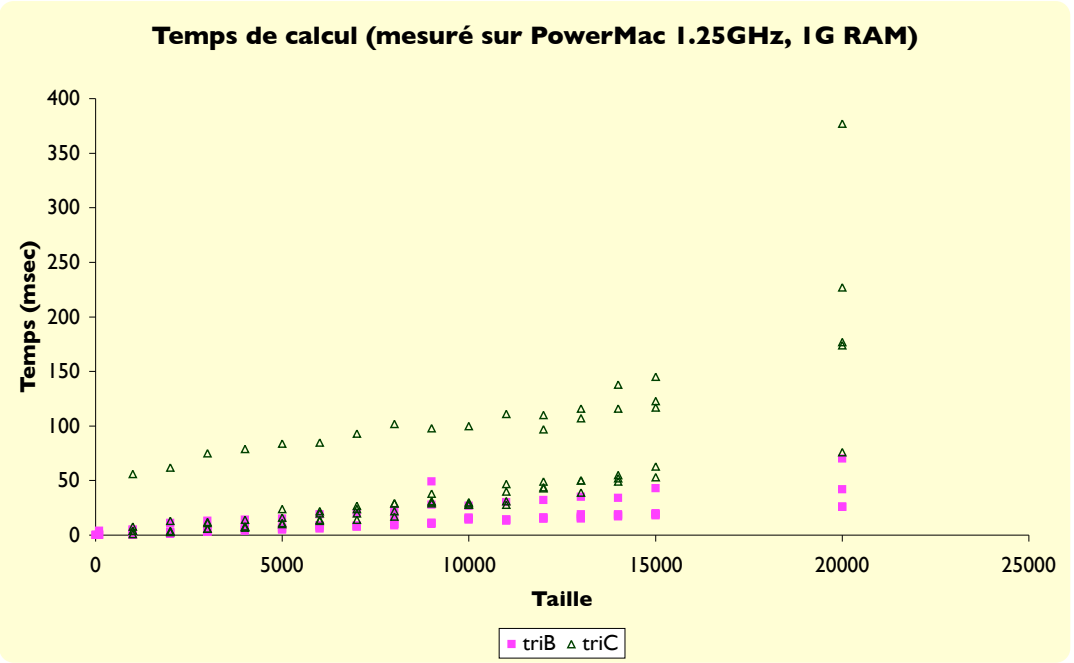
Structure de données

tri C : même algorithme que tri A mais on utilise `java.util.TreeSet` pour implanter IN



Structure de données 2

Pourquoi tri B et tri C ont essentiellement le même comportement ?

Parce que `TreeSet.first()` et `TreeSet.remove()` s'exécutent en temps de $f(n) + c \cdot \lg(n)$ avec $c > 0$ et $f(n) \ll \lg(n)$.

(Arbre binaire balancé : on verra dans ce cours comment le faire)

Notation asymptotique

«Croissance essentielle» d'une fonction :

$3n^2 - 2n + 1$ est de la même croissance essentielle que $12n^2 + 3 \lg n$

Déf. $f(n) \in O\Big(g(n)\Big)$ si et seulement si il existe $N \in \mathbb{N}$ et $c \in \mathbb{R}^+$ t.q. pour tout $n \geq N$,

$$f(n) \leq c \cdot g(n).$$

Noter : on écrit souvent $f(n) = O\Big(g(n)\Big)$ mais en vérité $O(g(n))$ est un *ensemble* de fonctions.

Exemples. . .

$2n + 1$ — linéaire
$3n^2 + \lg n$ — quadratique
$2^n + n$ — exponentiel
$4 \cdot 2^n$ et $7 \cdot 3^n$
$4n \log_7 n + n$
$3 \cdot (\log_5 n)^2$

Notez que $2n + 3 \in O(0.0001n + \log_7 n - \sqrt{n})$ est correct par notre définition.

En général, on choisit la fonction «la plus simple» dans les parenthèses de $O(\cdot)$ [ou o, Ω, Θ]
$\Rightarrow$ dans des devoirs, exercices, etc, seulement la forme la plus simple vaut 100% de crédit.

Notation asymptotique 2

Déf. $f(n) \in \Omega\Big(g(n)\Big)$ si et seulement si $g(n) \in O\Big(f(n)\Big)$.

Déf. $f(n) \in \Theta\Big(g(n)\Big)$ si et seulement si $f(n) \in O\Big(g(n)\Big) \cap \Omega\Big(g(n)\Big)$.

Déf. $f(n) \in o\Big(g(n)\Big)$ si et seulement si pour tout $c \in \mathbb{R}^+$ il existe $N(c) \in \mathbb{N}$ t.q. pour tout $n \geq N(c)$,

$$f(n) \leq c \cdot g(n)$$

Notation asymptotique 3

Thm. $f(n) \in o(g(n))$ ssi

$$f(n) \in O(g(n)) \setminus \Theta(g(n)) = O(g(n)) \setminus \Omega(g(n)).$$

Thm. $f(n) \in o(g(n))$ ssi

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0.$$

Thm. S'il existe $c \in \mathbb{R}^+$ t.q.

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = c,$$

alors $f(n) \in \Theta(g(n))$.

Notation asymptotique 4

Thm Si $f_1(n) \in O(g_1(n))$ et $f_2(n) \in O(g_2(n))$, alors

$$f_1(n) + f_2(n) \in O\Big(g_1(n) + g_2(n)\Big)$$

$$f_1(n) \cdot f_2(n) \in O\Big(g_1(n) \cdot g_2(n)\Big).$$

Thm Si $f(n) \in O(g(n))$ alors $f(n) + g(n) \in O(g(n))$.

Thm $\ln n \in o(n^\epsilon)$ pour tout $\epsilon > 0$

Analyse de temps

```
public static int somme_cubes(int n){
    int somme=0;
    for (int i=1; i<n; i++) somme += i*i*i;
    return i;
}
```

Règle 1. Temps de calcul pour boucle : nombre d'exécutions ⋆ temps d'exécution.

$$n \cdot O(1) = O(n)$$

Analyse de temps 2

```
...
for(int i=0; i<n; i++)
   for (int j=0; j<n; j++)
      k+= i*j;
...
```

Règle 1bis. Boucles imbriqués.

Analyse de temps 3

Règle 2. Étapes consécutives.

Règle 3. if/else

Analyse de temps 4

Thm Temps de calcul pour tri A est $O(n^2)$ et pour tri B est de $O(n \log n)$