Devoir Maison Algorithmique
Recherche de valeur majoritaire

Guillaume Bury

A rendre à Serge Haddad lors du cours du mercredi 26 octobre, rédigé à la main sur papier.

Soit $T[1..n]$ un tableau. Une valeur e présente dans T est dite majoritaire si T contient strictement plus de $\frac{n}{2}$ occurrences de e. Dans ce devoir, on s'intéressera à étudier la complexité d'algorithmes de recherche de valeur majoritaire.

1 Préliminaires

Supposons que les éléments du tableau supportent les comparaisons (on dispose d'un ordre total sur ces éléments). On rappelle qu'il existe un algorithme permettant de trouver la k-ième plus petite valeur d'un tableau en temps linéaire en la taille du tableau.

1. Donner un algorithme de recherche de la valeur majoritaire qui s'exécute en temps linéaire en la taille du tableau d'entrée.

Dans le reste de cette section, on suppose que les éléments du tableau ne prennent que deux valeurs.

2. Justifier que si n est impair, alors il existe une valeur majoritaire.
3. Donner un algorithme de recherche de valeur majoritaire qui effectue $n-1$ comparaisons dans le pire des cas.
4. Considérons le cas ou $n = 5$. Donner un algorithme qui permet de trouver la valeur majoritaire en 3 comparaisons.
5. Question ouverte (optionnelle) : Existe-t-il une suite strictement croissante $(u_n)_{n\in\mathbb{N}}$, telle qu'on puisse trouver la valeur majoritaire d'un tableau de taille u_n en au plus $u_n - 2$ comparaisons ? Si oui, donner un algorithme qui respecte cette borne, et justifier sa complexité.

2 Un algorithme avec égalité uniquement

A partir de maintenant, on suppose que l'on peut uniquement tester l'égalité de deux éléments du tableau (pas de relation d'ordre). On propose alors l'algorithme suivant :

Initialisation : On dispose d'un sac rempli de n balles de couleurs, et l'on voudrait trouver la couleur majoritaire, si elle existe. On dispose d'une étagère, d'une boite et d'une poubelle, toutes initialement vides.

Première phase : On pose une première balle sur l'étagère, puis pour chaque balle b restant dans le sac, si b a la même couleur que la dernière balle posée sur l'étagère, alors on met b dans la boite, sinon on met b sur l'étagère, et on prend une balle de la boite pour la mettre sur l'étagère, si possible (i.e. si la boite n'est pas vide).

Deuxième phase : Notons C la couleur de la dernière balle posée sur l'étagère. On va examiner les balles sur l'étagère, dans l'ordre inverse de la pose.

— Si la balle courante a une couleur différente de C :

— Si la boite est vide : on arrête et on déclare qu'il n'y a pas de couleur majoritaire ;

— Si la boite n'est pas vide, on prend la balle courante et une balle de la boite, et on les jette à la poubelle.
— Si la balle courante est de couleur C :
  — S'il reste au moins deux balles, on jette les deux dernières balles à la poubelle ;
  — Si c'est la dernière balle, on la met dans la boite.

Fin : Lorsqu'il n'y a plus de balles sur l'étagère, on déclare C la couleur majoritaire ssi la boite n'est pas vide.

1. Prouver qu'à tout moment de la première phase les couleurs de la boîte ont la même couleur que la dernière balle posée sur l'étagère et qu'il n'y a pas deux boules contiguës de la même couleur sur l'étagère. En déduire que s'il y a une couleur majoritaire, alors il s'agit de C.
2. Trouver et prouver un invariant maintenu lors de la deuxième phase.
3. Prouver que l'algorithme est correct
4. D'après le traitement fait par l'algorithme, on sait qu'il va s'exécuter en temps linéaire. Donner le nombre exact de comparaisons effectuées par l'algorithme dans le pire des cas (indice : ce nombre dépend de la parité de n).

3 Borne inférieure de l'algorithme de recherche

Dans cette section, on suppose que les éléments du tableaux appartiennent à un ensemble infini.

On s'intéresse maintenant à établir une borne inférieure de la complexité d'un algorithme de recherche d'élément majoritaire qui ne pratique que des tests d'égalité. Cette borne porte sur le nombre de tests d'égalité. On note $m = \lfloor n/2 \rfloor + 1$ le seuil de majorité.

Afin d'établir cette borne, nous imaginons que l'algorithme « joue » contre un adversaire. A chaque fois que l'algorithme effectue un test d'égalité entre deux éléments, l'adversaire a la possibilité de choisir le résultat de ce test, tant que cela reste cohérent avec les résultats de tests précédents. Le but de l'adversaire est de maximiser le nombre de comparaisons qu'effectue l'algorithme.

Affectation. Une entrée possible (qu'on appellera dans la suite *affectation*) est une partition de l'ensemble des indices telle que deux cellules du tableau sont identiques (i.e le résultat du test d'égalité est positif) ssi les indices correspondants appartiennent au même sous-ensemble de la partition.

