

Test de Primalité

Jean-Didier Pailleux - Robin Feron - Romain Robert - Damien Thenot -
Maxence Joulin

UVSQ

09/01/2018

- **Nombre Premier** : Entier divisible par 1 et lui-même.
pause
- **Test Probabiliste** : Test avec marge d'erreur très faible mais rapide.
- **Test Deterministe** : Test fiable mais plus lent.
- 2^{64} : Taille maximale des nombre a tester \Rightarrow Unsigned long long int
- **Nombre Hautement Composé** : Entier qui possède strictement plus de diviseur que les nombres qui le précède

- 1 Architecture
- 2 Outils et Langage de Programmation
- 3 Analyse des Résultats
- 4 Bilan Technique
- 5 Organisation interne du groupe
- 6 Conclusion

- 1 Architecture
- 2 Outils et Langage de Programmation
- 3 Analyse des Résultats
- 4 Bilan Technique
- 5 Organisation interne du groupe
- 6 Conclusion

Qui ?

Plan

- 1 Architecture
- 2 Outils et Langage de Programmation
- 3 Analyse des Résultats
- 4 Bilan Technique
- 5 Organisation interne du groupe
- 6 Conclusion

Qui ?

- 1 Architecture
- 2 Outils et Langage de Programmation
- 3 Analyse des Résultats**
- 4 Bilan Technique
- 5 Organisation interne du groupe
- 6 Conclusion

Fonctionnement du projet :

- Script appelé avec `./test.sh`

Options :

a : Tous les algorithmes

e : Euclide (computation bound)

m : Crible d'ératosthène

i : Miller-Rabin

H : Nombre hautement composé def

k : AKS

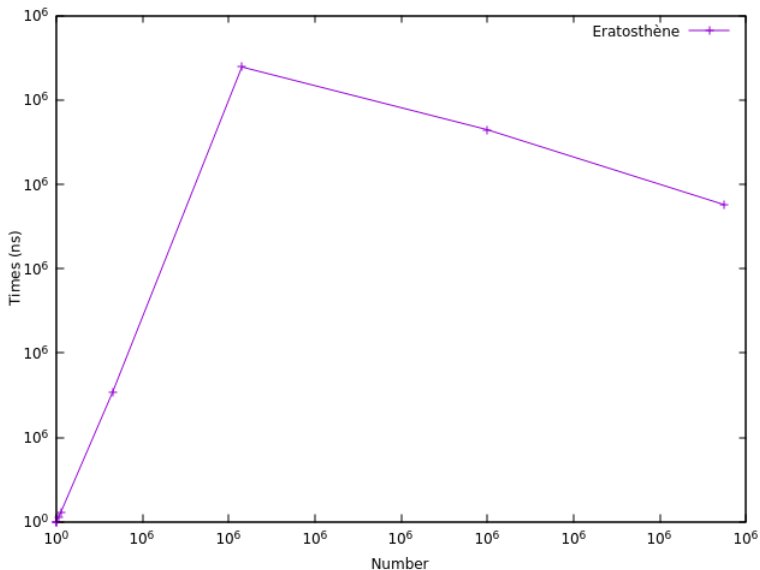
o : Modulo (computation bound)

