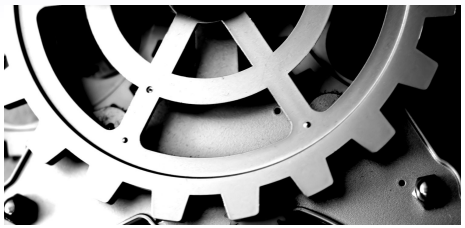
Algorithmique et structures de données
Algorithmes de tri dans un vecteur



Gaël Mahé

slides : Elise Bonzon et Gaël Mahé

Université Paris Descartes

Licence 2

Algorithmes de tri

Algorithmes de tri

Tri par sélection

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![1, n-1]\!]$
  - On cherche le plus petit élément de V à un indice $j \in [\![i, n]\!]$: $V(j) = min(V(1 \to n))$
  - On échange $V(i)$ et $V(j)$

1	2	3	4	5
12	9	3	43	18

Tri par sélection

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![1, n-1]\!]$
  - On cherche le plus petit élément de V à un indice $j \in [\![i, n]\!]$: $V(j) = min(V(1 \to n))$
  - On échange $V(i)$ et $V(j)$

1	2	3	4	5
12	9	3	43	18

Tri par sélection

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![1, n-1]\!]$
  - On cherche le plus petit élément de V à un indice $j \in [\![i, n]\!]$: $V(j) = min(V(1 \to n))$
  - On échange $V(i)$ et $V(j)$

1	2	3	4	5
12	9	3	43	18

Tri par sélection

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![1, n-1]\!]$
  - On cherche le plus petit élément de V à un indice $j \in [\![i, n]\!]$: $V(j) = min(V(1 \to n))$
  - On échange $V(i)$ et $V(j)$

1	2	3	4	5
3	9	12	43	18

Tri par sélection

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![1, n-1]\!]$
  - On cherche le plus petit élément de V à un indice $j \in [\![i, n]\!]$: $V(j) = min(V(1 \to n))$
  - On échange $V(i)$ et $V(j)$

1	2	3	4	5
3	9	12	43	18

Tri par sélection

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![1, n-1]\!]$
  - On cherche le plus petit élément de V à un indice $j \in [\![i, n]\!]$: $V(j) = min(V(1 \to n))$
  - On échange $V(i)$ et $V(j)$

1	2	3	4	5
3	9	12	43	18

Tri par sélection

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![1, n-1]\!]$
  - On cherche le plus petit élément de V à un indice $j \in [\![i, n]\!]$: $V(j) = min(V(1 \to n))$
  - On échange $V(i)$ et $V(j)$

1	2	3	4	5
3	9	12	43	18

Tri par sélection

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![1, n-1]\!]$
  - On cherche le plus petit élément de V à un indice $j \in [\![i, n]\!]$: $V(j) = min(V(1 \to n))$
  - On échange $V(i)$ et $V(j)$

1	2	3	4	5
3	9	12	43	18

Tri par sélection

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![1, n-1]\!]$
  - On cherche le plus petit élément de V à un indice $j \in [\![i, n]\!]$: $V(j) = min(V(1 \to n))$
  - On échange $V(i)$ et $V(j)$

1	2	3	4	5
3	9	12	18	43

Tri par sélection

- Hypothèse de la situation générale
  - les $i-1$ premiers éléments de V sont triés, $1 \leq i \leq n$
  - $\forall j \geq i,\ V(j) \geq V(i-1)$
- Progression vers la solution. Deux cas sont possibles :
  - $i = n$, c'est fini, le vecteur V est trié
  - $i < n$:
    - On cherche l'indice de l'élement minimum de V pour les indices $\{i, \ldots, n\}$
    - On échange $V(i)$ et $V(min)$
    - $i \leftarrow i+1$, on retrouve la situation générale
- Condition initiale : $i = 1$ satisfait les hypothèses de la situation générale

