Algorithmique PEIP2

TP2 : Autour du crible d'Ératosthène

1 Premières implémentation du crible

1. Écrire une fonction `Eratosthene(n)` qui renvoie la liste des nombres premiers inférieurs ou égaux à n. Cette fonction utilisera le crible d'Ératosthène décrit dans l'exercice 3 du TD2, sans utiliser les optimisations évoquées en cours. On pourra si besoin utiliser la fonction `sqrt` du module `math`

2. Tester cette fonction sur des valeurs de n croissantes en mesurant son temps d'exécution. On pourra utiliser pour cela la fonction `process_time()` du module `time`. Jusqu'à quelle valeur de n renvoie-t-elle un résultat en moins de 10 secondes ?

```
from time import process_time()

start = process_time()
# Calcul complexe
print("Le calcul complexe a pris", process_time()-start, "secondes")
```

3. Écrire une fonction `Eratosthene2(n)` qui utilise la première optimisation vue en cours : pour chaque nombre premier i, on retire les multiples de i à partir de i^2 au lieu de $2i$. Comparer le temps d'exécution de `Eratosthene2(n)` et de `Eratosthene(n)` sur des grandes valeurs de n.

4. Écrire une fonction `Eratosthene3(n)` qui traite à part le nombre premier $i = 2$, puis ne considère jamais les nombres pairs dans sa boucle principale. Comparer les temps d'exécution de votre trois fonctions. Ce résultat est-il surprenant ?

2 Implémentation linéaire du crible

On définit la classe `Case` qui contient deux valeurs `pred` et `succ`, initialisées à `None`.

```
class Case:
   pred=None
   succ=None
```

1. On veut construire une liste doublement chaînée qui initialement contient toutes les valeurs entre 0 et $n-1$. Pour cela, on utilise un tableau dont chaque élément est de type `Case`. Le premier élément a pour prédécesseur `None`, et le dernier élément a pour successeur `None`. Autrement, pour chaque indice i dans le tableau, le prédécesseur est l'indice $i-1$, et le successeur l'indice $i+1$. Écrire une fonction `ListeDoublementChainee(n)` qui initialise un tel tableau.

2. Écrire une fonction `RetirerElement(x,t)` qui retire l'élément `x` de la liste doublement chaînée `t`, de complexité $O(1)$. On utilisera pour cela les opérations décrites en cours. Attention au cas où `x` est le premier ou le dernier élément de `t` !

3. Étant donné un élément `x` présent dans `t`, et une valeur `xmax`, écrire une fonction `SousListe(x,xmax,t)` qui renvoie la liste des éléments compris entre `x` et `xmax` dans `t`.

4. Écrire une fonction `Eratosthene4(n)` qui implémente le crible d'Ératosthène de complexité linéaire décrit dans le cours . Pour chaque nombre premier i dans la boucle principale, on calcule la liste des multiples de i avec un nombre pas encore rayé compris entre i en n/i, puis on retire chacun de ces multiples de la liste

doublement chaînée des nombres pas encore rayés. Attention, il faut bien calculer tous les multiples avant de les supprimer de la liste, autrement on en oubliera (par exemple, si $i = 3$, on ne doit pas retirer 9 avant d'avoir calculé le multiple $3 * 9 = 27$).

5. Tester la fonction `Eratosthene4(n)`. Parvenez-vous à trouver une valeur de n pour laquelle son exécution est plus rapide que celle de `Eratosthène(n)` ? Essayez d'en estimer la valeur.