TP N°2 : La recherche d'éléments

Remarque : Les tableaux et les graphes doivent être remis sur feuille à la séance suivante.

A. Recherche d'un élément :

On étudie dans cet exercice la complexité d'un algorithme de recherche d'un élément x dans un ensemble A.

1. Soit un tableau de n (n>=2) valeurs entières **non triées** :
   a. Écrire une fonction *rechElets_TabNonTriés* permettant de vérifier l'existence d'une valeur x donnée.
   b. Calculer la complexité théorique au meilleur cas et au pire cas.
2. Soit un tableau de n (n>=2) valeurs entières **triées :**
   2.1. Recherche séquentielle :
   a. Écrire une fonction *rechElets_TabTriés* permettant de vérifier l'existence d'une valeur x donnée.
   b. Calculer la complexité théorique au meilleur cas et au pire cas.

   2.2. Recherche Dichotomique :
   a. Écrire une fonction *rechElets_Dicho* permettant de vérifier l'existence d'une valeur x donnée en utilisant la méthode dichotomique.
   b. Calculer la complexité théorique au meilleur cas. La complexité au pire cas est de l'ordre de $O(\log n)$ (le calcul sera vu en cours).
3. Remplir le tableau suivant en donnant le temps de recherche[1] de chacune des trois fonctions précédentes :

	n / x	100000	200000	400000	600000	800000	1000000	1200000	1400000	1600000	1800000
EletsNonTriés	Meilleur cas										
	Pire cas										
EletsTriés	Meilleur cas										
	Pire cas										
EletsTriésDicho	Meilleur cas										
	Pire cas										

[1] Inclure les fonctions de time.h dans votre programme pour calculer le temps de recherche d'un élément en faisant varier la taille n des données.

4. Représenter dans un graphe les variations du temps d'exécution T(N) des 3 fonctions de recherches, quand le tableau est trié, dans le pire cas.
5. Que constatez-vous ?

B. Recherche du maximum et du minimum

Nous supposons ici que les valeurs de l'ensemble considéré[2] sont distinctes.

1. Approche Naïve :
   a. Ecrire la fonction *MaxEtMinA* de recherche du maximum et du minimum d'un ensemble non trié de n éléments.
   b. Quelle est sa complexité théorique au pire cas en nombre de comparaisons ?
2. Algorithme plus efficace :
   a. Ecrire la fonction *MaxEtMinB* de recherche du maximum et du minimum d'un ensemble non trié de n éléments selon le principe suivant :
      i. On compare par paire les éléments de l'ensemble. On met d'un côté les plus grand éléments (dans les cases paires du tableau) et de l'autre côté les plus petits (dans les cases impaires).
      Exemple :

5	2	7	3	1	8	4

2	5	3	7	1	8	4

      ii. On cherche le minimum parmi tous les plus petits éléments.
      iii. On cherche le maximum parmi tous les plus grands éléments. (si on a un nombre impair d'éléments, il ne faut pas oublier le n^{e} élément qui n'a pas été comparé dans la première phase i.)
   b. Quelle est sa complexité théorique au pire cas en nombre de comparaisons ?
3. Comparer les deux fonctions *MaxEtMinA et MaxEtMinB* en rajoutant un compteur dans les deux fonctions pour compter le nombre de comparaison.
4. Mesurer et comparer les complexités temporelles expérimentales des deux fonctions en variant n (nombre d'éléments).

[2] Pour l'exercice 2 remplir le tableau avec des valeurs aléatoires (fonction random)