Amphithéâtre. Afin de maximiser le nombre de comparaison effectuées par l'algorithme, l'adversaire maintient une structure, appelée l'amphithéâtre, où les indices sont répartis entre les gradins et l'arène. Dans l'arène, les indices sont :

— soit groupés en troupeaux ; au sein d'un troupeau les cellules des indices doivent avoir la même valeur. On note Tr, le nombre de troupeaux et t le nombre total d'indices dans un troupeau.
— soit groupés en binômes ; au sein d'un binôme les deux cellules des indices doivent avoir une valeur différente. On note B, le nombre de binômes

Les troupeaux représentent les résultats positifs des tests d'égalité : c'est la clôture réflexive transitive des égalités connues par l'algorithme. A l'inverse les binômes représentent une partie des résultats négatifs des tests d'égalités : une paire représente deux éléments du tableaux qui doivent être distincts. Enfin les indices mis dans les gradins doivent chacun être différents de tout autre élément du tableau. On note g le nombre d'indices dans les gradins.

Admissibilité. Une affectation qui satisfait les contraintes donné par un amphithéâtre A (tel que spécifié ci-dessus) est dite *admissible* par rapport à A.

Stratégie de l'adversaire. Initialement, tous les indices sont dans l'arène et constituent des troupeaux singletons. Lorsque l'algorithme demande le résultat d'un test d'égalité $T[i] \stackrel{?}{=} T[j]$, la réponse au test d'égalité et les modifications de l'amphithéâtre sont décidées par l'adversaire en suivant la stratégie suivante :

- Si i ou j sont dans les gradins, alors la réponse est non et l'amphithéâtre est inchangé.
- Si i et j forment un binôme alors la réponse est non et l'amphithéâtre est inchangé.
- Si i (resp. j) est dans un binôme et j (resp. i) n'est pas son partenaire, alors la réponse est non et i (resp. j) est envoyé dans les gradins alors que son partenaire forme un troupeau singleton.
- Si i et j sont dans un même troupeau, alors la réponse est oui et l'amphithéâtre est inchangé.
- Si i et j sont dans des troupeaux différents, alors l'action dépend de la valeur $d = B + t$.
  - Si $d > m$ les deux troupeaux sont des singletons (voir la question 1) alors la réponse est non et i, j forment un binôme.
  - Si $d = m$ la réponse est oui et les troupeaux de i et j fusionnent.

Exécution partielle. Une exécution partielle de l'algorithme E est une suite $(T[i_k] \equiv_k T[j_k])_{1 \leq k \leq l}$ (où $\equiv_k \in \{=, \neq\}$) de résultats de tests d'égalité, telle que la réponse à l'indice o (i.e. $T[i_o] \equiv_o T[j_o]$) est déterminée à partir de l'état de l'amphithéâtre à l'instant $o-1$ par la stratégie donnée ci-dessus. On notera A_E l'amphithéâtre à la fin de l'exécution partielle E.

Cohérence. Une affectation a est dite cohérente avec une exécution partielle E ssi les résultats des tests d'égalité dans E sont ceux qui auraient été obtenus avec l'affectation a.

Terminaison de l'algorithme. Une exécution complète est une paire (E, r) d'une exécution partielle et d'un résultat r. L'exécution complète est dite correcte ssi pour toute affectation a cohérente avec E, le résultat r est correct pour a (i.e. soit r est la couleur majoritaire de a, soit a n'admet pas de couleur majoritaire et $r = null$).

Questions :

1. Montrer que pour toute exécution partielle :
   - d ne croît jamais ;
   - $d \geq m$;
   - $d > m$ implique que les troupeaux sont des singletons.
2. Montrer que pour toute exécution partielle E, toute affectation a admissible par rapport à A_E, est cohérente par rapport à E. En déduire que pour toute exécution partielle E, il existe au moins une affectation cohérente avec E.
3. Montrer que pour toute exécution partielle E de l'algorithme, si $d > m$ dans A_E, alors il n'existe pas de résultat r tel que (E, r) est une exécution complète de l'algorithme. En d'autres termes, tant que $d > m$, l'algorithme ne peut pas conclure sur l'existence/la valeur de l'élément majoritaire.
4. Montrer qu'à la fin de toute exécution complète, l'arène contient un unique troupeau de taille m.
5. Montrer que pour toue exécution partielle, le nombre de tests qui renvoient faux est au moins $2g + B$ et le nombre de tests qui renvoient vrai est au moins $t - Tr$.
6. Montrer que la complexité au pire des cas de tout algorithme de recherche d'élément majoritaire par test d'égalité est d'au moins $n + \lceil n/2 \rceil - 2$ tests.
7. Comparer cette borne inférieure à la complexité de l'algorithme de la Section 2.