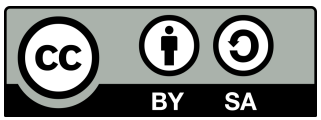
Algorithme du tri par segmentation

QuickSort

Jean-Marc.Vincent@univ-grenoble-alpes.fr[1]

[1] Laboratoire LIG
Équipe-Projet INRIA POLARIS
Université Grenoble-Alpes

L3-INFO : Algorithmique et Modélisation
Grenoble 2024



EXEMPLE : TRI PAR SEGMENTATION

```
def tri_rapide(T):
    plus_grand = []
    liste_pivot = []
    plus_petit = []
    if len(T) <= 1:
        return T
    else:
        # segmentation
        pivot =T[0]
        for x in T:
            if x < pivot:
                plus_petit.append(x)
            elif x > pivot:
                plus_grand.append(x)
            else:
                liste_pivot.append(x)
        # fusion des résultats
        return tri_rapide(plus_petit) + liste_pivot + tri_rapide(plus_grand)
```

UGA Université Grenoble Alpes

EXEMPLE : TRI PAR SEGMENTATION

```
def tri_rapide(T):
    plus_grand = []
    liste_pivot = []
    plus_petit = []
    if len(T) <= 1:
        return T
    else:
        # segmentation
        pivot =T[0]
        for x in T:
            if x < pivot:
                plus_petit.append(x)
            elif x > pivot:
                plus_grand.append(x)
            else:
                liste_pivot.append(x)
        # fusion des résultats
        return tri_rapide(plus_petit) + liste_pivot + tri_rapide(plus_grand)
```

Coût de l'algorithme

- au mieux $\mathcal{O}(n \log n)$ (borne inférieure atteinte) arbre d'appels équilibré
- au pire $\mathcal{O}(n^2)$ arbre d'appels déséquilibré (fil)
- en moyenne ?

UGA Université Grenoble Alpes

SPÉCIFICATION DU PROBLÈME

Le problème est de trier les éléments d'un ensemble de taille n. Les éléments de l'ensemble sont tous comparables deux à deux (unicité des clés) et on suppose qu'il n'y a pas de doublons (ordre total).

Méthode : L'algorithme consiste à choisir un élément pivot, segmenter l'ensemble par rapport à la valeur du pivot puis à trier récursivement les éléments plus petits, puis plus grand et assembler l'ensemble.

TriSegmentation (E)

Données: Un ensemble E de n éléments comparables

Résultat: Les éléments par ordre croissant

if $E \neq \emptyset$

  Pivot = **Extrait** (E)

  `// retire l'élément pivot de` E

  (E_1, E_2) = **Segmentation** (E,pivot)

  `//` E_1 `(resp.` E_2`) éléments plus petits (resp. plus grand) que pivot`

  return Assemble (TriSegmentation (E_1**),pivot, TriSegmentation (**E_2**))**

CORRECTION DE L'ALGORITHME

Preuve à faire en travail personnel

ANALYSE DE COMPLEXITÉ : PRÉLIMINAIRES

Objectif analyser le coût

- taille des données : taille du tableau notée n
- opérateur étudié : comparaison d'éléments

Déroulement de l'analyse

Étape 1 se donner des exemples simples pour comprendre le déroulement de l'algorithme.

Étape 2 rechercher des exemples présentant des "cas extrêmes" au pire et au mieux.

Étape 3 formaliser le problème en écrivant les équations vérifiées par les différents coûts de l'algorithme.

Étape 4 résout ces équations, ou bien, si on ne trouve pas de solution on essaye de majorer/minorer le coût (en ordre de grandeur).

Étape 5 si il y a une grande différence entre le coût au pire et le coût au mieux (par exemple s'il ne sont pas sur la même échelle), on évalue l'ordre de grandeur du coût moyen.