p : Pocklington

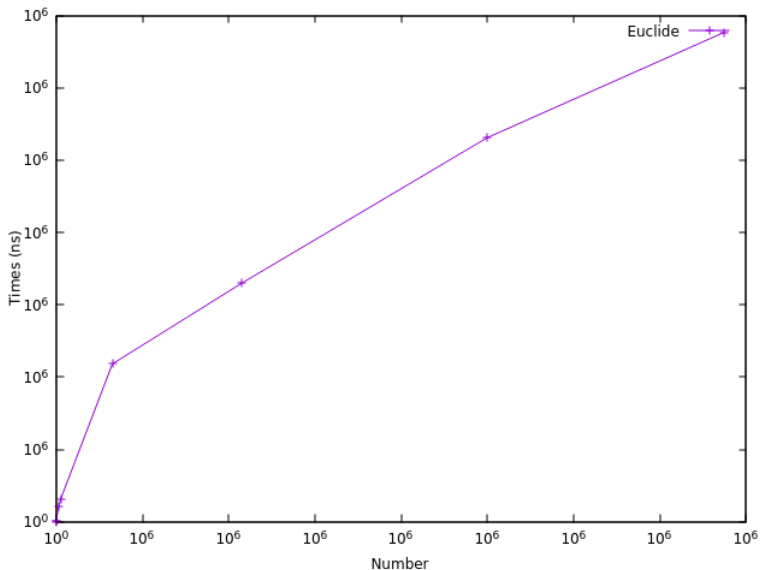
h : Nombre hautement composé naïve

- Lequel utiliser ?
- Itération ?
- Combien de nombre ?
- Donner les nombres

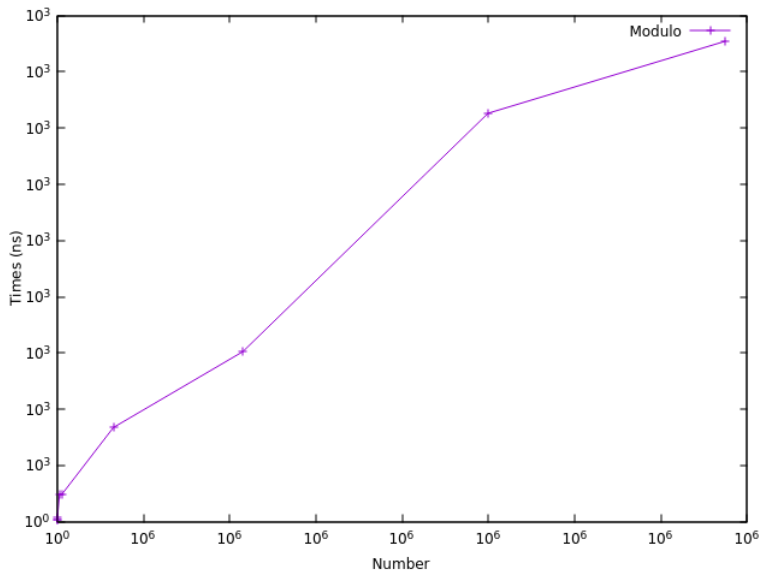
Evolution du temps d'exécution de Eratosthène/Memory Bound :



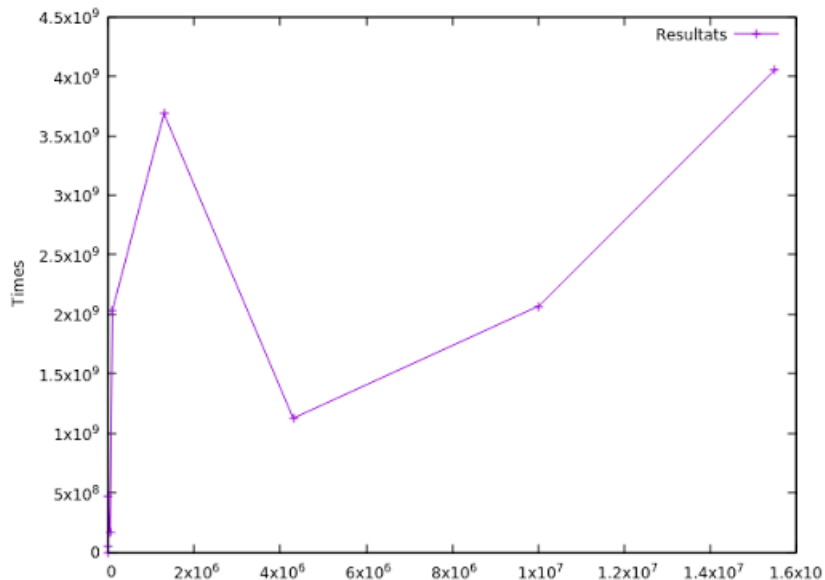
Evolution du temps d'exécution de Euclide/Computation Bound :