Tri par sélection

L'algorithme de tri par sélection va utiliser la fonction Echange(V, i, j)

- Prend en entrée un vecteur V et deux indices
- Echange les éléments de V correspondant à ces deux indices

Algorithme Echange

Algorithme 1 : Echange

début

```
    /* ENTRÉES: Un vecteur V, deux indices i et j */
    /* SORTIE: Le vecteur V dans lequel les éléments correspondant aux
    indices i et j ont été échangés */
    x ← V(i)
    V(i) ← V(j)
    V(j) ← x
    retourner V
```

fin

Algorithme de tri par sélection

Algorithme 2 : Tri par sélection

début

```
    /* ENTRÉES: Un vecteur V de taille n */
    /* SORTIE: Le vecteur V trié */
    pour i de 1 à n − 1 faire
        min ← i
        pour j de i + 1 à n faire
            si V(j) < V(min) alors min ← j
        Echange(V, i, min)
    retourner V
```

fin

Complexité du tri par sélection

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons et d'échanges ne dépend pas des données du vecteur à trier. Tous les cas sont équivalents en terme de complexité
- Complexité **moyenne** :
  - Etape 1, recherche du min : **(n-1)** comparaisons; **1** échange
  - Etape 2, recherche du min : **(n-2)** comparaisons; **1** échange
  - $\vdots$
  - Etape $n-2$, recherche du min : **2** comparaisons; **1** échange
  - Etape $n-1$, recherche du min : **1** comparaison; **1** échange
- $(n-1)+(n-2)+(n-3)+\ldots+2+1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- $(n-1)$ échanges. Complexité en $\Theta(n)$

Complexité du tri par sélection

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons et d'échanges ne dépend pas des données du vecteur à trier. Tous les cas sont équivalents en terme de complexité
- Complexité **moyenne** :
  - Etape 1, recherche du min : **(n-1)** comparaisons; **1** échange
  - Etape 2, recherche du min : **(n-2)** comparaisons; **1** échange

    ⋮
  - Etape $n-2$, recherche du min : **2** comparaisons; **1** échange
  - Etape $n-1$, recherche du min : **1** comparaison; **1** échange
- $(n-1)+(n-2)+(n-3)+\ldots+2+1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- $(n-1)$ échanges. Complexité en $\Theta(n)$

Complexité du tri par sélection

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons et d'échanges ne dépend pas des données du vecteur à trier. Tous les cas sont équivalents en terme de complexité
- Complexité **moyenne** :
  - Etape 1, recherche du min : **(n-1)** comparaisons; **1** échange
  - Etape 2, recherche du min : **(n-2)** comparaisons; **1** échange

    ⋮
  - Etape $n-2$, recherche du min : **2** comparaisons; **1** échange
  - Etape $n-1$, recherche du min : **1** comparaison; **1** échange
- $(n-1)+(n-2)+(n-3)+\ldots+2+1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- $(n-1)$ échanges. Complexité en $\Theta(n)$

Complexité du tri par sélection

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons et d'échanges ne dépend pas des données du vecteur à trier. Tous les cas sont équivalents en terme de complexité
- Complexité **moyenne** :
  - Etape 1, recherche du min : **(n-1)** comparaisons; **1** échange
  - Etape 2, recherche du min : **(n-2)** comparaisons; **1** échange

    ⋮
  - Etape $n-2$, recherche du min : **2** comparaisons; **1** échange
  - Etape $n-1$, recherche du min : **1** comparaison; **1** échange
- $(n-1)+(n-2)+(n-3)+\ldots+2+1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- $(n-1)$ échanges. Complexité en $\Theta(n)$

Complexité du tri par sélection

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons et d'échanges ne dépend pas des données du vecteur à trier. Tous les cas sont équivalents en terme de complexité
- Complexité **moyenne** :
  - Etape 1, recherche du min : **(n-1)** comparaisons; **1** échange
  - Etape 2, recherche du min : **(n-2)** comparaisons; **1** échange