Étape 6 synthétiser les résultats et les commenter

UGA Université Grenoble Alpes

Étapes 1& 2

Étape 1

Faire une trace de l'algorithme sur l'exemple $E = \{D, C, F, G, B, E, A\}$.
L'opération extraire prend les éléments de E dans l'ordre, le pivot étant le premier élément de l'ensemble. Total 11 comparaisons (faire la trace d'exécution). On sent que l'algorithme va être efficace si on arrive à segmenter E en 2 sous-ensembles de même taille. Faire d'autres exemples, tracer l'arbre des appels pour comprendre la structure sous-jacente.

Étape 2

Exemples extrémaux.

- Un premier cas consiste à segmenter au mieux l'ensemble, c'est à dire $|E_1| = |E_2| = \frac{|E|-1}{2}$, par exemple si E={D,C,A,B,E,F,G} on fera 6+2.2=10 comparaisons. Pour un ensemble de taille $n = 2^k - 1$ notre exemple ferait de l'ordre de $k2^k$ comparaisons (voir plus loin), soit de l'ordre de $n \log n$.
- A contrario si la segmentation est très déséquilibrée, par exemple pour $E = \{A, F, B, E, C, G, D\}$ on aura $6 + 5 + 4 + 3 + 2 + 1 = 21$ comparaisons. Pour le même exemple de taille n on aurait $1 + 2 + \cdots + n - 1 = \frac{n(n-1)}{2}$.

On peut remarquer que sur 2 exemples on obtient des valeurs de coût ont des ordres de grandeur différents, ce qui encourage à pousser davantage l'analyse du coût.

UGA Université Grenoble Alpes

ETAPE 3 : FORMALISATION

- $C(E)$: coût de l'algorithme en nombre de comparaisons pour l'entrée E
- n : taille de E.

L'opération de segmentation nécessite $n-1$ opérations donc le coût de l'algorithme vérifie

$$C(E) = n - 1 + C(E_1) + C(E_2).$$

avec $C(E) = 0$ si $n = 0$ ou 1.
Évidemment ce coût dépend fortement du choix du pivot, c'est à dire de la taille respective de E_1 et E_2. Les seules contraintes que l'on connaisse sur E_1 et E_2 sont

$$0 \leqslant |E_1| \leqslant n-1;\ 0 \leqslant |E_2| \leqslant n-1;\ |E_1| + |E_2| = n-1.$$

Notations

$$C_{max}(n) = \max_{|E|=n} C(E) \text{ et } C_{min}(n) = \min_{|E|=n} C(E).$$

UGA Université Grenoble Alpes

Étape 4 : évaluation du coût

Fournir un exemple et démontrer qu'il réalise le pire cas n'est pas facile (l'exemple de l'étape 1 est-il un pire cas ?).
Il est souvent plus simple de majorer $C_{max}(n)$ avec un majorant proche de l'exemple.

Majorant du pire cas : Montrer par récurrence que $C_{max}(n) \leqslant n^2$.

En effet, cette assertion est vraie pour $n = 0$, supposons-la vérifiée pour tout k avec $0 \leqslant k \leqslant n-1$ et soit E une instance réalisant le coût maximum. On a

$$C_{max}(n) = C(E) = n-1+C(E_1)+C(E_2) \leqslant n-1+\max_{0\leqslant k\leqslant n-1}\{C_{max}(k)+C_{max}(n-1-k)\}$$

On applique l'hypothèse de récurrence

$$C_{max}(n) \leqslant n-1+\max_{0\leqslant k\leqslant n-1} k^2+(n-1-k)^2 \leqslant n-1+(n-1)^2 = n(n-1) \leqslant n^2.$$

(la fonction $x^2+(n-1-x)^2$ est une parabole tournée vers le haut donc maximale sur l'une des extrémité de l'intervalle $[0, n-1]$.) La propriété est donc héréditaire, on en déduit que pour tout n

$$C_{max}(n) \leqslant n^2,$$

comme il existe un exemple dont le coût est $\frac{n(n-1)}{2}$ en en conclut que

$$C_{max}(n) = \Theta(n^2)$$

Étape 4 : évaluation du coût (suite)

