Trier – Divide and conquer

G. Aldon - J. Germoni - J.-M. Mény

mars 2012

Fusion de deux paquets de cartes triées

On pose devant soi deux paquets de cartes triées, la plus petite carte au-dessus.

Pour fusionner les deux paquets en un seul paquet trié :

- on prend la plus petite carte que l'on voit sur les deux tas
- puis on prend la plus petite carte que l'on voit sur les deux tas et on la glisse sous la carte que l'on a déjà dans la main
- et on recommence au point précédent...

Fusion de deux listes

Exercice. Écrire une fonction prenant en paramètres deux listes triées $T1$ et $T2$ et renvoyant la fusion des deux listes.

Fusion de deux listes – Programme Xcas

Xcas

```
fusion(T1,T2) :={
si dim(T1)==0 alors retourne T2 ;fsi ;
si dim(T2)==0 alors retourne T1 ;fsi ;
si T1[0]<T2[0] alors retourne prepend(fusion(tail(T1),T2),T1[0]) ;
sinon retourne prepend(fusion(T1,tail(T2)),T2[0]) ;
fsi ;} : ;
```

Fusion de deux listes – Programme Python

Python

```
def fusion(T1,T2) :
    if T1==[] :return T2
    if T2==[] :return T1
    if T1[0]<T2[0] :
        return [T1[0]]+fusion(T1[1 :],T2)
    else :
        return [T2[0]]+fusion(T1,T2[1 :])
```

Tri fusion (merge sort)

Principe du tri fusion d'une liste T :

- Scinder la liste en deux listes T1, T2 que l'on trie par trifusion si elles ne le sont pas,
- fusionner T1 et T2.

Tri fusion (merge sort)

Principe du tri fusion d'une liste T :

- Scinder la liste en deux listes T1, T2 que l'on trie par trifusion si elles ne le sont pas,
- fusionner T1 et T2.

Écrire un programme triant une liste par ce principe.

Tri fusion d'une liste – Programme Xcas

Xcas

```
trifusion(T) :={
local x,T1,T2;
si dim(T)<=1 alors retourne T ;fsi;
T1 :=seq(T[x],x,0,floor(dim(T)/2)-1);
T2 :=seq(T[x],x,floor(dim(T)/2),dim(T)-1);
retourne fusion(trifusion(T1),trifusion(T2));} :;
```

Tri fusion d'une liste – Programme Python

Python

```
def trifusion(T) :
    if len(T)<=1 : return T
    T1=[T[x] for x in range(len(T)//2)]
    T2=[T[x] for x in range(len(T)//2,len(T))]
    return fusion(trifusion(T1),trifusion(T2))
```

Complexité en nombre de comparaisons $C(n)$.

Pour simplifier, supposons que la taille du tableau initial est une puissance de 2 : $n = 2^i$.

- fusion demande à chaque appel au plus $\frac{n}{2}$ comparaisons.

Complexité en nombre de comparaisons $C(n)$.

Pour simplifier, supposons que la taille du tableau initial est une puissance de 2 : $n = 2^i$.

- fusion demande à chaque appel au plus $\frac{n}{2}$ comparaisons.
- $C(n) \leqslant 2 \times C\left(\frac{n}{2}\right) + \frac{n}{2}$

Nombre de comparaisons $C(n)$.

$$C(n) \leqslant 2 \times \left[2C\left(\frac{n}{2^2}\right) + \frac{n}{2^2}\right] + \frac{n}{2}$$

soit

$$C(n) \leqslant 2^2 \times C\left(\frac{n}{2^2}\right) + 2\frac{n}{2}$$

$$C(n) \leqslant 2^3 \times C\left(\frac{n}{2^3}\right) + 3\frac{n}{2}$$

$\vdots$

Nombre de comparaisons $C(n)$.

$$C(n) \leqslant 2 \times \left[2C\left(\frac{n}{2^2}\right) + \frac{n}{2^2}\right] + \frac{n}{2}$$

soit

$$C(n) \leqslant 2^2 \times C\left(\frac{n}{2^2}\right) + 2\frac{n}{2}$$

$$C(n) \leqslant 2^3 \times C\left(\frac{n}{2^3}\right) + 3\frac{n}{2}$$

$\vdots$

$$C(n) \leqslant 2^i \times C\left(\frac{n}{2^i}\right) + i\frac{n}{2}$$

soit

$$C(n) \leqslant k \times n + \frac{1}{2} n \log_2(n) \leqslant Kn\log(n)$$

Complexités – Remarque.

L'implantation proposée détériore en fait la complexité temps et est mauvaise en complexité espace : nombreuses recopies de tableaux lors des appels.