11 Tri par fusion

11.1 Inversion d'une liste triée

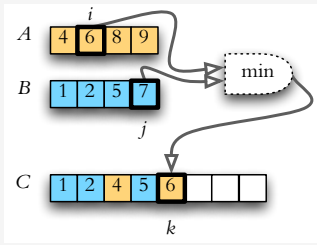
```
    Algo REVERSE(A[0..n − 1])                          // inversion en place
R1  i ← 0; j ← n − 1
R2  tandis que i < j
R3      échanger A[i] ↔ A[j]
R4      i ← i + 1; j ← j − 1
```

11.2 Fusion de deux tableaux triés.

On peut fusionner deux listes (implantées comme tableaux ou listes chaînées), par la récurrence

$$\mathsf{fusion}(A,B) = \begin{cases} B & \text{si } A \text{ est vide} \\ A & \text{si } B \text{ est vide} \\ A[0] \odot \mathsf{fusion}(A[1..],B) & \text{si } A[0] \le B[0] \\ B[0] \odot \mathsf{fusion}(A,B[1..]) & \text{si } B[0] < A[0] \end{cases}$$

où $\odot$ dénote la concaténation.



Dans le cas de deux tableaux, une solution itérative utilise deux indices parcourant les deux listes.

```
    Algo FUSION(A[0..n − 1], B[0..m − 1]) (de type Comparable[], triés)
F1  initialiser C[0..n + m − 1]                    // on place le résultat dans C
F2  i ← 0; j ← 0                                   // i est l'indice dans A ; j est l'indice dans B
F3  pour k ← 0, 1, . . . , n + m − 1 faire
F4      si j ≥ m ou A[i] ≤ B[j] alors C[k] ← A[i]; i ← i + 1
F5      sinon C[k] ← B[j]; j ← j + 1
```

Théorème 11.1. *L'algorithme* FUSION *calcule la fusion de deux tableaux triés en un temps* $\Theta(n+m)$*, avec* $n+m$ *comparaisons [d'éléments entre les listes] au pire.*

11.3 Récurrence du tri par fusion

Tri par fusion (*mergesort*) utilise le principe de **diviser pour régner** dans une procédure récursive. W(fr)

$$\mathsf{tri}\big(A[0..n-1]\big) = \begin{cases} A & \{n<2\} \\ \mathsf{fusion}\Big(\mathsf{tri}\big(A[0..\lfloor n/2\rfloor]\big), \mathsf{tri}\big(A[\lfloor n/2\rfloor + 1..n-1]\big)\Big) & \{n \ge 2\} \end{cases}$$

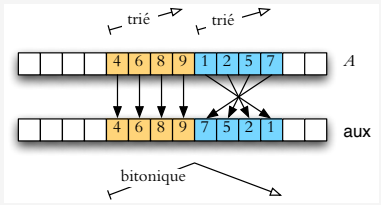
Pour adapter les récurrences à des listes chaînées, il faut séparer une liste en deux moitiés — une implantation simple ajoute des éléments en altérnant entre les deux «demi-listes».

```
tandis que A n'est pas vide faire B_i.insertLast(A.deleteFirst()) ; i ← 3 − i // i = 1, 2, 1, 2, . . .
trier B_1, B_2 ; fusionner
```

11.4 Tri d'un tableau

```
Algo MERGESORT(A[0..n − 1], g, d)
                    // récursion pour trier le sous-tableau A[g..d − 1] : appel initial avec g = 0, d = n
M1  si d − g < 2 alors retourner        // cas de base : tableau vide ou un seul élément
M2  m ← ⌊(d + g)/2⌋                     // m est au milieu
M3  MERGESORT(A, g, m)                  // trier partie gauche
M4  MERGESORT(A, m, d)                  // trier partie droite
M5  FUSION(A, g, m, d)                  // fusion des résultats
```

Typiquement, on utilise un seul tableau auxiliaire pour faire la fusion. Avec un arrangement **bitonique**, on peut simplifier les conditions dans la boucle de la fusion.



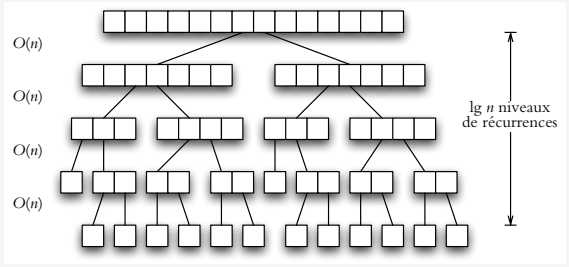

```
Algo FUSION(A[ ], g, m, d) // fusion «en place» pour A[g..m − 1] et A[m..d − 1]
F1  pour i ← g, g + 1, . . . , m − 1 faire aux[i] ← A[i]
F2  pour j ← m, m + 2, . . . , d − 1 faire aux[m + d − 1 − j] ← A[j]
F3  i ← g ; j ← d ; k ← g
F4  tandis que k < d faire                  // meilleure condition : i < m
F5      si aux[i] ≤ aux[j] alors A[k] ← aux[i] ; i ← i + 1; k ← k + 1
F6      sinon A[k] ← aux[j] ; j ← j − 1; k ← k + 1
```

11.5 Temps de calcul du tri par fusion

Dans la fusion, on combine les deux tableaux triés en un troisième en un temps linéaire (Théorème 11.1). Le temps de calcul s'écrit donc comme

$$T(n) = T\big(\lfloor n/2 \rfloor\big) + T\big(\lceil n/2 \rceil\big) + T_{\text{fusion}}\big(\lfloor n/2 \rfloor, \lceil n/2 \rceil\big) + O(1) \tag{11.1}$$

où $T_{\text{fusion}}(k, m) = \Theta(k + m)$ est le temps pour fusionner deux tableaux de tailles k et m.



On a donc $T_{\text{fusion}}(k, m) = \Theta(k + m)$ en (11.1), qui mène à la récurrence classique

$$T(n) = T\big(\lfloor n/2 \rfloor\big) + T\big(\lceil n/2 \rceil\big) + \Theta(n).$$

La solution de la récurrence est $T(n) = \Theta(n \log n)$ (voir Théorème 3A.5). On peut voir cette solution directement en dessinant l'**arbre de récursions** : on a $\lceil \lg n \rceil$ niveaux et $O(n)$ travail (fusions) à chaque niveau.

11.6 Fusion de $m > 2$ listes

En général, on peut fusionner m listes (*multi-way fusion*) de longueur totale n en $O(n \log m)$ temps : voir §8.2 pour une solution avec un tas binaire de taille m.

Solution par récurrence. On peut utiliser le principe de **diviser pour régner** en une procédure récursive.

```
Algo MULTI-FUSION-REC(A_0[ ], A_1[ ], . . . , A_{m−1}[ ]) (tableaux triés)
MR1  si m = 1 alors retourner A_0                    // cas de base : un seul tableau
MR2  si m > 1 alors
MR3      B_1 ← MULTI-FUSION-REC(A_0, . . . , A_{⌊m/2⌋−1})     // fusion d'une moitié
MR4      B_2 ← MULTI-FUSION-REC(A_{⌊m/2⌋}, . . . , A_{m−1})   // fusion de l'autre moitié
MR5      retourner FUSION(B_1, B_2)                  // combiner le résultat des deux moitiés
```