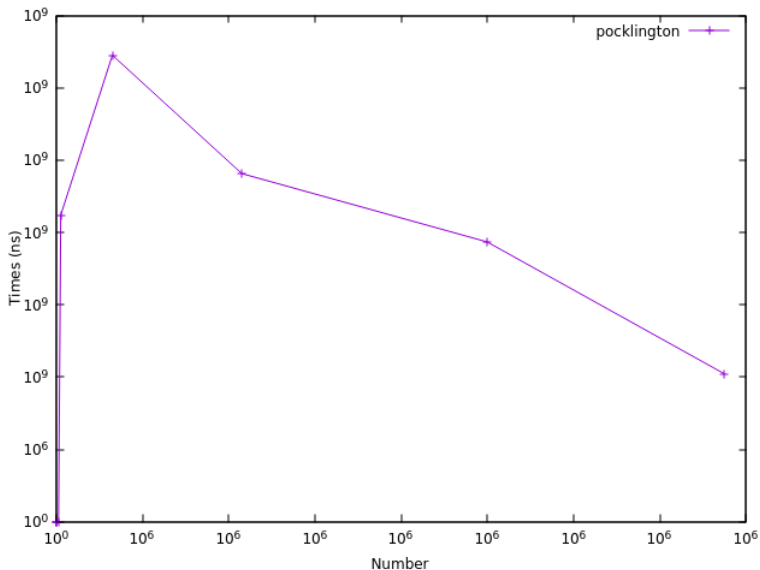
Evolution du temps d'exécution de Modulo/Computation Bound :



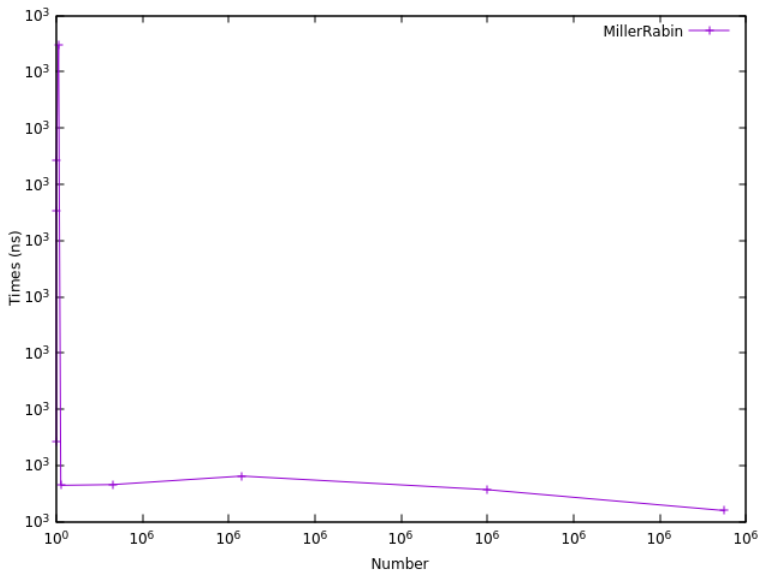
Evolution du temps d'exécution d'AKS :



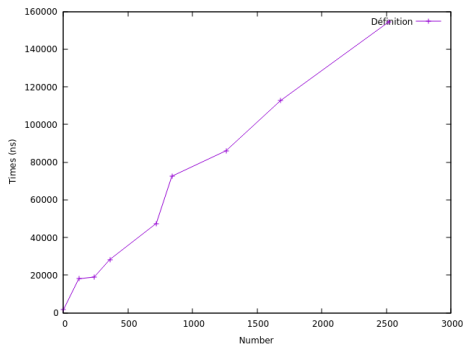
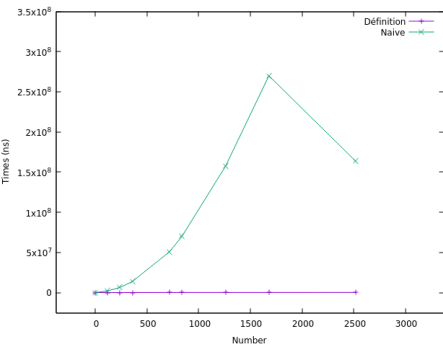
Evolution du temps d'exécution de Pocklington :



