IUT Robert Schuman
Institut universitaire de technologie
Département **informatique**
Université de Strasbourg

P41 : Complexité Algorithmique

2023/24

1 Tri par Insertion (Insertion Sort)

Le principe est simple, c'est celui d'un classement de jeu de cartes : on reçoit des cartes au fur et à mesure et chaque nouvelle carte est insérée de manière à garder une main bien rangée. Voici l'algorithme pour un tableau A de taille n :

---

triInsertion(A,n) :

  pour $i = 1$ à $n - 1$ **faire**
    $clé = A[i]$
    $j = i - 1$
    tant que $j \geqslant 0$ et $A[j] > clé$ **faire**
      $A[j + 1] = A[j]$
      $j = j - 1$
    fin tant que
    $A[j + 1] = clé$
  fin pour

Cet algorithme, bien que rudimentaire, est efficace sur de petits tableaux ainsi que sur des tableaux « partiellement triés ». C'est aussi un tri *stable*.

2 Tri Fusion (Merge Sort)

C'est un algorithme moins intuitif mais cependant très connu illustrant une technique algorithmique : « diviser et régner » (« divide and conquer »).
Division du problème : on trie les deux moitiés du tableau.
On combine alors les deux sous-tableaux obtenus en les « fusionnant » suivant un algorithme très simple.
Et comment trie-t-on les deux sous-tableaux ? On règne en appliquant récursivement la méthode ci-dessus et on s'arrête lorsqu'on arrive à un sous-tableau de taille 1 qui est trivialement trié !
Ci-après, le $sousTriFusion$ qui triera le sous-tableau $A[p, r[$ (le sous-tableau des éléments d'indice i vérifiant $p \leqslant i < r$) :

---

sousTriFusion(A,p,r) :
  si $p < r - 1$ **alors**
    $q = Ent((p + r)/2)$ (partie entière)
    sousTriFusion(A, p, q)
    sousTriFusion(A, q, r)
    fusion(A, p, q, r)
  fin si

L'algorithme ci-dessus ne fait rien si la longueur est 1. Sinon il scinde puis fusionne les deux sous-tableaux après tri de manière récursive.

Ci-après l'algorithme de fusion :

fusion(A,p,q,r) :

$n_1 = q - p$ (nb de valeurs dans $A[p, q[$)
$n_2 = r - q$ (nb de valeurs dans $A[q, r[$)
Créer des tableaux A_g et A_d de tailles respectives n_1 et n_2
Copier $A[p, q[$ dans A_g et $A[q, r[$ dans A_d
$ind_g = 0$
$ind_d = 0$
$i = p$
tant que $i < r$ **faire**
  si $ind_g == n_1$ **alors**
    $A[i] = A_d[ind_d]$
    $ind_d + +$
  sinon si $ind_d == n_2$ **alors**
    $A[i] = A_g[ind_g]$
    $ind_g + +$
  sinon si $A_g[ind_g] < A_d[ind_d]$ **alors**
    $A[i] = A_g[ind_g]$
    $ind_g + +$
  sinon
    $A[i] = A_d[ind_d]$
    $ind_d + +$
  fin si
  $i + +$
fin tant que

Évidemment, la fonction de tri fusion d'un tableau A de taille n appelle $sousTriFusion$:

triFusion(A,n) :

sousTriFusion$(A, 0, n)$

3 Tri Rapide (Quick Sort)

Un algorithme qui porte bien son nom et qui est très répandu. L'idée est encore une fois simple : on se choisit un élément particulier dans le tableau (le *pivot*) et on réorganise le tableau en plaçant à gauche du pivot tous les éléments plus petits que lui et à sa droite tous les éléments plus grands (on appelle cela la *partition*).
Ensuite, on applique la même méthode aux sous-tableaux de manière récursive.
Dans l'exemple ci-dessous, on donne l'algorithme sous sa forme bête et méchante, à savoir par exemple que le choix du pivot sera toujours le dernier élément du tableau à trié. L'algorithme $sousTriRapide$ triera le sous-tableau $A[p, r[$ (les indices i avec $p \leqslant i < r$). L'algorithme *partition* retourne l'indice du pivot après avoir partitionné le sous-tableau.

---

sousTriRapide(A, p, r) :
  si p<r-1 **alors**
    q = partition(A,p,r)
    sousTriRapide(A,p,q)
    sousTriRapide(A,q+1,r)
  fin si

---

partition(A, p, r) :
  $pivot = A[r-1]$
  $i = p$
  pour $j = p$ à $r-2$ **faire**
    si $A[j] \leqslant pivot$ **alors**
      permuter $A[i]$ et $A[j]$
      $i{+}{+}$
    fin si
  fin pour
  permuter $A[i]$ et $A[r-1]$
  retourner i

Évidemment, la fonction de tri rapide d'un tableau A de taille n appelle `sousTriRapide` :

---

triRapide(A,n) :
  sousTriRapide$(A, 0, n)$

Autant se l'avouer tout de suite, il faudra réfléchir au rôle du pivot ou alors on risque d'avoir de mauvaises surprises.
Cet algorithme n'est pas stable mais est en place.