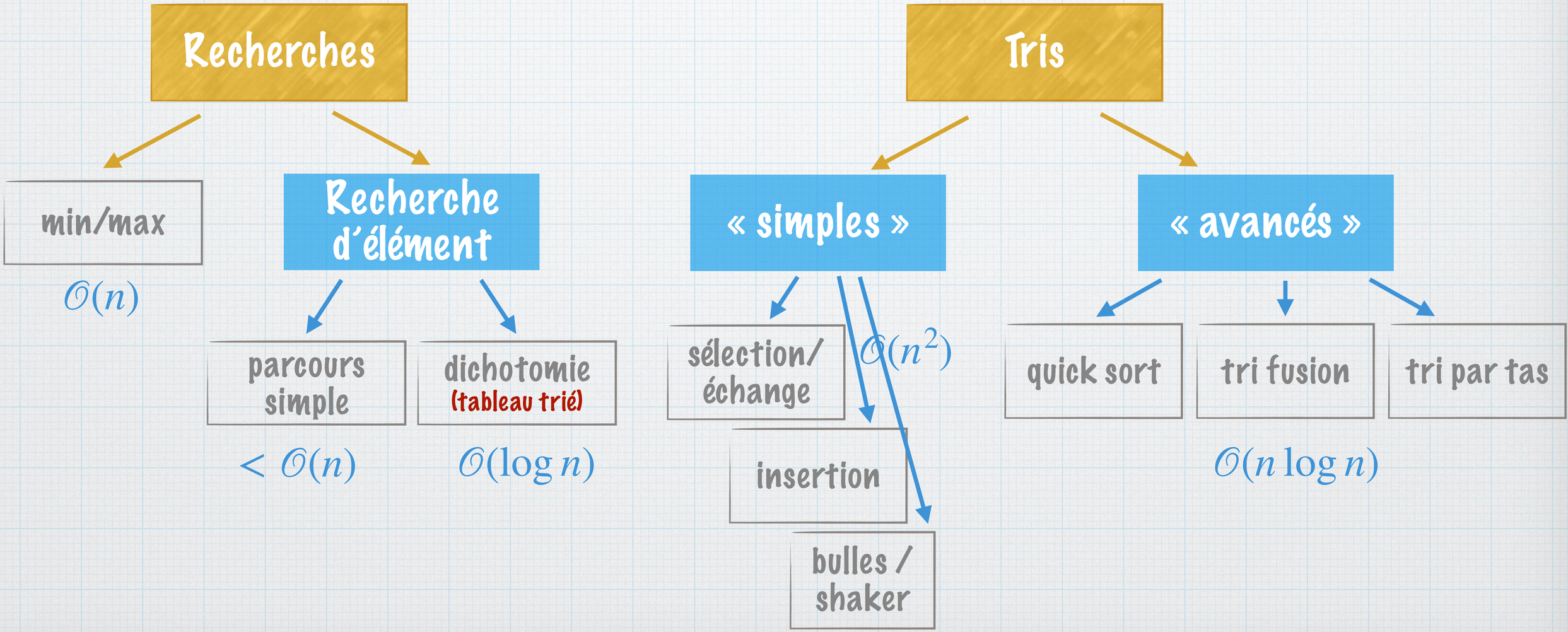
Algorithmique et structures de données

Cours 1 : tableaux et tris

Algorithmique des tableaux



Recherche dichotomique

$\mathcal{O}(\log n)$

Entree : tableau T de n éléments triés
x un élément
Sortie : res contient la position de x dans T s'il est présent et -1 sinon trouvé contient un booléen

```
// i, j : bornes dans le tableau entre lesquelles on recherche x

i ← 0
j ← n-1
trouvé ← FAUX
res ← -1

tant que ((i ⩽ j) ET NON(trouvé))
    m ← partie_entière((i+j)/2)
    si (x = T[m])
        trouvé ← VRAI
        res <- m
    sinon
        si (x < T[m])  // on doit cherche dans la première moitié
            j ← m-1
        sinon  // on doit chercher dans la deuxième moitié
            i ← m+1 fin si
        fin si
    fin si
fin tant que
```