Même démarche pour le meilleur cas que pour le pire cas

Minorant du meilleur cas

On peut montrer par récurrence que $C_{min}(n) \geqslant \frac{1}{2} n \log n$, on a un exemple de l'ordre de $n \log n$, on en déduit que

$$C_{min}(n) = \Theta(n \log n)$$

Pour un cas général on aura

$$\Theta(n \log n) = C_{min}(n) \leqslant C(E) \leqslant C_{min}(n) = \Theta(n^2)$$

Comment se répartissent les valeurs de $C(n)$ entre $n \log n$ et n^2 ?

UGA Université Grenoble Alpes

INTERMÈDE POUR SE POSER DES QUESTIONS

Complexité moyenne

Hypothèse (discussion)

Donnée :

- tableau de n valeurs distinctes mélangées aléatoirement, uniformément
- tableau $[1, 2, \cdots, n]$ auquel on opère un "shuffle" (mélange aléatoire uniforme)
- tableau de n valeurs réelles tirages aléatoires indépendants de même loi uniforme sur $[0, 1[$

Complexité moyenne

Hypothèse (discussion)

Donnée :

- tableau de n valeurs distinctes mélangées aléatoirement, uniformément
- tableau $[1, 2, \cdots, n]$ auquel on opère un "shuffle" (mélange aléatoire uniforme)
- tableau de n valeurs réelles tirages aléatoires indépendants de même loi uniforme sur $[0, 1[$

Équation de récurrence

$$C(E) = n - 1 + C(E_1) + C(E_2)$$

$$C_{moy}(n) = \mathbb{E}C(E) = (n-1) + \mathbb{E}C(E_1) + \mathbb{E}C(E_2)$$

conditionner par la taille de E_1 (rang du pivot dans E moins 1)

$$C_{moy}(n) = (n-1) + \sum_{i=0}^{n-1} (C_{moy}(i) + C_{moy}(n-1-i))\, \mathbb{P}(\text{pivot de rang } i+1)$$

or $\mathbb{P}(\text{pivot de rang } i+1) = \frac{1}{n}$ (cf hypothèses), en simplifiant la somme

$$C_{moy}(n) = n - 1 + \frac{2}{n} \sum_{i=0}^{n-1} C_{moy}(i). \tag{1}$$

COMPLEXITÉ MOYENNE (2)

Résolution

on réécrit (1) au rang n et au rang $n-1$

$$nC_{moy}(n) = n(n-1) + 2\sum_{i=0}^{n-1} C_{moy}(i)$$

$$(n-1)C_{moy}(n-1) = (n-1)(n-2) + 2\sum_{i=0}^{n-2} C_{moy}(i)$$

$$nC_{moy}(n) - (n-1)C_{moy}(n-1) = n(n-1) - (n-1)(n-2) + C_{moy}(n-1)$$

En simplifiant

$$\frac{C_{moy}(n)}{n+1} = \frac{4}{n+1} - \frac{2}{n} + \frac{C_{moy}(n-1)}{n} = 2\sum_{i=1}^{n} \frac{1}{i} + \frac{4}{n+1} - 4 \simeq 2H_n.$$

D'où

$$C_{moy}(n) = \mathcal{O}(n \log n)$$

ce qui est excellent puisque proche (et au moins sur la même échelle) du meilleur cas pour le tri rapide et atteignant la complexité du problème du tri.

