Nu Huyen Trang PHAM
Van Triet NGUYEN
Université de Technologie de Compiègne
February 8, 2019

Rapport mini-projet SR02

Crible d'Eratosthenes

Tâche 1:

1. Dérouler l'exécution de l'algorithme avec n=20

   floor(√n) = floor(√20) = 4

   i=2: j=4,6,8,10,12,14,16,18,20 A[j] = faux

   i=3: j=9,12,15,18 A[j] = faux

   i=4: rien faire car A[i] faux

2. La deuxième boucle commence à i^2 parce qu'il est sure que i^2 peut diviser par un nombre différent de 1 et i^2 (c'est i) donc i^2 n'est pas un nombre premier; on n'est pas sure que i est un nombre premier (par exemple i=4) et 0 n'est pas un nombre premier.
3. Le première boucle s'exécute jusqu'à √n parce que l'ensemble de diviseurs entiers positifs et différentes à 1 de n n'est pas plus grand que √n.

   Si √n n'est pas un entier, on arrondit sa valeur avec la fonction floor().

Tâche 2:

Nous avons transmis l'algorithme proposé dans l'énoncé en code C.

Cf: fichier tache2.c

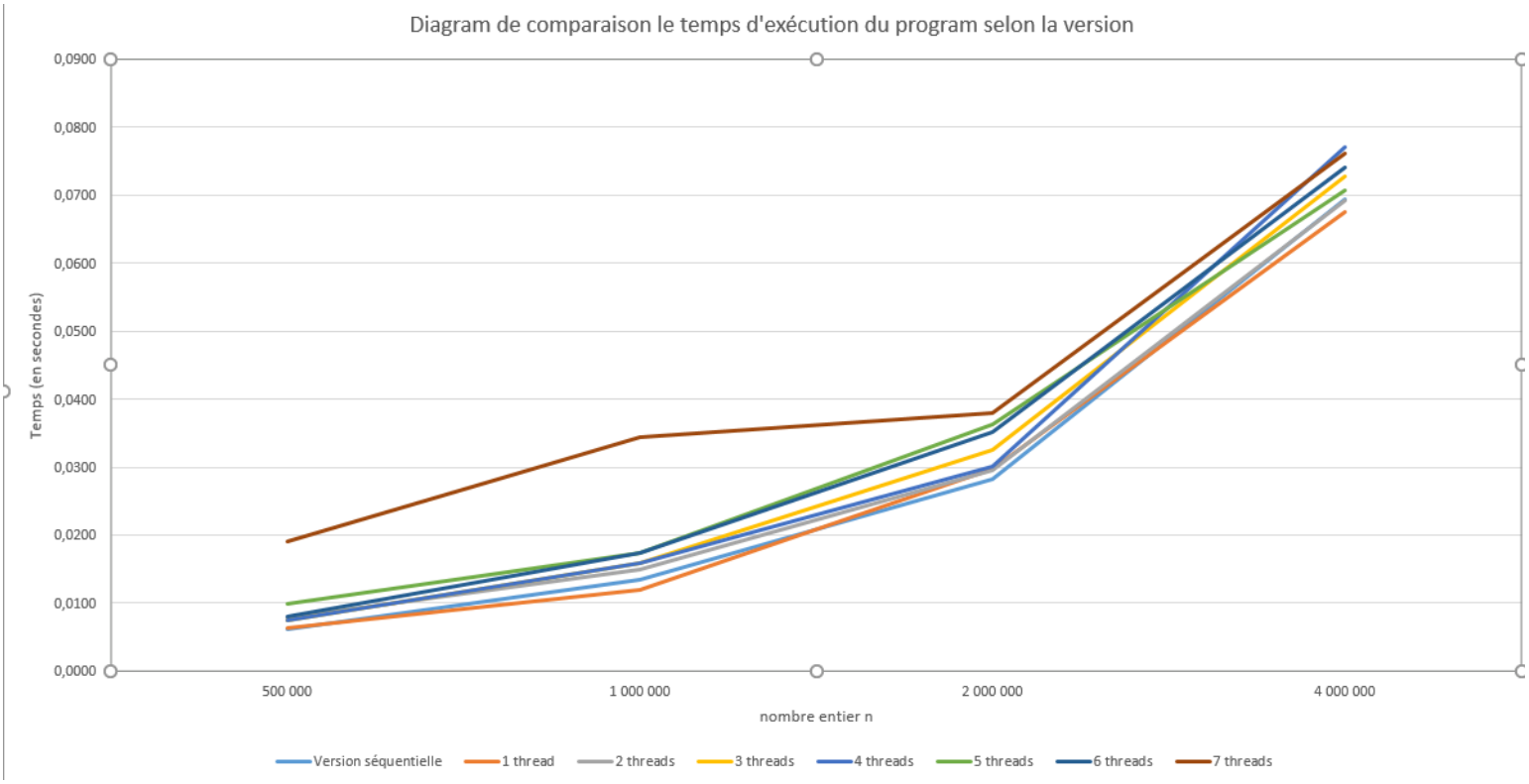
Tâche 3:

Cf: fichier tache3.c

Hypothèses faites: la plage de travail correspond à un seul i

Tâche 4:

Nous avons mesuré le temps d'exécution directement sur bash avec les instructions que nous avons enregistrées sous un fichier texte « *command_lines_tache4.txt* ».



Tâche 5:

Accélérer la boucle interne: Pour ce programme, nous avons mis les valeurs divisibles par 2 à faux (on est sure qu'ils ne sont pas les nombres premiers). Par la suite, nous avons effectué seulement l'élimination pour les chiffres impairs. Cf: fichier tache5_q1.c

Réduction de l'espace mémoire: Pour éviter de traiter des valeurs inutiles, nous ne traitons pas l'ensemble des valeurs paires et nous nous concentrons seulement sur les valeurs impaires. Cf: fichier tache5_q2.c

Bref, nous avons trouvé que l'accélération de la boucle interne augmente beaucoup la rapidité de l'exécution