M1103 Amphi 7 : Tri

Thomas Nowak

Université Paris-Sud

La dernière fois

Tri

Tri

- l'algorithme de tri par sélection a une complexité en temps de $O(L^2)$ où $L =$ `vec.size()`
- si on fait un nombre élevé de recherches dans un vecteur, il peut être mieux de trier le vecteur avant de faire les recherches (car les recherches seront beaucoup plus rapides)
- peut-on améliorer la complexité $O(L^2)$ du tri ?

Tri fusion

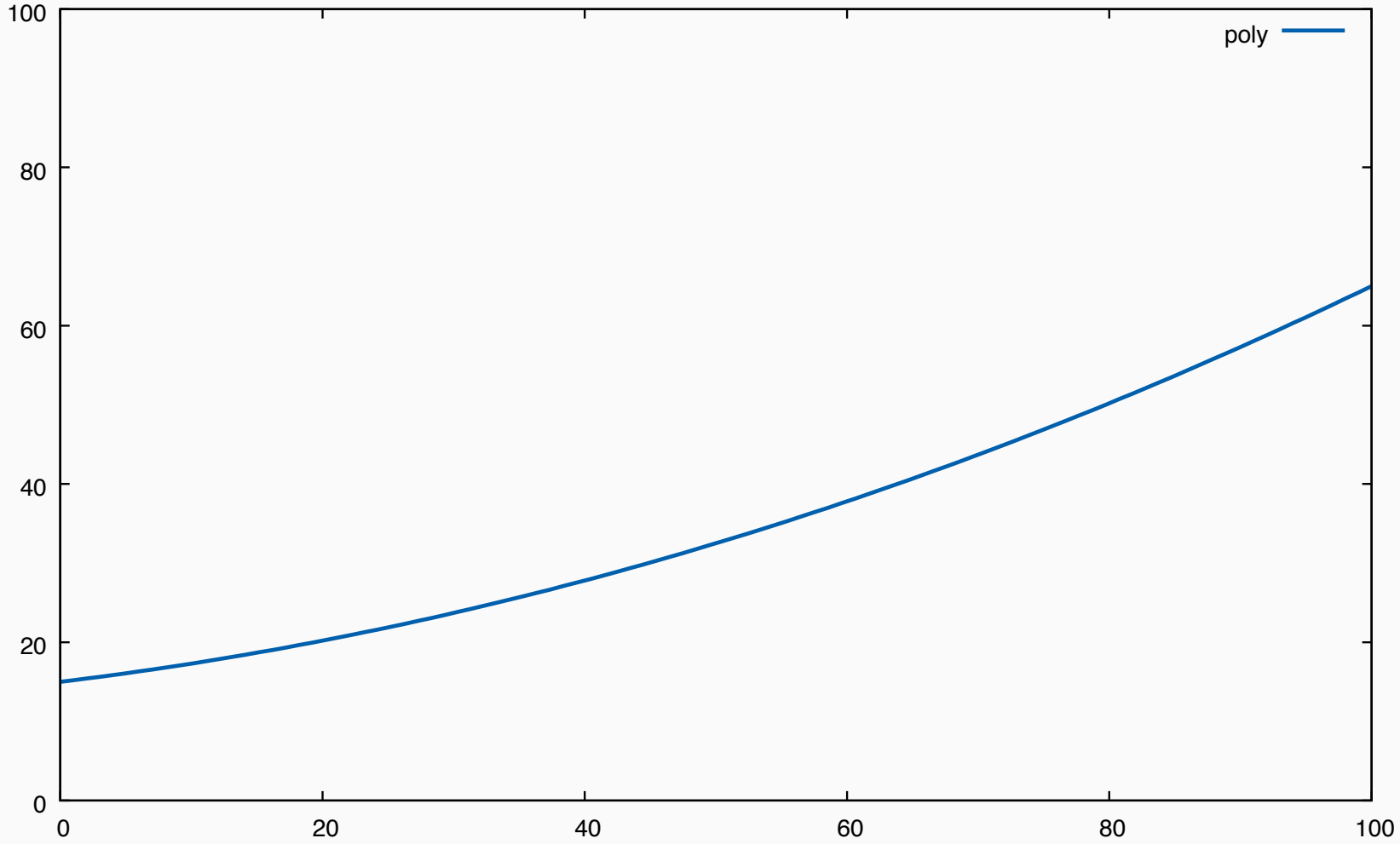
- même principe que pour la recherche dichotomique : couper le vecteur en deux, puis travailler sur des vecteur de taille plus petite

1. couper le vecteur en deux parties de taille $\approx L/2$ (si possible)
2. trier chaque partie indépendamment
3. fusionner les deux parties triées