    ⋮
  - Etape $n-2$, recherche du min : **2** comparaisons; **1** échange
  - Etape $n-1$, recherche du min : **1** comparaison; **1** échange
- $(n-1)+(n-2)+(n-3)+\ldots+2+1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- $(n-1)$ échanges. Complexité en $\Theta(n)$

Complexité du tri par sélection

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons et d'échanges ne dépend pas des données du vecteur à trier. Tous les cas sont équivalents en terme de complexité
- Complexité **moyenne** :
  - Etape 1, recherche du min : **(n-1)** comparaisons; **1** échange
  - Etape 2, recherche du min : **(n-2)** comparaisons; **1** échange

    ⋮
  - Etape $n-2$, recherche du min : **2** comparaisons; **1** échange
  - Etape $n-1$, recherche du min : **1** comparaison; **1** échange
- $(n-1)+(n-2)+(n-3)+\ldots+2+1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- $(n-1)$ échanges. Complexité en $\Theta(n)$

Algorithme de tri par sélection amélioré

Algorithme 3 : Tri par sélection

début

```
    /* ENTRÉES: Un vecteur V de taille n */
    /* SORTIE: Le vecteur V trié */
    pour i de 1 à n − 1 faire
        min ← i
        pour j de i + 1 à n faire
            si V(j) < V(min) alors min ← j
        si min ≠ i alors Echange(V, i, min)
        retourner V
```

fin

→ nombre d'échanges entre 0 et n,
mais le nombre de comparaisons reste d'ordre $O(n^2)$

Algorithmes de tri

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
12	9	3	43	18

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
12	9	3	43	18

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
9	12	3	43	18

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
9	12	3	43	18

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
9	3	12	43	18

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
9	3	12	43	18

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
9	3	12	43	18

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
9	3	12	18	43

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
9	3	12	18	43

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
9	3	12	18	43

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
3	9	12	18	43

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
3	9	12	18	43

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
3	9	12	18	43

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
3	9	12	18	43

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
3	9	12	18	43

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
3	9	12	18	43

Tri à bulles

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur :
  - Chaque fois que l'on rencontre deux éléments consécutifs non ordonnés, on les permute
  - En fin de parcours, le plus grand élément se trouve à droite.
  - On recommence, sans considérer le dernier élément

1	2	3	4	5
3	9	12	18	43

Tri à bulles

- Hypothèse de la situation générale
  - Les éléments $[\![i+1, n]\!]$ de V sont triés
  - Ils sont tous $\geq$ à ceux des rangs $[\![1, i]\!]$
- Progression vers la solution. Deux cas sont possibles :
  - $i = 1$, c'est fini, le vecteur V est trié
  - $2 \leq i \leq n$. Soit $j \in [\![1, i]\!]$:
    - Hypothèse locale : $\forall k \in [\![1, j]\!]$, $V(k) \leq V(j)$
    - Progression locale :
      Si $j = i$, $V(i)$ est à la bonne place, $i \leftarrow i - 1$ et on retrouve l'hypothèse générale
      Si $j < i$: si $V(j) > V(j+1)$, on permute ; $j \leftarrow j + 1$, on retrouve l'hypothèse locale
    - Condition initiale locale : $j = 1$ satisfait l'hypothèse locale
- Condition initiale : $i = n$ satisfait les hypothèses de la situation générale

Algorithme du tri à bulle

Algorithme 4 : Tri à bulle

début

/* ENTRÉES: Un vecteur V de taille n */

/* SORTIE: Le vecteur V trié */

pour i *de* n *à* 2 **faire**

pour j *de* 1 *à* $i-1$ **faire**

si $V(j) > V(j+1)$ **alors** $Echange(V, j, j+1)$

retourner V

fin

Algorithme du tri à bulle

Algorithme 5 : Tri à bulle

début
  /* ENTRÉES: Un vecteur V de taille n */
  /* SORTIE: Le vecteur V trié */
  pour i *de* n *à* 2 **faire**
    pour j *de* 1 *à* $i-1$ **faire**
      si $V(j) > V(j+1)$ **alors** $Echange(V, j, j+1)$
  retourner V
fin