Tris de tableaux

Tri « simple » : tri par sélection/échange

Entree : tableau T de n éléments
Sortie : tableau T trié

```
pour i de 0 à n-2
    // A chaque étape, on trouve le minimum des éléments de i à n-1
    // Puis on le permute pour le mettre à sa place en première position
    min ← i
    pour j de i+1 à n-1
        si (T[j] < T[min])
            min ← j
        fin si
    fin pour
    // Permutation des éléments d'indice i et min de T
    tmp ←T[i]
    T[i] ← T[min]
    T[min] ← tmp
fin pour
```

$\mathcal{O}(n^2)$

Pas de meilleur cas

Tri « simple » : tri par insertion

Entree : tableau T de n éléments
Sortie : tableau T trié

```
pour i de 1 à n-1
    // A chaque étape, on insère le ième élément parmi les éléments déjà triés (donc les éléments de 0 à i-1)
    // L'insertion se fait en parcourant ces éléments de droite à gauche (indice j)

    j ← i-1
    tmp ← T[i]
    tant que ((j>=0) ET (T[j] > tmp))
        // Décaler T[j] d'une case vers la droite T[j+1] <- T[j]
        j ← j-1
    fin tant que
    // j+1 indique la case où l'élément doit être inséré
    T[j+1] ← tmp
fin pour
```

$\mathcal{O}(n^2)$

Meilleur cas : tableau trié

$\mathcal{O}(n)$

Tri « simple » : tri bulle

$\mathcal{O}(n^2)$

En pratique $\ll \mathcal{O}(n^2)$

Meilleur cas : tableau trié

$\mathcal{O}(n)$

Entree : tableau T de n éléments
Sortie : tableau T trié

```
// on parcourt le tableau de droite à gauche : chaque bulle (i-1 / i) est triée
// à chaque étape, au moins un élément de plus est bien trié (le plus léger)

echange ← VRAI
k ← 1 // position du plus petit échange
tant que (echange)  // on teste s'il y a eu des échanges (ie. tri pas encore terminé)
   echange ← FAUX
   ktmp ←n-1
   pour i de n-1 à k (pas -1) // bulle considérée : i-1 / i
      si ( T[i-1] > T[i] ) // bulle à échanger
         tmp ←T[i-1]
         T[i-1] ←T[i]
         T[i] ←tmp
         ktmp ← i
         echange ←VRAI
      fin si
   fin pour
   k ← ktmp+1 // l'élément le plus bas qui a été échangé est bien trié fin tant que
fin tantque
```

Déséquilibre dans la vitesse de convergence des éléments légers/ lourds

Tri « simple » : tri shaker (ie. bulle++)

Entree : tableau T de n éléments
Sortie : tableau T trié

```
echange ← VRAI
k ← 1 // position du plus petit échange (à la descente)
K ← n-1 // position du plus grand échange (à la montée)
tant que (echange)
   echange <- FAUX

   // Passe descendante
   ktmp ← K
   pour i de K à k (pas -1) // bulle considérée : i-1 / i
      si ( T[i-1] > T[i] ) // bulle à échanger échanger
         T[i] ↔T[i-1]
         ktmp ← i
         echange ← VRAI
      fin si
   fin pour
   k ← ktmp+1 // l'élément le plus bas qui a été échangé est bien trié

   // Passe montante
   ktmp ← k
   pour i de k à K // bulle considérée : i-1 / i
      si ( T[i-1] > T[i] ) // bulle à échanger
         échanger T[i] ↔T[i-1]
         ktmp ← i
         echange ← VRAI
      fin si
   fin pour
   K ← ktmp-1 // l'élément le plus haut qui a été échangé est bien trié
fin tant que
```