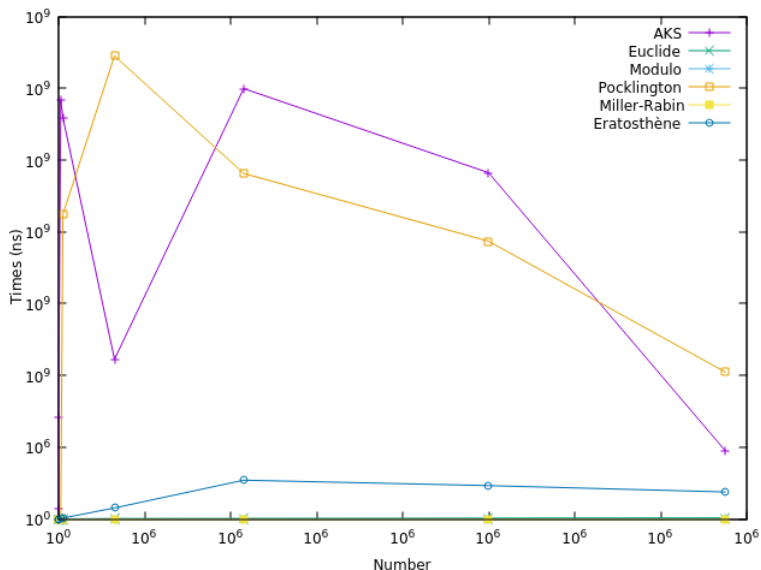
Evolution du temps d'exécution de Miller-Rabin :



Evolution du temps d'exécution de Hautement composé (méthode naïve et définition) :



Analyse des Résultats



- 1 Architecture
- 2 Outils et Langage de Programmation
- 3 Analyse des Résultats
- 4 Bilan Technique**
- 5 Organisation interne du groupe
- 6 Conclusion

- **crible d'Eratosthène/ Memory Bound** : Création d'un tableau de taille $N+1$ dans le crible \Rightarrow limité au niveau de la RAM pour N grand sur nos machines. Complexité de N pour le remplissage de la liste `memory_bound`.
- **Computation Bound** : Effectue $\sqrt{2^{\log_2(n)}}$ divisions euclidiennes \Rightarrow Exécution en temps exponentiel
- **Pocklington** : Limite causé par la factorisation du nombre $N-1 \Rightarrow$ factorisation très longues pour N très grand.

- **Miller-Rabin** : Résultats faux dans certains cas + Nombre d'itérations demandé élevé pour un meilleur résultat => augmentation du temps d'exécution.
- **AKS** : Avantage : Sa complexité en $\log(n)^{12}$.
Inconvénient : Utilisation de NTL qui effectue des vérifications superflue + Implémentation compliquée.
- **Hautement Composé** : N calcul du nombre de diviseurs d'un nombre => exécution très lente pour la méthode naïve.

- 1 Architecture
- 2 Outils et Langage de Programmation
- 3 Analyse des Résultats
- 4 Bilan Technique
- 5 Organisation interne du groupe
- 6 Conclusion

Tableau de répartition du travail :

Tâches	Jean-Didier	Maxence	Romain	Robin	Damien
Eratosthène/Memory Bound	x				
Euclide/Computation Bound		x			
AKS			x		
Pocklington					x
Miller-Rabin				x	
Highly Composite	x				
Cmake	x	x			
Script	x	x			

- 1 Architecture
- 2 Outils et Langage de Programmation
- 3 Analyse des Résultats
- 4 Bilan Technique
- 5 Organisation interne du groupe
- 6 Conclusion**

Qui ?