Exemple : $V = (1, 2, 3, 6, 5, 4)$

Algorithme du tri à bulle

Algorithme 6 : Tri à bulle

début
  /* ENTRÉES: Un vecteur V de taille n */
  /* SORTIE: Le vecteur V trié */
  pour *i de n à 2* **faire**
    pour *j de* 1 *à* $i-1$ **faire**
      si $V(j) > V(j+1)$ **alors** $Echange(V, j, j+1)$
  retourner V
fin

Exemple : $V = (1, 2, 3, 6, 5, 4)$
Si on ne permute pas dans un tour de boucle,
alors le vecteur est déjà trié et il est inutile de décrémenter i
$\rightarrow$ on peut optimiser cet algorithme.

Algorithme du tri à bulle optimisé

Algorithme 7 : Tri à bulle optimisé

```
début
    /* ENTRÉES: Un vecteur V de taille n */
    /* SORTIE: Le vecteur V trié */
    i ← n
    répéter
        unEchange ← faux
        pour j de 1 à i − 1 faire
            si V(j) > V(j + 1) alors
                Echange(V, j, j + 1)
                unEchange ← vrai
        i ← i − 1
    jusqu'à NON(unEchange) OU i = 1
    retourner V
fin
```

Complexité du tri à bulles

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons ne dépend pas du vecteur à trier.
  Tous les cas sont équivalents en terme de complexité
- Complexité pour les comparaisons :
  - $i = n$: **(n-1)** comparaisons

    ⋮
  - $i = 2$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons.
  Complexité en $\Theta(n^2)$
- Le nombre d'échanges dépend du vecteur à trier
- Complexité pour les échanges :
  - Meilleur cas : Le tableau est trié, aucun échange
  - Pire cas : Le tableau est trié en ordre inverse.
    Autant d'échanges que de comparaisons, soit $n(n-1)/2$.
    Complexité en $\Theta(n^2)$

Complexité du tri à bulles

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons ne dépend pas du vecteur à trier.
  Tous les cas sont équivalents en terme de complexité
- Complexité pour les comparaisons :
  - $i = n$: **(n-1)** comparaisons
  - $\vdots$
  - $i = 2$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons.
  Complexité en $\Theta(n^2)$
- Le nombre d'échanges dépend du vecteur à trier
- Complexité pour les échanges :
  - Meilleur cas : Le tableau est trié, aucun échange
  - Pire cas : Le tableau est trié en ordre inverse.
    Autant d'échanges que de comparaisons, soit $n(n-1)/2$.
    Complexité en $\Theta(n^2)$

Complexité du tri à bulles

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons ne dépend pas du vecteur à trier.
  Tous les cas sont équivalents en terme de complexité
- Complexité pour les comparaisons :
  - $i = n$: **(n-1)** comparaisons
  - $\vdots$
  - $i = 2$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons.
  Complexité en $\Theta(n^2)$
- Le nombre d'échanges dépend du vecteur à trier
- Complexité pour les échanges :
  - Meilleur cas : Le tableau est trié, aucun échange
  - Pire cas : Le tableau est trié en ordre inverse.
    Autant d'échanges que de comparaisons, soit $n(n-1)/2$.
    Complexité en $\Theta(n^2)$

Complexité du tri à bulles

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons ne dépend pas du vecteur à trier.
  Tous les cas sont équivalents en terme de complexité
- Complexité pour les comparaisons :
  - $i = n$: **(n-1)** comparaisons
  - $\vdots$
  - $i = 2$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons.
  Complexité en $\Theta(n^2)$
- Le nombre d'échanges dépend du vecteur à trier
- Complexité pour les échanges :
  - Meilleur cas : Le tableau est trié, aucun échange
  - Pire cas : Le tableau est trié en ordre inverse.
    Autant d'échanges que de comparaisons, soit $n(n-1)/2$.
    Complexité en $\Theta(n^2)$

