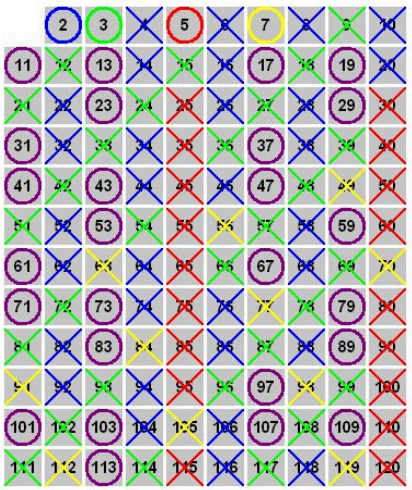
Programmation parallèle

Parallélisation du Crible

Exercice 1. Crible d'Ératosthène



Prime numbers

2	3	5	7
11	13	17	19
23	29	31	37
41	43	47	53
59	61	67	71
73	79	83	89
97	101	103	107
109	113		

Le Crible, un algorithme antique !

Le Crible d'Ératosthène est un algorithme très ancien utilisé pour trouver tous les nombres premiers jusqu'à un certain nombre entier N donné. L'idée principale de l'algorithme est de marquer tous les multiples de chaque nombre premier, à partir de 2, puis de passer au nombre suivant non marqué.

Implémentation

Écrivez tous les nombres naturels de 2 jusqu'à une borne supérieure N, et nous allons marquer ces nombres comme premiers ou non-premiers. Tous les nombres sont initialement non marqués.

- Le premier nombre non marqué est 2 : marquez-le comme premier et marquez tous ses multiples comme non-premiers.
- Le premier nombre non marqué est 3 : marquez-le comme premier et marquez tous ses multiples comme non-premiers.
- Le prochain nombre est 4, mais il a déjà été marqué, donc marquez 5 et ses multiples.
- Le prochain nombre est 6, mais il a déjà été marqué, donc marquez 7 et ses multiples.
- Les nombres 8, 9, 10 ont été marqués, donc continuez avec 11. • Et ainsi de suite.

Questions

1. Dans un premier temps, proposer un pseudo-co séquentiel pour l'algorithme du Crible.
2. Un premier niveau d'optimisation de cet algorithme serait d'appliquer une structure de données à listes chaînées, où les nombres marqués non-premiers seraient élagués de la liste de nombres au fur et à mesure que les nombres premiers correspondants sont identifiés. Ce qui permettrait de réduire drastiquement la longueur des itérations. Estimer la réduction progressive de l'espace de recherche et les nombres d'itération au fur et à mesure que l'algorithme est exécuté ?

3. Proposer un pseudo-code d'implémentation de cet algorithme avec l'utilisation de listes chaînées.
4. Comparer le gain théorique de temps obtenu en confrontant les deux algorithmes.
5. Cet algorithme peut-il être parallélisé ? Si oui, proposer une stratégie de division du problème en plusieurs opérations parallèles ?
6. En se basant sur les théories énoncées en cours, calculer f_p la proportion de code parallèle et en déduire f_s la proportion de code séquentiel.
7. Calculer le gain théorique obtenu en utilisant la méthode d'Amdahl.
8. Proposer un code sur MPI qui implémente votre stratégie.