TD5 : Algorithmique

MP2I Lycée Pierre de Fermat

Dans ce TD, on étudie plusieurs problèmes algorithmiques. Pour chaque problème, le but est de trouver un algorithme avec la meilleure complexité asymptotique possible.

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. 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*** :
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`(T, n, x)

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;
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
```

Q2. 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. Représenter le tableau à chaque début de passage de boucle, en barrant les cases ayant été éliminées, i.e. dont les indices ne sont plus dans l'intervalle de recherche $[\![A, B]\!]$.

Q3. Justifier que si l'algorithme renvoie Vrai, alors T contient x.

Q4. On suppose maintenant que l'algorithme renvoie Faux. La propriété "$T[A] \leq x \leq T[B]$" est-elle un invariant de boucle ?

Q5. Proposer un invariant de boucle convenable et montrer que T ne contient pas x.

Q6. Modifier l'algorithme `dichotomie`(T, n, x) pour qu'il renvoie **l'indice** d'une case contenant x, ou bien -1 si aucun tel indice n'existe.

Q7. 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 $\mathcal{O}(\log_2 n)$.

Exercice 2.

Recherche dans une matrice triée

On considère des tableaux 2D, c'est à dire des matrices. 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.

Q1. Compléter la matrice suivante pour qu'elle soit doublement ordonnée et ne contienne aucun doublon :

$$\begin{pmatrix} 6 & ? & ? \\ ? & 9 & 12 \\ 10 & ? & 14 \end{pmatrix}$$

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

Q2. Donnez un premier algorithme en $\mathcal{O}(nm)$.

Q3. Donnez un deuxième algorithme en $\mathcal{O}(n \log m)$ ou en $\mathcal{O}(m \log n)$.

Cherchons un algorithme plus efficace. On suppose qu'on cherche un entier x dans une matrice M.

Q4. 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 fois que vous avez répondu à la question précédente et pas avant), vous pouvez aller au lien suivant et tester des stratégies : perso.ens-lyon.fr/guillaume.rousseau/mp2i/demos/recherche2D (dézoomez un peu pour voir l'écran en entier, en particulier l'élément recherché). Cliquer sur une case élimine toutes les cases jugées inutiles selon le critère de la question précédente.

Q5. Donnez un troisième algorithme en $\mathcal{O}(n + m)$ et montrez sa correction.

Exercice 3.

Soit $K \in \mathbb{N}$. Étant donné T un tableau de taille n dont tous les éléments sont dans $[\![0, K-1]\!]$, on souhaite trouver $k \in [\![0, K-1]\!]$ ayant le plus d'occurrences dans T.

Q1. Proposer un algorithme en $\mathcal{O}(Kn)$.

Afin d'améliorer l'algorithme précédent, on passe par la construction d'un **tableau des occurrences de** T. Ce tableau, noté O_T, est de taille K, et est tel que $O_T[k]$ contient le nombre d'occurrences de k dans T.

Q2. Pour $K = 5$, et $T = [2, 3, 2, 4, 0, 2, 3]$, donner O_T.

Q3. Proposer un algorithme construisant O_T en $\mathcal{O}(K + n)$. On supposera que créer un tableau de taille X prend un temps $\mathcal{O}(X)$:

Algorithme 2 : `occurrences`(T, n, K)

Entrée(s) : T tableau de taille n, $K \in \mathbb{N}$.
Précondition : $\forall i \in [\![0, n-1]\!], 0 \leq T[i] < K$
Sortie(s) : O_T tableau des occurrences de T

Q4. En déduire un algorithme pour résoudre le problème initial en $\mathcal{O}(n + K)$.

Une deuxième application du tableau des occurrences : sur les tableaux dont les valeurs sont comprises entre 0 et $K - 1$, on peut trouver un algorithme de tri très efficace.

Q5. Proposer un algorithme permettant de trier T en $\mathcal{O}(n + K)$.

Exercice 4.

Carrés de 1

Cet exercice est plus ouvert (et donc plus dur) que les autres.
Étant donné une matrice M carrée de taille $n \times n$ booléenne (i.e. dont tous les coefficients sont 0 ou 1), on souhaite trouver le plus grand rectangle entièrement constitué de 1, c'est à dire deux indices de lignes $i_1 \leq i_2$ et deux indices $j_1 \leq j_2$ tels que $\forall i \in [\![i_1, i_2]\!], \forall j \in [\![j_1, j_2]\!]$, $M_{ij} = 1$, avec $(i_2 - i_1 + 1)(j_2 - j_1 + 1)$ maximal.
Trouver un algorithme le plus efficace possible. Commencez par chercher un algorithme naïf et par évaluer sa complexité, puis essayez de l'améliorer.