MASTER 1 Info - RO
Algos probabilistes

1 Dépasser la médiane (4 points) (exercice partiel 2018)

Nous avons un tableau T de taille 1000001 contenant des valeurs numériques toutes différentes. Rappelons que la médiane est l'élément qui possède autant de valeurs inférieures à lui que de valeurs supérieures à lui (donc ici 500000 valeurs plus petites et 500000 valeurs plus grandes). Le problème est de trouver une entrée $T[i]$ strictement plus grande que la médiane. Mais notons aussi qu'à moins de la calculer, la médiane n'est pas connue (il faudrait alors certainement trier tous les éléments et prendre le 500001 ème)

Pour répondre à ce problème, une approche peut consister à déterminer le maximum dans tout le tableau. Ceci nécessite $n-1$ (donc 1000000) de comparaisons mais la valeur obtenue est bien sûr correcte, c'est à dire supérieure à la médiane.

1. Ce n'est toutefois pas nécessaire de déterminer le maximum de tout le tableau. De combien d'éléments peut-on se contenter pour être certain que leur maximum est bien supérieur à la médiane ? Et donc combien de comparaisons ?
2. Comme alternative, on propose l'algorithme probabiliste suivant :

   Upper-Half(T) k paramètre
   n = 1000001; k=10
   Pour i allant de 1 à k faire
   - j = Random(1,n);
   - $B[i] = T[j]$;

   Retourner Maximum(B[1..10])

   S'agit-il d'un algorithme de Monte-Carlo, de Las-Vegas ou d'Atlantic-City ? Justifier votre réponse.
3. Quelle est la probabilité que cet algorithme fournisse une réponse erronnée ? Combien de comparaisons effectue cet algorithme ?

2 Chercher l'élément répété

Un tableau T de taille N (N pair) contient $\frac{N}{2}$ éléments de valeurs distinctes et $\frac{N}{2}$ occurrences d'une même valeur distincte des $\frac{N}{2}$ premières. Par exemple (1,2,6,2,2,4,5,2,3,2) est un tel tableau avec $N = 10$. L'objectif est de trouver la valeur répétée dans le tableau (dans l'exemple trouver le 2).

1. Proposer un algorithme déterministe qui permet de résoudre ce problème.
2. Quel est le pire cas pour votre algorithme et quelle est alors la complexité ?
3. Proposer et écrire un algorithme de Monte-Carlo basé sur l'idée de tirer deux éléments aléatoires et de comparer leurs valeurs.

4. Quelle est la probabilité que votre algorithme trouve la valeur? Si N vaut 100, que vaut cette probabilité ?

5. Combien de fois faut-il répéter ce tirage aléatoire pour que la probabilité d'échec devienne inférieure à 10% (pour N=100) ?

3 Trouver la clé

On dispose de N clés distinctes triées en ordre croissant. Ces clés sont stockés sous forme de deux tableaux correspondant à une liste chainée. Le premier tableau VAL contient les valeurs des clés dans l'ordre où elles ont été crées, le second tableau NEXT donne l'indice de la clé suivante dans l'ordre croissant (il correspond au pointeur). Pour la plus grande clé NEXT[i] vaut 0. De plus une variable TETE donne l'indice de la première clé.

1. Compléter le tableau suivant et indiquer la valeur de TETE

I	1	2	3	4	5	6	7
VAL	2	13	4	1	5	21	8
NEXT	3	6	5	1	7	0	2

2. On s'intéresse à l'algorithme de recherche d'une clé X (que l'on suppose existante) dans le tableau dont on souhaite récupérer l'indice. L'algorithme déterministe proposé est le suivant :

   Fonction CHERCHE(X,i) :

   - Tant que $(X > VAL[i])$ faire $i := NEXT[i]$
   - Retourner(i);

   MAIN: $position := CHERCHE(X, TETE)$;

   Déterminer les complexités dans le meilleur des cas, dans le pire des cas et en moyenne de cet algorithme.

3. L'algorithme probabiliste proposé est le suivant:

   - $i := random(1, N)$;
   - $Y := VAL[i]$;
   - Si $(X < Y)$ Alors $Position := CHERCHE(X, TETE)$;
   - Sinon Si $(X > Y)$ Alors $Position := CHERCHE(X, NEXT[i])$;
   - Sinon Position := i;

   (a) Simuler l'algorithme sur l'exemple précédent en supposant que l'on cherche X=8 et que l'on a tiré i=1 ou i=2 ou i=7 (3 cas à simuler)

   (b) Est-ce un algorithme de Monte-Carlo, de Las-Vegas ou d'Atlantic-City ?

   (c) On voudrait calculer l'espérance de complexité de cet algorithme. Pour aider à ce calcul un peu complexe, commencez par établir le nombre de comparaisons à effectuer selon le rang de la clé cherchée et le rang de l'élément initial tiré au hasard.