Tri « avancé » : quicksort

$\mathcal{O}(n \log n)$
Meilleur cas / cas moyen

$\mathcal{O}(n^2)$
Pire cas

Entree : tableau T de n éléments
Sortie : tableau T trié

```
// Partition de T entre les indices i et j
// pivot placé en T[i]
partition(T,i,j) =
    x ← T[i]
    g ← i+1
    d ← j
    tant que (g <= d)
        si ( T[g] > x ) // à placer à la fin du tableau
            échanger T[g] ↔T[d]
            d ← d-1
        sinon
            g ← g+1
    fin tant que
    échanger T[i] ↔T[d] // remettre le pivot
    retourner d
fin fonction
```

Récursive

```
quicksort_aux(T,i,j) =
    si (j-i >= 1)
    // Choix du pivot : placé en début de tableau
        pivot ← ... // premier, dernier élément, milieu, médiane ...
        échanger T[i] ↔T[pivot]
        k ← partition(T,i,j)
        quicksort_aux(T,i,k-1)
        quicksort_aux(T, k+1,j)
    fin si
fin fonction


quicksort(T) = quicksort_aux(T,0,n-1)
```

Tri « avancé » : tri fusion

$\mathcal{O}(n \log n)$

Entree : tableau T de n éléments
Sortie : tableau T trié

```
fusion(T,i,m,j) =
    Tmp ← tableau temporaire de taille j-i+1
    k ← 0
    u ← i
    v ← m
    tant que ((u<=m) && (v<=j))
    // il reste des éléments dans les deux sous-tableaux
        si (T[u] < T[v])
            Tmp[k] ← T[u]
            u ← u+1
            k ← k+1
        sinon
            Tmp[k] ← T[v]
            v ← v+1
            k ← k+1
        fin si
    fin tant que
    // S'il reste des éléments dans la 2ème partie du tableau, ils sont bien placés
    // Si c'est dans la première partie, il faut les copier à la fin
    si (u<=m)
        pour l de m à u
            off ← m-u
            T[j-off] ← T[i]
        fin pour
    fin si
    // Recopie de Tmp à sa place : au début de T
    pour l de 0 à k
        T[i] ← Tmp[k]
    fin pour
fin fonction
```

Récursive

```
tri_fusion(T,i,j) =
    si (i<j-1)
    // S'il y a plus de deux éléments, il faut trier/fusionner
        m ← partie_entière((i+j)/2)
        tri_fusion(T,i,m)
        tri_fusion(T,m+1,j)
        fusion(T,i,m,j)
    fin si
fin fonction
```

Synthèse

		Complexité	Avantages	Inconvénients
Sélection / échange	Moyenne	$\mathcal{O}(n^2)$		- Lent - Pas de meilleur cas
	Meilleur cas			
	Pire des cas			
Insertion	Moyenne	$\mathcal{O}(n^2)$	- Meilleur cas : linéaire sur un tableau trié ou presque	- Lent
	Meilleur cas	$\mathcal{O}(n)$		
	Pire des cas	$\mathcal{O}(n^2)$		
Bulle/shaker	Moyenne	$\mathcal{O}(n^2)$	- Meilleur cas : linéaire sur un tableau trié ou presque - En pratique, bien plus rapide que $\mathcal{O}(n^2)$	- Plus lent que quicksort / fusion
	Meilleur cas	$\mathcal{O}(n)$		
	Pire des cas	$\mathcal{O}(n^2)$		
Quicksort	Moyenne	$\mathcal{O}(n \log n)$	- Rapide - Tri en place (pas de tableau auxiliaire)	- Pire des cas de complexité $\mathcal{O}(n^2)$ - Tableau trié (ou presque) n'est pas un cas favorable
	Meilleur cas	$\mathcal{O}(n \log n)$		
	Pire des cas	$\mathcal{O}(n^2)$		
Tri fusion	Moyenne	$\mathcal{O}(n \log n)$	- Rapide - Complexité constante	- Tableau trié (ou presque) n'est pas un cas favorable - Utilisation de tableaux auxiliaires (complexité en espace $\mathcal{O}(2n)$)
	Meilleur cas	$\mathcal{O}(n \log n)$		
	Pire des cas	$\mathcal{O}(n \log n)$		