UGA Université Grenoble Alpes

TAKE HOME MESSAGE

Analyse d'algorithme : complexité

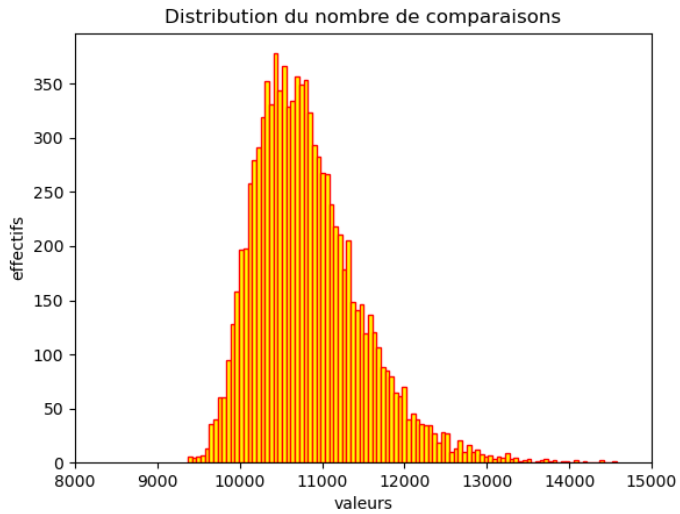
- meilleur cas
- pire cas
- cas moyen
- complexité du problème

TAKE HOME MESSAGE

Analyse d'algorithme : complexité

- meilleur cas $\sim n \log n$
- pire cas $\sim n(n-1)/2$
- cas moyen $\sim 2(n+1)\log n$
- complexité du problème

Expérimentation



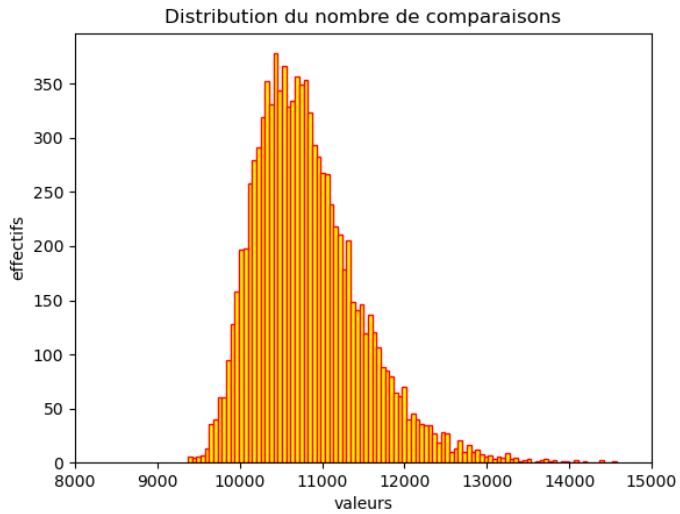
- Taille du tableau $n = 1000$
- Taille de l'échantillon $= 10000$
- meilleur cas
- pire cas
- cas moyen

TAKE HOME MESSAGE

Analyse d'algorithme : complexité

- meilleur cas $\sim n \log n$
- pire cas $\sim n(n-1)/2$
- cas moyen $\sim 2(n+1)\log n$
- complexité du problème

Expérimentation



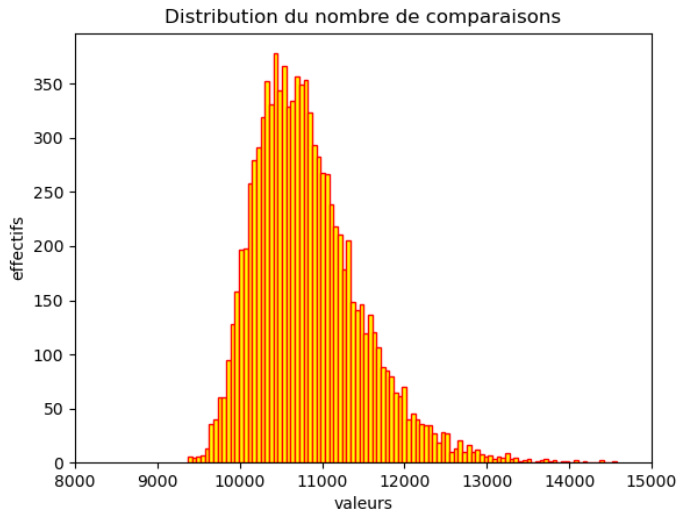
- Taille du tableau $n = 1000$
- Taille de l'échantillon $= 10000$
- meilleur cas ~ 700
- pire cas ~ 50000
- cas moyen ~ 14000

