

Algorithmique

Programmation Objet

Python



Andrea G. B. Tettamanzi

Université de Nice Sophia Antipolis

Département Informatique

andrea.tettamanzi@unice.fr

CM - Séance 8

Algorithmes de tri

Plan

- Avertissement
- Tri par comptage
- Tri par base
- Tri par insertion (déjà vu)
- Tri fusion
- Tri par sélection
- Tri par tas

Avertissement

Ce qui était vrai dans les années 70 ou 80 ne l'est plus maintenant, car

- On a beaucoup plus de mémoire
- On a intérêt à être localisés en mémoire

Tri par comptage

- Quand on a des nombres de 1 à p on peut les trier facilement en comptant le nombre d'occurrence de chaque nombre.
 - On crée un tableau de p valeurs
 - On met toutes ces valeurs à 0
 - On traverse T . On compte le nombre de fois ou $T[i]$ est pris = on incrémente $P[T[i]]$
 - Ensuite on balaie le tableau P et on copie autant de fois une valeur qu'elle apparaît dans P

Tri par comptage

```
Booleen tri(T[],n)
min ← T[1]; max ← T[1]
pour i de 1 à n
    si T[i] < min: min ← T[i]
    si T[i] > max: max ← T[i]
pour k de 1 à (max – min + 1)
    P[k] ← 0
pour i de 1 à n
    P[T[i] – min + 1] ← P[T[i] – min + 1] + 1
i ← 1
pour k de 1 à (max – min + 1)
    pour j de 1 à P[k]
        T[i] ← k + min - 1
        i ← i + 1
```

Tri par comptage

- La complexité de ce tri est de $O(n + p)$
- Si p est de l'ordre de n alors le tri est linéaire, mais si $p \gg n$?
 - Si $n = 100$ et $p = 10\,000$: c'est quadratique!
 - Si $n = 100$ et $p = 1\,000\,000$: c'est cubique !

Tri par base

- Le tri par base (ou tri radix ou radix sort) est un algorithme de tri rapide qui suppose que les éléments à trier sont des nombres ou des chaînes de caractères
- Cet algorithme était utilisé pour trier des cartes perforées en plusieurs passages.
- On veut trier des cartes à jouer :
 - On fait 4 tas : un pour les ♥, un pour les ♦, un pour les ♣ et un pour les ♠
 - Puis on trie les cartes de chaque couleur par un tri par comptage
- On veut trier des noms de famille
 - Des idées ?

Tri par base

- L'ordre de tri est typiquement le suivant :
 - les clefs courtes viennent avant les clefs longues,
 - les clefs de même taille sont triées selon un ordre lexical.
- Cette méthode correspond à l'ordre naturel des nombres s'ils sont représentés par des chaînes de chiffres.
- Son mode opératoire est :
 - prend le chiffre (ou groupe de bits) le moins significatif de chaque clef,
 - trie la liste des éléments selon ce chiffre, mais conserve l'ordre des éléments ayant le même chiffre (ce qui est un tri stable).
 - répète le tri avec chaque chiffre plus significatif.

Tri par base : exemple

- Trier la liste : 170, 45, 75, 90, 2, 24, 802, 66
- tri par le chiffre le moins significatif (unités) :
- 170, 90, 2, 802, 24, 45, 75, 66
- tri par le chiffre suivant (dizaines) :
- 2, 802, 24, 45, 66, 170, 75, 90
- tri par le chiffre le plus significatif (centaines) :
- 2, 24, 45, 66, 75, 90, 170, 802

Tri par base

- Pour i de 1 à d
 - Utiliser un tri stable pour trier le tableau T sur le digit i
 - Créer un tableau de (base) listes
 - Ajouter chaque clef à la liste correspondante
 - Recomposer le tableau T en parcourant les listes
- digit i dans la base b du nombre n
 - $\text{digit}(i, b, n) = [n / b^{(i-1)}] \text{ modulo } b$
 - $\text{digit}(2, 10, 324) = [324/10] \text{ modulo } 10 = 2$
- Complexité ?

Tri par insertion (rappel)

- Dans l'algorithme, on parcourt le tableau à trier du début à la fin. Au moment où on considère le i -ème élément, les éléments qui le précèdent sont déjà triés.
- L'objectif d'une étape est d'insérer le i -ème élément à sa place parmi ceux qui précèdent.
 - trouver où l'élément doit être inséré en le comparant aux autres,
 - décaler les éléments afin de pouvoir effectuer l'insertion.
- Le tri par insertion est
 - un tri stable (conservant l'ordre d'apparition des éléments égaux)
 - un tri en place (il n'utilise pas de tableau auxiliaire).

