Temps de calcul 3

Théorème similaire pour le nombre de comparaisons par tri B ?

- $A(n)$ ne dépend pas de l'ordre des éléments en `in[]`

- pour tri B, ce nombre dépend de l'ordre

$\Rightarrow$ il faut décider si on veut savoir le pire temps ou le temps moyen ou le meilleur temps ou . . .

Noter : notion du «moyen» dépend de la distribution [probabiliste] des entrées !

Fusion de deux tableaux

```
1  Entrée A1, A2 (de type int[])
2  intialiser Sortie
3  i1 ← 0, i2 ← 0
4  while (i1 < A1.length et i2 < A2.length)
5        if (A1[i1] ≤ A2[i2])
6        then ajouter A1[i1] à la fin de Sortie, i1 ← i1 + 1
7        else ajouter A2[i2] à la fin de Sortie, i2 ← i2 + 1
8  while (i1 < A1.length) ajouter A1[i1] à la fin de Sortie, i1 ← i1 + 1
9  while (i2 < A2.length) ajouter A2[i2] à la fin de Sortie, i2 ← i2 + 1
10 return Sortie
```

Implantation de Sortie par `int[]` :

initialiser par

```
int[] Sortie = new int[A1.length+A2.length]; int sortie_idx=0;
```

ajouter x par

```
Sortie[sortie_idx]=x; sortie_idx++;
```

Temps de calcul 4

Thm. Nombre de comparaisons d'éléments performés par tri B est inférieur à $B(n) = n\lceil \lg n \rceil$. $(n > 0)$

Preuve. Nombre de comparisons pour la fusion de `out1` et `out2` :

$$\leq \texttt{out1.length} + \texttt{out2.length} - 1$$

(Lemme démontré au cours.)

Nombre total de comparaisons sur chaque niveau de récurrences est $\leq n$

Nombre total de niveaux de récurrences : $\lceil \lg(n) \rceil$.

□

Remarque : on utilise $\lg n$ pour dénoter $\log_2(n)$