Programmation du crible d'Ératosthène

Obtenir la liste des nombres premiers inférieurs à un entier donné

On désigne sous le nom de crible d'Ératosthène (vers 276 av. J.-C. – vers 194 av. J.-C.) une méthode de recherche de tous les nombres premiers inférieurs ou égaux à un entier naturel $N \geqslant 2$ donné.

L'objet de ce chapitre est d'étudier la programmation de l'algorithme du crible. Nous allons tout d'abord élaborer un algorithme simple puis nous donnerons ensuite des améliorations permettant d'optimiser le temps d'exécution des programmes correspondants.

I. Introduction

1°) Rappel de la méthode du crible

On choisit un entier naturel $N \geqslant 2$.

On écrit la liste de tous les entiers naturels de 1 à N (N inclus).

- On élimine 1.
- On souligne 2 et on élimine tous les multiples de 2.
- Puis on fait de même avec 3.
- On choisit alors le plus petit nombre non souligné et non éliminé, ici 5, et on élimine tous ses multiples.
- On réitère le procédé jusqu'à la partie entière de la racine carrée de N.

Les nombres non éliminés sont tous les nombres premiers jusqu'à N.

Une animation du crible d'Ératosthène est proposée sur le site de Thérèse Éveilleau à l'adresse suivante : http://therese.eveilleau.pagesperso-orange.fr/pages/truc_mat/pratique/textes/crible_an.htm

2°) Démarche de base pour créer un algorithme

On entre un entier naturel $N \geqslant 2$.
On utilise une liste L supposée créée avant le lancement de l'algorithme et initialement vide.
On remplit la liste en y mettant dans cet ordre tous les entiers naturels de 1 à N.
Les nombres que l'on barre vont être remplacés par des 0 dans cette liste.
On obtient une nouvelle liste L contenant des entiers non nuls et des 0.
On affiche les entiers non nuls de cette liste. Il ne s'agit que de nombres premiers.

Commentaires :

• On va remplacer tous les nombres non premiers par un même nombre ; on choisit 0.
On pourrait choisit tout autre nombre non premier mais le choix apparaît dans une certaine mesure assez logique.

• Dans un premier temps, nous n'allons pas nous occuper de la condition d'arrêt à la partie entière de la racine carrée de N.

II. Un premier algorithme

1°) Version en langage intermédiaire

Entrée :
Saisir N

Traitement :
Remplir la liste L avec tous les entiers naturels de 1 à N dans cet ordre
Remplacer le nombre 1 de la liste L par 0
Pour *i* allant de 2 à N **Faire**
    Remplacer par 0 tous les termes de L dont le rang est un multiple de *i* strictement supérieur à *i*
FinPour

Sortie :
Afficher les éléments de L qui sont non nuls

2°) Exemple

On fait tourner l'algorithme « à la main » pour $N = 10$ afin de comprendre l'évolution de la liste L au fur et à mesure du déroulement de l'algorithme.

La liste créée au début du traitement est $L = [1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 ; 10]$.

1ère étape : On remplace 1 par 0.

La nouvelle liste devient alors $L = [0 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 ; 10]$.

2e étape : On passe à 2. On supprime les multiples de 2 autres que 2.

La nouvelle liste devient alors $L = [0 ; 2 ; 3 ; 0 ; 5 ; 0 ; 7 ; 0 ; 9 ; 0]$.

3e étape : On passe à 3. On supprime les multiples de 3 autres que 3.

La nouvelle liste devient alors $L = [0 ; 2 ; 3 ; 0 ; 5 ; 0 ; 7 ; 0 ; 0 ; 0]$.

4e étape : On passe à 0 (au rang 4 de la liste). Il n'y a rien à faire. Le terme de rang 8 est déjà 0.

La nouvelle liste devient alors $L = [0 ; 2 ; 3 ; 0 ; 5 ; 0 ; 7 ; 0 ; 0 ; 0]$.

5e étape : On passe à 5. On passe à 5 et on supprime les multiples de 5 autres que 5.

La nouvelle liste devient alors $L = [0 ; 2 ; 3 ; 0 ; 5 ; 0 ; 7 ; 0 ; 0 ; 0]$.

6e étape : On passe à 0 (au rang 6 de la liste). Il n'y a rien à faire.

