TD d'Algorithmique et structures de données
Calcul de coûts en pire cas et en moyenne

Équipe pédagogique Algo SD

Soit T un tableau d'entiers. Soit n le nombre de cases de T.

1 Calcul du maximum dans un tableau

1. Écrire en python ou en pseudo-langage un algorithme qui calcule le maximum des éléments de T.
2. On choisit comme mesure de coût de l'algorithme le nombre de comparaisons. Donner le coût exact de votre algorithme, en fonction de n.
3. On choisit comme mesure de coût de l'algorithme le nombre d'affectations. Donner le coût en meilleur cas et le coût en pire cas de votre algorithme, en fonction de n.
4. Toujours dans le cas du nombre d'affectations, donner le coût en moyenne de votre algorithme. Pour cela nous supposerons que l'entrée est une permutation aléatoire des n éléments distincts choisie uniformément parmi toutes les permutations possibles. En particulier, le maximum a donc autant de chances de se trouver dans la première case de T que dans la deuxième, la troisième, ..., la n-ème.
5. On cherche à trouver en une seule passe le min et le max de T. Écrire une fonction résolvant le problème. On compte le nombre de conditionnelles exécutées. Quel est son coût au pire cas ? Au meilleur cas ? En moyenne ? Pouvez-vous trouver un algorithme exécutant en moyenne $n+O(\log(n))$ conditionnelles ?

2 Recherche séquentielle

1. Ecrire en python ou en pseudo-langage un algorithme qui cherche un élément `e` dans T. L'algorithme doit renvoyer un booléen : vrai si e est présent dans T, faux sinon.
2. On choisit comme mesure de coût de l'algorithme le nombre de comparaisons. Donner le coût en meilleur cas et le coût en pire cas de votre algorithme, en fonction de n.
3. On suppose qu'on sait que e est présent dans T. Donner dans ce cas le coût en moyenne de votre algorithme, sous hypothèse d'équirépartition des données.
4. On relâche l'hypothèse précédente. La probabilité que e soit présent dans le tableau est notée α. Donner dans ce cas le coût en moyenne de votre algorithme, toujours sous hypothèse d'équirépartition des données.

3 Recherche dichotomique

On suppose maintenant que T est **trié** par ordre croissant.

1. Ecrire en python ou en pseudo-langage un algorithme qui cherche un élément e dans T **par dichotomie**. L'algorithme doit renvoyer un booléen : vrai si e est présent dans T, faux sinon.
2. On choisit comme mesure de coût de l'algorithme le nombre de comparaisons. Donner le coût en meilleur cas et le coût en pire cas de votre algorithme, en fonction de n.
3. On suppose qu'on sait que e est présent dans T. Donner le coût en moyenne de votre algorithme, sous hypothèse d'équirépartition des données. Pour simplifier le raisonnement, on peut supposer que n est un $2^k - 1$.