



UNIVERSITÉ
CÔTE D'AZUR

Nombres premiers

Algo & Prog avec R

A. Malapert, B. Martin, M. Pelleau, et J.-P. Roy

18 novembre 2021

Université Côte d'Azur, CNRS, I3S, France

`firstname.lastname@univ-cotedazur.fr`

Qu'est-ce qu'un nombre premier ?

Nombre premier

Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs.

- ▶ Ces deux diviseurs sont 1 et le nombre considéré, puisque tout nombre a pour diviseurs 1 et lui-même (comme le montre l'égalité $n = 1 \times n$).
- ▶ Les nombres premiers étant ceux qui n'en possèdent aucun autre.
- ▶ Selon cette définition, les nombres 0 et 1 ne sont pas premiers.

Wikipedia.

Calculons le vecteur des nombres premiers inférieurs à n !

Détermination des nombres premiers inférieurs à n

Comme d'habitude, on va construire une itération.

Passer à l'étape suivante (**ITÉRATION**)

Étant en possession de `prems`, un vecteur contenant les nombres premiers de $[0, d]$.

- ▶ Il suffira donc d'ajouter 1 à d ,
- ▶ puis d'ajouter d à `prems` si d est premier (**test de primalité**).

Détecter si le calcul est terminé (**TERMINAISON**)

On aura terminé lorsque d sera égal à n puisqu'alors `prems` contiendra tous les nombres premiers inférieurs à n .

Trouver les valeurs initiales des variables (**INITIALISATION**)

Au début du calcul, je peux prendre `prems` vide et $d = 0$.

Test de primalité I

Pour savoir si un nombre $n \geq 2$ est premier, il suffit donc d'examiner les nombres entiers d de $[2, n - 1]$ à la recherche d'un diviseur de n .

- ▶ Si l'on en trouve un, on s'arrête au premier avec le résultat FALSE.
- ▶ Sinon, le résultat sera TRUE.

```
EstPremier <- function(n) { # version I
  if(n < 2) return(FALSE) # 0 et 1 ne sont pas premiers
  d <- 2 #le premier diviseur non trivial
  while( d < n) {
    if( n %% d == 0) return(FALSE) # Échappement
    d <- d + 1
  }
  return(TRUE)
}
```

```
> EstPremier(1003)
[1] FALSE
> EstPremier(2003)
[1] TRUE
```

Comment améliorer ce test ?

Il est appelé $n - 1$ fois pour déterminer les nombres premiers inférieurs à n

Test de primalité II

En fait, il suffit de tester la parité et d'examiner les nombres entiers impairs dans $[3, \sqrt{n}]$ à la recherche d'un diviseur de n .

```
EstPremier <- function(n) { # version II
  if(n < 2) return(FALSE) # 0 et 1 ne sont pas premiers
  if( n %% 2 == 0) return(ifelse( n == 2, TRUE, FALSE)) #
    test de parité
  d <- 3 # le deuxième diviseur non trivial
  m <- floor(sqrt(n)) # on calcule la racine une seule fois
  while( d <= m) { # on compare avec la racine
    if( n %% d == 0) return(FALSE) # Échappement
    d <- d + 2 # on itère sur les nombres impairs
  }
  return(TRUE)
}
```

Performance des tests de primalité

Comparons les performances des deux versions du test de primalité sur des nombres de Mersenne premiers.

Les nombres de Mersenne sont de la forme : une puissance de 2 moins 1.

```
> system.time(replicate(50, EstPremier(2**19 - 1)))
```

utilisateur	système	écoulé
0.018	0.000	0.019

n	version I	version II
$2^{13} - 1$	152	2
$2^{17} - 1$	1145	13
$2^{19} - 1$	4260	19

Table 1 – Durée du test de primalité en millisecondes.

Construction du vecteur des premiers inférieurs à n

Seuls les nombres impairs peuvent être premiers.

```
Premiers <- function(n) { # Version I
  if(n < 2) return(numeric(0))
  prems <- c(2)
  if(n > 2) {
    for(i in seq(3, n, 2)) { ## erreur de seq si n = 2
      if(EstPremier(i)) prems <- append(prems, i)
    }
  }
  return(prems)
}
```

```
Premiers <- function(n) { # Version II
  if(n < 2) return(numeric(0))
  if(n == 2) return(c(2))
  return(c(2, Filter(EstPremier, seq(3, n, 2))))
}
```

Seuls les nombres impairs peuvent être premiers.

```
Premiers <- function(n) { # Version III
  if(n < 2) return(numeric(0))
  if(n == 2) return(c(2))
  require(parallel) # chargement d'un package
  prems <- seq(3, n, 2)
  ind <- mcmapply(prems, FUN = EstPremier, mc.cores = 8)
  return(c(2, prems[ind]))
}
```

Performance du calcul des premiers inférieurs à n

Comparons les performances des trois versions de la construction du vecteur des nombres premiers inférieurs à n pour quelques valeurs de n .

```
> system.time(replicate(5, Premiers(10**5)))  
utilisateur      système      écoulé  
      0.018         0.000         0.019
```

n	version I	version II	version III
10^4	0.08	0.07	0.13
10^5	1.85	1.44	0.82
10^6	57.25	32.20	10.95

Table 2 – Durée du calcul des premiers en secondes.

Crible d'Ératosthène (à faire en TP)

Il suffit d'examiner les diviseurs premiers d'un nombre.

Questions?

Retrouvez ce cours sur le site web

www.i3s.unice.fr/~malapert/R