La nouvelle liste devient alors $L = [0 ; 2 ; 3 ; 0 ; 5 ; 0 ; 7 ; 0 ; 0 ; 0]$.

7e étape : On passe à 7. On passe à 7 et on supprime les multiples de 7 autres que 7.

La nouvelle liste devient alors $L=[0\ ;\ 2\ ;\ 3\ ;\ 0\ ;\ 5\ ;\ 0\ ;\ 7\ ;\ 0\ ;\ 0\ ;\ 0]$.

8e étape : On passe à 0 (au rang 8 de la liste). Il n'y a rien à faire.

La nouvelle liste devient alors $L=[0\ ;\ 2\ ;\ 3\ ;\ 0\ ;\ 5\ ;\ 0\ ;\ 7\ ;\ 0\ ;\ 0\ ;\ 0]$.

9e étape : On passe à 0 (au rang 9 de la liste). Il n'y a rien à faire.

La nouvelle liste devient alors $L=[0\ ;\ 2\ ;\ 3\ ;\ 0\ ;\ 5\ ;\ 0\ ;\ 7\ ;\ 0\ ;\ 0\ ;\ 0]$.

10e étape : On passe à 0 (au rang 10 de la liste). Il n'y a rien à faire.

La nouvelle liste devient alors $L=[0\ ;\ 2\ ;\ 3\ ;\ 0\ ;\ 5\ ;\ 0\ ;\ 7\ ;\ 0\ ;\ 0\ ;\ 0]$.

On s'arrête là.

3°) Version en langage naturel opérationnelle pour la programmation

Il y a deux aménagements à faire en introduisant deux boucles « Pour » dans le traitement (à l'intérieur de la première boucle) et dans la sortie lors de l'affichage.

Dans le traitement, pour chaque valeur de *i*, il faut remplacer par 0 tous les termes de L dont le rang est un multiple de *i* strictement supérieur à *i* les multiples de $L(i)$. Cela, consiste à remplacer par 0 tous les termes $L(i\times j)$ avec *j* tel que $2\leqslant i\times j\leqslant N$ c'est-à-dire que $2\leqslant j\leqslant \frac{N}{i}$.

On va faire une « double » boucle.

Entrée :
Saisir N

Traitement :
Remplir la liste L avec tous les entiers naturels de 1 à N dans cet ordre
$L(1)$ prend la valeur 0
Pour *i* allant de 2 à N **Faire**
  Pour *j* allant de 2 à $E\left(\frac{n}{i}\right)$ **Faire**
    $L(ij)$ prend la valeur 0
  FinPour
FinPour

Sortie :
Pour *k* allant de 1 à N **Faire**
  Si $L(k)\neq 0$
    Alors afficher $L(k)$
  FinSi
FinPour

3°) Programmation sur calculatrice

La programmation de cet algorithme ne pose pas de problème.

• Dans le programme, la liste L sera notée L1.

• On ajoute une instruction en début pour effacer la liste : ClrList ou EffListe
Pour cela, appuyer sur la touche stats puis aller dans EDIT et sélectionner le choix 4 : EffListe ou y
À chaque nouvelle utilisation du programme, la liste précédente est effacée.

• Le remplissage de la liste L au début du traitement peut se faire de deux manières :
- La première manière consiste à utiliser une boucle « Pour » ;
- La deuxième manière consiste à utiliser une fonction importante des listes : seq(ou suite(.
Cette fonction sert à créer des suites finies.
Ici l'instruction sera : suite(K,K,1,N,1) → L1 ou seq(K,K,1,N,1) → L1.
Pour cela, appuyer sur les touches 2nde stats puis aller dans OP et sélectionner le choix 5.

• La calculatrice possède une limitation technique. Pour les calculatrices TI, les listes sont limitées et l'entier naturel N saisi en entrée doit être inférieur ou égal à 999.

Le programme correspondant sur calculatrice TI :

```
: Prompt N
: EffListe L1
: suite(K,K,1,N,1) → L1
: 0 → L1(1)
: For(I,2,N)
: For (J,2,N/I)
: 0 → L1(I*J)
: End
: End
: For(K,1,N)
: If L1(K)≠0
: Disp L1(K)
: End
```

III. Améliorations de l'algorithme

Dans ce paragraphe, on s'intéresse à des améliorations permettant une optimisation du temps d'exécution.