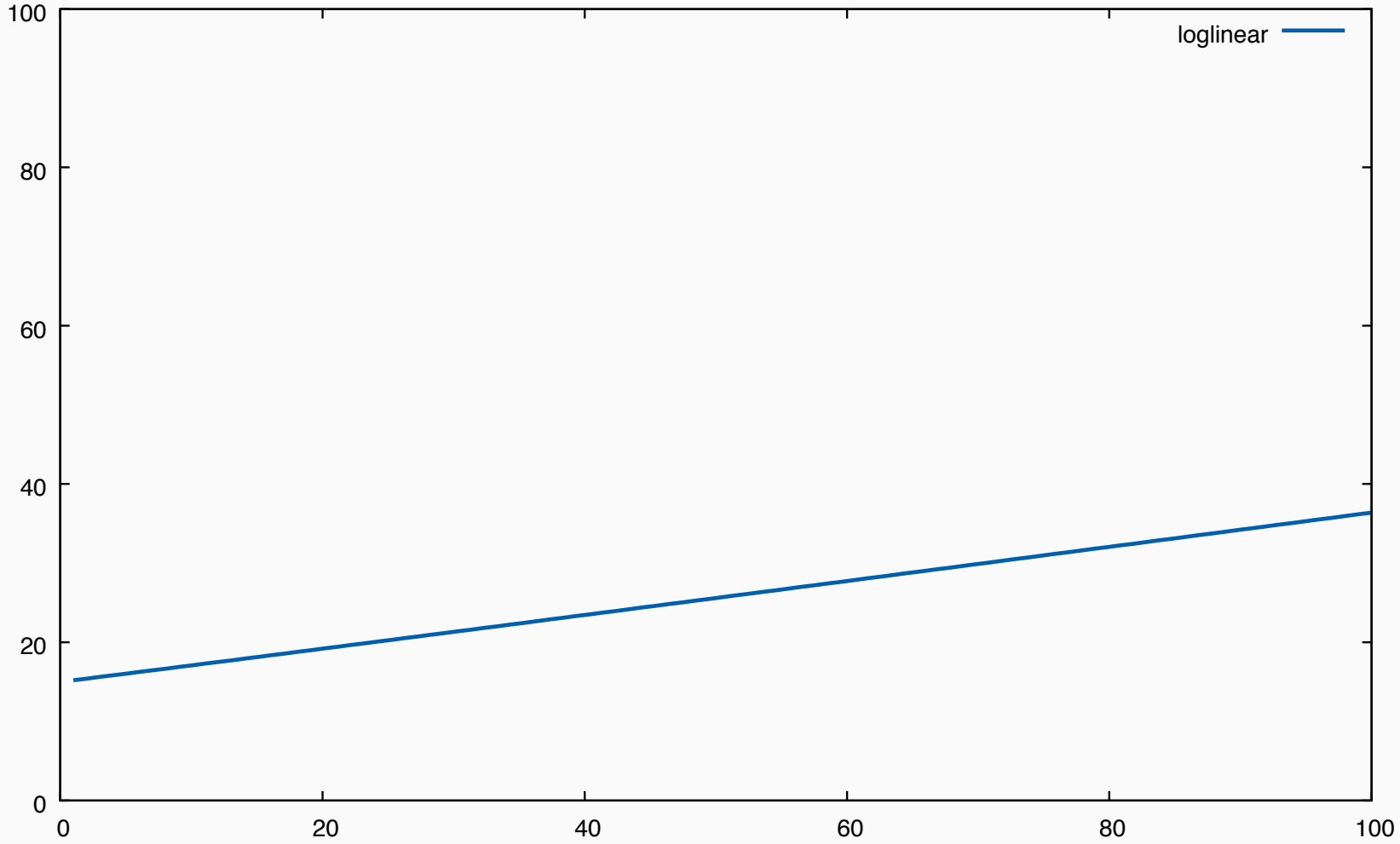
Tri fusion

- l'implémentation la plus directe est une fonction qui s'appelle elle-même (!)
- complexité en temps = $O(\log L)\times$ coût pour la fusion dans chaque niveau
- fusion de deux vecteurs triés en temps linéaire $O(L_1 + L_2)$
- (voir feuille 1, exercice 2)
- $\rightarrow O(L \log L)$

Complexité quadratique



Complexité log-linéaire



Conclusion

- l'idée de diviser en deux le vecteur marche aussi pour le tri
- algorithme de *tri fusion*
- temps $O(L \log L)$ au lieu de $O(L^2)$
- cette complexité en temps est, en fait, optimale

Questions