Tri par insertion

- Avantages
 - Implémentation simple
 - Efficace pour les petits ensemble de données
 - Efficace pour les ensembles de données qui sont presque déjà triés : complexité $O(n + d)$, où d est le nombre d'inversions
 - Stable
 - En place
 - Au vol (online) : on peut trier une liste comme on la reçoit
- Inconvénients
 - Pire des cas quadratique
 - Fait beaucoup d'échanges (décalages)

Tri fusion

- L'algorithme peut être décrit récursivement :
 - On découpe en deux parties à peu près égales les données à trier
 - On trie les données de chaque partie
 - On fusionne les deux parties
- La récursivité s'arrête car on finit par arriver à des listes composées d'un seul élément et le tri est alors trivial.
- Complexité : $O(n \log(n))$
 - Chaque niveau procède à des fusions dont le coût total est $O(n)$
 - La profondeur maximale est $\log(n)$

Tri fusion

- Très bonne complexité
- Toujours au pire $O(n \log(n))$
- Pénible à écrire pour des tableaux
- Beaucoup plus adapté aux listes

Tri par sélection

- Sur un tableau de n éléments (numérotés de 1 à n), le principe du tri par sélection est le suivant :
 - rechercher le plus petit élément du tableau, et l'échanger avec l'élément d'indice 1 ;
 - rechercher le second plus petit élément du tableau, et l'échanger avec l'élément d'indice 2 ;
 - continuer de cette façon jusqu'à ce que le tableau soit entièrement trié.
- L'invariant de boucle suivant permet de prouver sa correction :
 - à la fin de l'étape i , le tableau est une permutation du tableau initial et les i premiers éléments du tableau coïncident avec les i premiers éléments du tableau trié.

Tri par sélection

- Tri sur place (les éléments sont triés dans la structure)
- Complexité : dans tous les cas, pour trier n éléments, le tri par sélection effectue $n(n - 1)/2$ comparaisons. Sa complexité est donc $O(n^2)$. De ce point de vue, il est inefficace. Il est même moins bon que le tri par insertion
- Par contre, le tri par sélection n'effectue que peu d'échanges :
 - $n - 1$ échanges dans le pire cas, qui est atteint par exemple lorsqu'on trie la séquence $2, 3, \dots, n, 1$;
 - en moyenne
 - aucun si l'entrée est déjà triée.
- Ce tri est donc intéressant lorsque les éléments sont aisément comparables, mais coûteux à déplacer dans la structure.

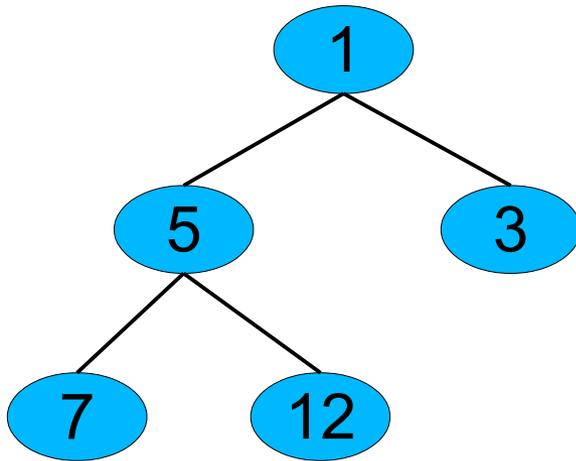
Tri par Tas

- Le tri par tas fonctionne comme le tri par sélection mais utilise une structure de données particulière pour accélérer la recherche : un tas
- Conceptuellement, le tas est un type d'arbre
- Cependant, on peut le réaliser par un tableau

Tas

- Un tas est descendant si
 - chaque noeud est de valeur inférieure à celle de ces deux fils.
- Propriétés
 - Chaque chemin est croissant
 - Le père est le minimum
- Un tas est ascendant si
 - chaque noeud est de valeur supérieure à celle de ces deux fils.
- Propriétés
 - Chaque chemin est décroissant
 - Le père est le maximum

Réalisation d'un tas par un tableau



T[1] = 1
T[2] = 5
T[3] = 3
T[4] = 7
T[5] = 12
T[6] = vide
T[7] = vide

Le père de l'élément T[i] est l'élément T[i/2]

Tas : insertion

```
insérer(T[], n, i, elt)
  T[i] ← elt
  père ← i/2
  fils ← i
  tant que père > 0 et T[père] > T[fils]
    échanger(T[père], T[fils])
    fils ← père
    père ← père/2
```

Tas : extraction

- L'opération de **tamissage** consiste à échanger la racine avec le plus petit de ses fils (si elle est plus grande), et ainsi de suite récursivement jusqu'à ce qu'elle soit à sa place.
- L'extraction consiste à
 - Supprimer la racine
 - Mettre à sa place le dernier élément du tas
 - Tamiser le tas

Tas : tamisage

```
tamiser(T[],n)
  père ← 1
  fils ← 2*père
  fini ← faux
  tant que fils ≤ n et non fini
    si fils < n et T[fils + 1] < T[fils]
      fils ← fils+1
    si T[père] > T[fils]
      échanger(T[père], T[fils])
      père ← fils
      fils ← 2*père
  sinon
    fini ← vrai
```

Tas : extraction

```
entier extraire(T[], n)
    res ← T[1]
    échanger(T[1], T[n])
    tamiser(T, n - 1)
    renvoyer res
```

Tri par tas

- On insère tous les éléments dans le tableau
- On extrait successivement toutes les racines
- Complexité : $O(n \log n)$
- En place
- Un des meilleurs algorithmes de tri
- Le tri par tas a certains défauts (ex : il n'est pas stable) mais la structure de tas est très pratique pour répondre à d'autres problèmes

Merci de votre attention