1°) Une première amélioration

Dans la boucle « Pour », il est inutile de considérer les multiples des termes de la liste qui sont égaux à 0.
On introduire donc une condition.

On peut donner une version en langage intermédiaire.

Entrée :
Saisir N

Traitement :
Remplir la liste L avec tous les entiers naturels de 1 à N
Remplacer le nombre 1 de la liste L par 0
Pour *i* allant de 2 à N **Faire**
  Si $L(i) \neq 0$
    Alors
    Remplacer par 0 tous les termes de L dont le rang est un multiple de *i* strictement supérieur à *i*
  FinSi
FinPour

Sortie :
Afficher les éléments de L qui sont non nuls

2°) Deux améliorations sur les tests d'arrêts

• Dans l'instruction « **Pour** *i* allant de 2 à N **Faire** », il n'est pas nécessaire que *i* aille de 2 à N.

Il suffit que *i* reste inférieur à $\sqrt{N}$.

On peut donc faire varier *i* entre 2 et $E\left(\sqrt{N}\right)$.

En réduisant ainsi les valeurs entre lesquelles varie *i*, on diminue le temps d'exécution du programme.

« **Pour** *i* allant de 1 à $E\left(\sqrt{N}\right)$ **Faire** »

• Dans l'instruction « **Pour** *j* allant de 2 à $E\left(\frac{n}{i}\right)$ », on peut faire démarrer *j* à *i*.

« **Pour** *j* allant de *i* à $E\left(\frac{n}{i}\right)$ »,

Sur calculatrice TI, on obtient alors le programme suivant :

```
: Prompt N
: EffListe L1
: suite(K,K,1,N,1) → L1
: 0 → L1(1)
: For(I,2,√(N))
: If L1(I)≠0
: For (J,I,N/I)
: 0 → L1(I*J)
: End
: End
: End
: For(K,1,N)
: If L1(K)≠0
: Disp L1(K)
: End
```

3°) Utilisation d'une deuxième liste

On peut faire intervenir une deuxième liste pour rassembler tous les nombres premiers de la première liste c'est-à-dire tous les nombres non nuls de la liste après la boucle ;
C'est cette liste que l'on affichera à la fin de l'algorithme.
Pour le remplissage de cette liste, il y a deux options :
- soit on le fait dans la boucle « Pour » ;
- soit on le fait après la boucle « Pour ».

On va reprendre l'algorithme simple donné au début dans le paragraphe **II** en introduisant une liste L'. Les modifications données sur cet algorithme sont facilement transposables pour les algorithmes améliorés.

Version modifiée avec la première option :

Entrée :
Saisir N

Traitement :
Remplir la liste L avec tous les entiers naturels de 1 à N
Remplacer le nombre 1 de la liste L par 0
Pour *i* allant de 2 à N **Faire**
  Si $L(i) \neq 0$
    Alors
    Ajouter $L(i)$ à la liste L'
    Remplacer par 0 tous les termes de L dont le rang est un multiple de *i* strictement supérieur à *i*
  FinSi
FinPour

Sortie :
Afficher L'

Version modifiée avec la deuxième option :

Entrée :
Saisir N

Traitement :
Remplir la liste L avec tous les entiers naturels de 1 à N
Remplacer le nombre 1 de la liste L par 0
Pour *i* allant de 2 à N **Faire**
    Si $L(i) \neq 0$
        Alors
        remplacer par 0 tous les termes de L dont le rang est un multiple de *i* strictement supérieur à *i*
    FinSi
FinPour
Créer une liste L' avec tous les termes non nuls de L

Sortie :
Afficher L'

Appendice : Programme écrit en avril 2016 par Théo Manfredi élève de TS2 très doué en programmation

```
: ClrList L1
: ClrList L2
: Input N
: For(I,2,N)
: I→L1(Dim(L1)+1)
: End
: For(I,1,N−1)
: If L1(I)≠0
: Then
: L1(I)→L2(Dim(L2)+1)
: For(A,I+L1(I),N,L1(I))
: 0→L1(A)
: End
: End
: End
```