

Algorithmique TD6: Tris Rapides

20 novembre 2009

```
procedure fusion(ES tableau t(N):entier ,d:entier ,m:entier ,f:entier)
    entier)
variables
    tableau l():entier
    tableau r():entier
    tl, tr, i, j, k:entier
tl ← m-d+1
t2 ← f-m
creertableau(l, tl+1)
creertableau(r, tr+1)
pour i ← 1 a tl pas 1
    l(i) ← t(d+i-1)
fpour
pour j ← 1 a tr pas 1
    r(j) ← t(m+j)
fpour
l(tl+1) ← MAXINT
r(tr+1) ← MAXINT
i ← 1
j ← 1
pour k ← d a f pas 1
    si l(i) ≤ r(j) alors
        t(k) ← l(i)
        i ← i+1
    sinon
        t(k) ← r(j)
        j ← j+1
    fsi
fpour
finprocedure

procedure trifusion(ES: tableau t(N):entier ,d:entier ,f:entier)
variables
m:entier
si d < f alors
    m ← (d+f)/2
    trifusion(t, d, m)
    trifusion(t, m+1, f)
    fusion(t, d, m, f)
fsi
finprocedure
```

```

fonction partition(ES tableau t(N):entier ,d:entier ,f:entier):
    entier
    variables
    x:entier
    x ← t(f)
    i ← d-1
    pour j ← d a f-1 pas 1
        si t(j) ≤ x alors
            i ← i+1
            swap(t(i),t(j))
        fsi
    fpour
    swap(t(i+1),t(f))
    retourner i+1
finfonction

procedure trirapide(ES tableau t(N):entier ,d:entier ,f:entier)
    variables
    m:entier
    si d < f alors
        m ← partition(t,d,f)
        trirapide(t,d,m-1)
        trirapide(t,m+1,f)
    fsi
finprocedure

```

1. Dérouler le tri fusion et le tri rapide sur le tableau suivant :
 $A = (13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 21)$
2. Amélioration du tri fusion
 Comme nous l'avons vu en cours, le tri fusion a une meilleur complexité que le tri insertion. Cependant, les facteurs constants rendent le tri insertion plus rapide pour des tableaux de petite taille.
 - (a) Modifier l'algorithme du tri fusion afin d'y appliquer le tri par insertion pour des sous tableaux de taille k , k étant une valeur à déterminer et n la taille initiale du tableau. Il y a donc n/k tableaux à trier par le tri insertion.
 - (b) Montrer que les n/k sous tableaux peuvent être triées avec le tri par insertion en $\Theta(nk)$ dans le cas le plus défavorable.
 - (c) Montrer que ces sous tableaux peuvent être fusionnées en $\Theta(n * \log_2(n/k))$ dans le cas le plus défavorable.
 - (d) En déduire la complexité de ce nouveau tri et déterminer une bonne valeur pour k en pratique.
3. Amélioration du tri rapide
 - (a) Ecrire une version randomisé du tri rapide, qui choisit le pivot de partitionnement aléatoirement.
 - (b) Améliorer la version randomisée du tri rapide en choisissant le pivot par la méthode du *médian de 3* : on choisit comme pivot le médian

(élément du milieu), d'un ensemble de trois éléments sélectionnés aléatoirement dans le sous-tableau.

- (c) Récursivité terminale : trouver un moyen de ne faire qu'un seul appel récursif dans le tri rapide (technique souvent employée par les bons compilateurs), en utilisant une structure itérative.