

Lecture 9

Algorithmes

Le Bon, la Brute et le Truand

Cours Optimisation Discrète 21 avril 2011

Friedrich Eisenbrand
EPFL

123

Notes

Notation grand O

Motivation

Les algorithmes nécessitent un certain nombre d'opérations de base et donc de temps. Plus l'entrée est grande, plus il faut d'opérations. Comment évaluer la qualité d'un algorithme ?

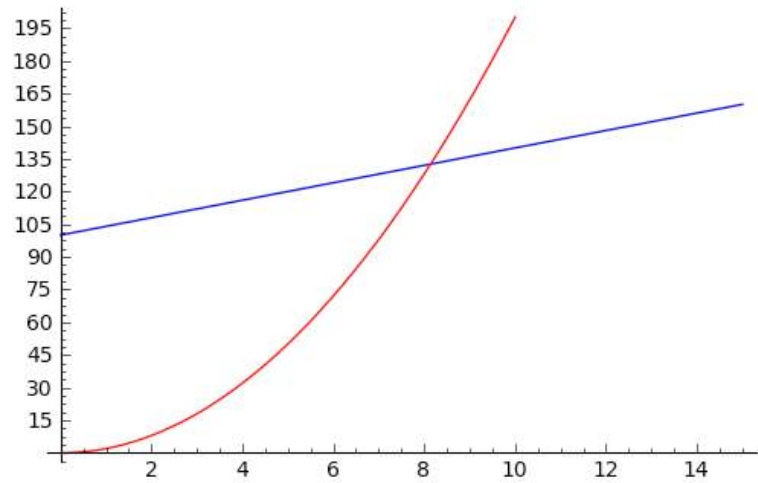
Exemple

- ▶ Un algorithme A effectue $4 \cdot n + 100$ opérations de base sur une entrée de longueur n .
- ▶ Un algorithme B effectue $2 \cdot n^2$ opérations de base sur une entrée de longueur n .
- ▶ L'algorithme B est plus rapide pour les petites entrées.
- ▶ L'algorithme A est plus rapide dès que la longueur de l'entrée dépasse 9.

124

Notes

Grphe des deux fonctions



125

Notes

Notation grand O

Définition

Soient $f, g : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ des fonctions. On écrit $f = O(g)$ s'il existe une constante $c \in \mathbb{R}$ et un nombre $N \in \mathbb{N}$ tels que

$$f(n) \leq c \cdot g(n) \quad \text{si } n \geq N.$$

On écrit $f = \Omega(g)$ if $g = O(f)$.

Exemple

- ▶ $5n^3 + n + 7 = O(n^3)$
- ▶ $\sin(n) = O(1)$
- ▶ $n^k = O(2^n)$ pour tout $k \in \mathbb{N}$

126

Notes

Analyse des algorithmes de tri

Trier une liste de nombres

Entrée : n entiers naturels a_1, \dots, a_n

But : Réordonner les nombres (c'est-à-dire trouver une permutation $\pi \in S_n$) de telle façon que $a_{\pi(1)} \leq \dots \leq a_{\pi(n)}$.

Modèle de calcul

Nous supposons que les opérations de base incluent la comparaison de deux nombres (opération en temps constant).

127

Notes

La brute

Tri à bulles

```
def bubble(List, i):  
    k = i  
    while k >= 1 and List[k] < List[k-1] :  
        List[k-1:k+1] = reversed(List[k-1:k+1])  
        k = k - 1  
  
def bubblesort(List):  
    for k in range(len(List)):  
        bubble(List, k)
```

128

Notes

Analyse du tri à bulles

Théorème

Pour trier n nombres, le tri à bulle nécessite $O(n^2)$ opérations de base. Il existe une famille de listes L_n , chacune de longueur n , sur lesquelles le tri à bulles effectue $\Omega(n^2)$ opérations.

129

Notes

Le bon

Tri fusion

```
def mergesort(List):
    if len(List) < 2:
        return List
    else:
        m = len(List) / 2
        Left = mergesort(List[:m])
        Right = mergesort(List[m:])
        return merge(Left, Right)
```

130

Notes

Le bon

Procédure de fusion

```
def merge(Left, Right):  
    List = []  
    L, R = 0, 0  
  
    while L <= len(Left)-1 and R <= len(Right)-1 :  
        if Left[L] <= Right[R] :  
            List.append(Left[L])  
            L = L+1  
        else :  
            List.append(Right[R])  
            R = R+1  
    List += Left[L:]  
    List += Right[R:]  
    return List
```

131

Notes

Analyse du tri fusion

Théorème

Pour trier n nombres, le tri fusion nécessite $O(n \log n)$ opérations de base.

Théorème

Un algorithme qui n'obtient de l'information sur les nombres de l'entrée qu'en comparant deux nombres (modèle par comparaisons) nécessite $\Omega(n \log n)$ comparaisons pour trier une suite de n nombres.

132

Notes

Le truand

Code spaghetti

Notes

133

Algorithmes en temps polynômial

Définition (Algorithme en temps polynômial)

Un algorithme s'exécute en temps polynômial s'il existe une constante k telle que le nombre d'opérations de base que l'algorithme effectue est majoré par $O(n^k)$, où n est la longueur de l'entrée de l'algorithme.

Exemple

Le tri fusion et le tri à bulles sont des algorithmes en temps polynômial. Mais le tri fusion est meilleur en pire cas, et aussi en pratique.

Exercice de 5 minutes

L'algorithme suivant est-il en temps polynômial ?

Entrée : Entier naturel k en notation unaire

$s = 2$

Répéter k fois : $s = s^2$

Supposer que s est stocké en notation binaire !

Notes

134

L'algorithme du simplexe est-il en temps polynômial ?

Nombre d'opérations arithmétiques

- ▶ Chaque itération de l'algorithme du simplexe nécessite un nombre polynômial d'opérations arithmétiques.
- ▶ On peut montrer que si les nombres rationnels sont sans facteurs communs (c'est-à-dire que leurs numérateur et dénominateur sont premiers entre eux) pendant l'élimination de Gauss-Jordan, alors tous les nombres sont de longueur polynômiale durant les étapes intermédiaires de l'élimination de Gauss-Jordan.
- ▶ On ne sait pas s'il existe une variante de l'algorithme du simplexe qui n'effectue qu'un nombre polynômial (en n et m) d'opérations.

135

Notes

L'algorithme du simplexe est-il en temps polynômial ?

Taille de l'entrée

- ▶ Taille de l'entier a : $\lceil \log(|a| + 1) \rceil$
- ▶ Taille du nombre rationnel p/q avec $\gcd(p, q) = 1$: $\text{taille}(p) + \text{taille}(q)$
- ▶ Taille de la matrice $A \in \mathbb{Q}^{m \times n}$: $m \cdot n \cdot \text{taille}(U)$, où U est un majorant des numérateurs et dénominateurs des entrées.
- ▶ Taille du vecteur $v \in \mathbb{Q}^n$: $n \cdot \text{taille}(U)$, où U est un majorant des numérateurs et dénominateurs des entrées.

136

Notes

Algorithme en temps polynômial pour la programmation linéaire

Théorème (Khachiyan 79)

Il existe un algorithme pour résoudre le programme linéaire

$$\max\{c^T x : x \in \mathbb{R}^n, Ax \leq b\}$$

qui effectue un nombre polynômial d'opérations arithmétiques sur des nombres rationnels de taille polynômiale. Polynômial signifie ici $O(n^k)$ pour une constante k , et n est un majorant des tailles de A , b et c .