TAKE HOME MESSAGE

Analyse d'algorithme : complexité

- meilleur cas $\sim n \log n$
- pire cas $\sim n(n-1)/2$
- cas moyen $\sim 2(n+1)\log n$
- complexité du problème

Expérimentation



- Taille du tableau $n = 1000$
- Taille de l'échantillon $= 10000$
- meilleur cas $\sim$ 700 observé 9367
- pire cas $\sim$ 50000 observé 14578
- cas moyen $\sim$ 14000 observé 10824

TAKE HOME MESSAGE

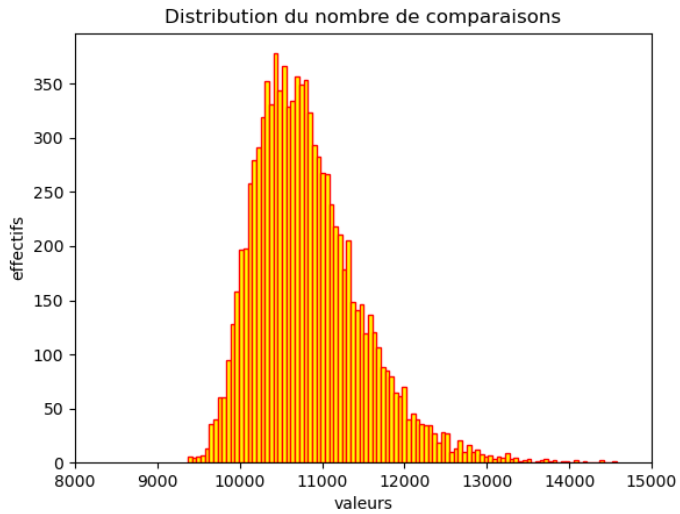
Analyse d'algorithme : complexité

- meilleur cas $\sim n \log n$
- pire cas $\sim n(n-1)/2$
- cas moyen $\sim 2(n+1)\log n$
- complexité du problème

Conclusion

- Tri efficace en moyenne $\mathcal{O}(n \log n)$
- Tri en place (mémoire $\mathcal{O}(n)$)
- Tri non stable
- Justification du choix de pivot randomisé

Expérimentation



- Taille du tableau $n = 1000$
- Taille de l'échantillon $= 10000$
- meilleur cas $\sim$ 700 observé 9367
- pire cas $\sim$ 50000 observé 14578
- cas moyen $\sim$ 14000 observé 10824

EXEMPLE : TRI PAR SEGMENTATION EN PLACE

```
def tri_rapide_en_place(T, l=None, m=None):
    # appel initial
    if l is None: l = 0
    if m is None: m = len(T) - 1
    if m - l > 1:                    # cas généra (cas de base on ne fait rien)
        pivot = T[m]                 # le pivot est la valeur du dernier élément
        i = l                        # place pour l'élément si plus petit que pivot
        j = m                        # place pour l'élément si plus grand que pivot
        k = m - 1                    # place pour l'élément si égal au pivot
        # mise en place du drapeau hollandais (segmentation en place)
        for _ in range(l, m):
            if compare(T[i], pivot):
                if T[i] == pivot:        # zone du milieu
                    T[i], T[k] = T[k], T[i]
                    k = k - 1
                else:                    # zone a gauche du pivot
                    i = i + 1
            else:                        # zone à droite du pivot
                T[i], T[k] = T[k], T[i]
                T[k], T[j] = T[j], T[k]
                j = j - 1
                k = k - 1
        # appels récursifs sur les 2 sous-tableaux
        tri_rapide_en_place(T, l, i - 1)
        tri_rapide_en_place(T, j + 1, m)
```

UGA Université Grenoble Alpes