà fait : il ne reste pour chaque exercice qu'à remplir la fonction principale, celle qui implémente l'algorithme de résolution.

Exercice 1. *Recherche dans un tableau trié*

On considère le problème suivant : "Étant donné un tableau T d'entiers triés dans l'ordre croissant et un entier x, déterminer si x est dans T."
La spécification précise d'un algorithme de résolution serait donc :

Entrée(s) : T tableau croissant de n entrées, x élément à rechercher
Sortie(s) : Vrai si $\exists i \in [\![0, n-1]\!], T[i] = x$, Faux sinon

---

Q1. Donnez la spécification précise d'un algorithme répondant à ce problème.

Le fichier `exercice1.c` de l'archive est un programme générant un tableau, l'affichant, puis demandant un élément à chercher dans le tableau. Le programme affiche ensuite "Oui" ou "Non" selon si l'élément a été trouvé ou non. Le programme est incomplet, la fonction `bool search(int* t, int n, int x)` effectuant la recherche n'a pas été implémentée. Le but de l'exercice est de compléter cette fonction, et d'étudier la correction et la complexité des différents algorithmes trouvés.

Q2. Écrivez une première version en $\mathcal{O}(n)$ à l'aide d'un algorithme naïf. Donnez un invariant de boucle permettant de montrer sa correction.

Pour exploiter le fait que le tableau en entrée est trié, on utilise le principe de ***dichotomie*** (déjà vu au TD3) :

On regarde la case du milieu et, en notant y sa valeur :

— si $x = y$, alors x est dans le tableau : on a fini la recherche.

— si $x < y$ alors x est dans la première moitié du tableau

— si $x > y$ alors x est dans la deuxième moitié du tableau

Dans les deux derniers cas, on peut éliminer la case du milieu ainsi que toutes celles étant dans la mauvaise moitié du tableau, on a donc éliminé plus de la moitié des valeurs et il ne reste qu'à continuer la recherche dans la partie restante.

On propose le pseudo-code suivant :

Algorithme 1 : Dichotomie

Entrée(s) : T tableau croissant de n entrées, x élément à rechercher
Sortie(s) : Booléen indiquant la présence de x dans T

```
1  A ← 0;
2  B ← n − 1;
   // Invariant: ?
3  tant que B ≥ A faire
4      M ← (A+B)/2;
5      si T[M] = x alors
6          retourner Vrai
7      sinon
8          si T[M] < x alors
9              A ← M + 1;
10         sinon
11             B ← M − 1;
12 retourner Faux
```

Q3. Exécuter à la main l'algorithme sur deux exemples : un où l'élément est dans le tableau, et l'autre où il n'y est pas. A chaque étape, barrer les cases du tableau ayant été éliminées, i.e. n'étant plus dans l'intervalle de recherche.

Q4. Renommez votre première fonction de recherche en `search_lin`, et implémentez l'algorithme de dichotomie pour faire une nouvelle version de la fonction `search`.

Q5. On se place au début d'un passage, et on note A', B' les valeurs de A et B à la fin de ce passage. Montrer que $B' - A' \leq \frac{B-A}{2}$. En déduire que l'algorithme est en $\mathcal{O}(\log_2 n)$.

Montrons la correction de cet algorithme.

Q6. La propriété "$T[A] \leq x \leq T[B]$" est-elle un invariant de boucle ?

Q7. Proposer un invariant de boucle convenable et montrer qu'à nouveau l'algorithme est correct dans ce cas. Une fois l'invariant identifié, on pourra faire une disjonction de cas, selon que l'algorithme renvoie Vrai ou Faux.

Q8. Écrire une fonction `bool invariant(int* T, int n, int x, int A, int B)` utilisant la fonction `lin_search` qui calcule l'invariant trouvé à la question précédente. Ajouter une assertion dans le code de la fonction `search` permettant de vérifier l'invariant.

Exercice 2. *Recherche dans une matrice triée*

On considère maintenant des tableaux 2D. Pour M une matrice de dimensions $n \times m$, on dit que M est ***doublement ordonnée*** si chaque ligne de M est triée par ordre croissant, et si chaque colonne de M est triée par ordre croissant.

On se pose le problème suivant : étant donné M doublement ordonnée et x un entier, est-ce-que x est dans M ?

Le fichier `exercice2.c` contient du code pour répondre à ce problème, où seule la fonction de recherche est à remplir.

Q1. Donnez un premier algorithme en $\mathcal{O}(nm)$.

Q2. Donnez un deuxième algorithme en $\mathcal{O}(n \log m)$ ou en $\mathcal{O}(m \log n)$.

Q3. On suppose qu'on cherche x. On teste la case $M[i][j]$. Si $x < M[i][j]$, quelles sont les cases du tableau que l'on peut éliminer pour notre recherche ? Même question si $x > M[i][j]$?

Une page interactive vous permettant de tester des stratégies est également disponible au lien suivant : perso.ens-lyon.fr/guillaume.rousseau/mp2i/demos/recherche2D. Cliquez sur une case pour éliminer toutes celles qui deviennent inutiles.

Q4. Donnez un troisième algorithme en $\mathcal{O}(n + m)$ et montrez sa correction.

Q5. En s'inspirant de l'algorithme trouvé à la question précédente, proposer un algorithme efficace pour résoudre le problème suivant : Étant donné T et U deux tableaux croissants, de tailles respectives n et m, et $S \in \mathbb{N}$, déterminer s'il existe deux indices i, j tels que $T[i] + U[j] = S$. *Indication : on pourra chercher à définir une bonne matrice doublement ordonnée.*

Exercice 3. *Achat et vente d'actions*

On considère un tableau de valeurs T, de taille n, représentant le cours d'une action. On cherche les meilleurs moments pour acheter et vendre, autrement dit on cherche $0 \leq i < j < n$ deux entiers tels que $T[j] - T[i]$ est maximal.

Q1. Pour simuler le cours d'une action, on peut remplir chaque case du tableau de manière aléatoire, mais ça ne donne pas un cours très réaliste. Une simulation un peu plus réaliste est de remplir le tableau avec une valeur pouvant varier d'un petit nombre aléatoire à chaque instant. Implémenter cette idée

Q2. Donner un algorithme en $\mathcal{O}(n^2)$.

Q3. Supposons que l'on vend à l'instant j_0. Quel est l'instant i_0 optimal pour acheter ?

Q4. Donner un algorithme en $\mathcal{O}(n)$.