TP N°1 : Test de primalité

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

L'objectif de ce TP est de tester différentes méthodes pour un même problème et aussi apprendre à mesurer le temps d'exécution d'un programme.
Il s'agit ici de tester si un nombre entier naturel est premier.
Quatre algorithmes sont proposés qu'il faut implémenter en langage C, puis les comparer en utilisant les fonctions de gestion du temps qui sont fournies dans la bibliothèque time.h.

Rappel : Un nombre entier naturel N est premier s'il n'a que 2 diviseurs : le nombre 1 et le nombre N lui-même.

Algorithme 1 (A1) : Approche naïve

1. Cette solution comporte une boucle dans laquelle on va tester si le nombre N est divisible par 2, 3, …, N-1. Ecrire l'algorithme correspondant.
2. Calculer la complexité théorique au pire cas de cet algorithme notée O(N).
3. Ecrire le programme correspondant.
   a. Vérifier que les nombres N proposés dans le tableau ci-dessous (1000003, 2000003, …) sont premiers.
   b. Mesurer les temps d'exécution T pour l'échantillon des nombres N ci-dessous et compléter le tableau :

N	1000003	2000003	4000037	8000009	16000057	32000011	64000031
T							

N	128000003	256000001	512000009	1024000009	2048000011
T					

   c. Que remarque-t-on sur les données de l'échantillon et sur les mesures obtenues ? (Indication : comparer chaque nombre N avec le suivant et chaque mesure du temps avec la suivante.)
   d. Comparer la complexité théorique et les mesures expérimentales.
   Les prédictions théoriques sont-elles compatibles avec les mesures expérimentales ?

e. Représenter avec 2 graphes les variations du temps d'exécution T(N) et les variations de la complexité au pire cas O(N). Utiliser pour cela un logiciel graphique comme Excel.

Algorithme 2 (A2) : Amélioration de l'approche naïve

On sait que tout diviseur i du nombre N vérifie la relation : $i \leq N/2$, avec $i \neq N$.

1. Développer un 2ème algorithme en tenant compte de cette propriété et reprendre les mêmes questions précédentes[1].
2. Comparer algorithmes A1 et A2 (représenter pour cela dans une même figure les graphes des 2 algorithmes). Lequel des 2 algorithmes est meilleur (ou plus performant) ?

Algorithme 3 (A3) :

Il existe une propriété mathématique sur les nombres entiers :

Propriété : Les diviseur d'un nombre entier N sont pour la moitié $\leq N^{1/2}$ ($=\sqrt{N}$) et pour l'autre moitié $> N^{1/2}$

1. Développer un 3ème algorithme A3 en tenant compte de cette propriété et reprendre les mêmes questions précédentes[1].
2. Comparer les 3 algorithmes. Lequel des 3 algorithmes est meilleur (ou plus performant) ?

Algorithme 4 (A4) :

Une autre amélioration possible consiste à tester si N est impair et dans ce cas dans la boucle, il ne faut tester la divisibilité de N que par les nombres impairs.

3. Développer un 4ème algorithme A4 en tenant compte de cette proposition et reprendre les mêmes questions précédentes[1].
4. Comparer les 4 algorithmes. Lequel des 4 algorithmes est meilleur (ou plus performant) ?

[1] Copier-coller le programme précédent puis modifier-le.