Complexité du tri à bulles

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons ne dépend pas du vecteur à trier.
  Tous les cas sont équivalents en terme de complexité
- Complexité pour les comparaisons :
  - $i = n$: **(n-1)** comparaisons
  - $\vdots$
  - $i = 2$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons.
  Complexité en $\Theta(n^2)$
- Le nombre d'échanges dépend du vecteur à trier
- Complexité pour les échanges :
  - Meilleur cas : Le tableau est trié, aucun échange
  - Pire cas : Le tableau est trié en ordre inverse.
    Autant d'échanges que de comparaisons, soit $n(n-1)/2$.
    Complexité en $\Theta(n^2)$

Complexité du tri à bulles

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons ne dépend pas du vecteur à trier.
  Tous les cas sont équivalents en terme de complexité
- Complexité pour les comparaisons :
  - $i = n$: **(n-1)** comparaisons
  - $\vdots$
  - $i = 2$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons.
  Complexité en $\Theta(n^2)$
- Le nombre d'échanges dépend du vecteur à trier
- Complexité pour les échanges :
  - Meilleur cas : Le tableau est trié, aucun échange
  - Pire cas : Le tableau est trié en ordre inverse.
    Autant d'échanges que de comparaisons, soit $n(n-1)/2$.
    Complexité en $\Theta(n^2)$

Complexité du tri à bulles

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons ne dépend pas du vecteur à trier.
  Tous les cas sont équivalents en terme de complexité
- Complexité pour les comparaisons :
  - $i = n$: **(n-1)** comparaisons
  - $\vdots$
  - $i = 2$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons.
  Complexité en $\Theta(n^2)$
- Le nombre d'échanges dépend du vecteur à trier
- Complexité pour les échanges :
  - Meilleur cas : Le tableau est trié, aucun échange
  - Pire cas : Le tableau est trié en ordre inverse.
    Autant d'échanges que de comparaisons, soit $n(n-1)/2$.
    Complexité en $\Theta(n^2)$

Complexité du tri à bulles

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Echange de deux éléments de V
- Le nombre de comparaisons ne dépend pas du vecteur à trier.
  Tous les cas sont équivalents en terme de complexité
- Complexité pour les comparaisons :
  - $i = n$: **(n-1)** comparaisons
  - $\vdots$
  - $i = 2$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons.
  Complexité en $\Theta(n^2)$
- Le nombre d'échanges dépend du vecteur à trier
- Complexité pour les échanges :
  - Meilleur cas : Le tableau est trié, aucun échange
  - Pire cas : Le tableau est trié en ordre inverse.
    Autant d'échanges que de comparaisons, soit $n(n-1)/2$.
    Complexité en $\Theta(n^2)$

Complexité du tri à bulles optimisé

- Pire cas : le vecteur est trié en ordre inverse.
  Même complexité que pour le tri à bulles “classique”
- Meilleur cas : le vecteur est déjà trié.
  On ne parcourt V qu'une fois.
  Donc :
  - $n - 1$ comparaisons
  - 0 échanges

Algorithmes de tri

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	1	1	1	1	1	1	1

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	1	2	1	1	1	1	1

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	1	2	2	1	1	1	1

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	1	2	2	2	1	1	1

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	2	2	2	2	1	1	1

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	2	2	2	1	1	1

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	2	2	2	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	3	2	2	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	4	2	2	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	5	2	2	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	6	2	2	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	2	2	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	2	3	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	3	3	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	4	3	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	5	3	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	5	4	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	5	5	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i) =$ place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	5	6	1	1	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	5	6	1	2	2

Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	5	6	1	2	3

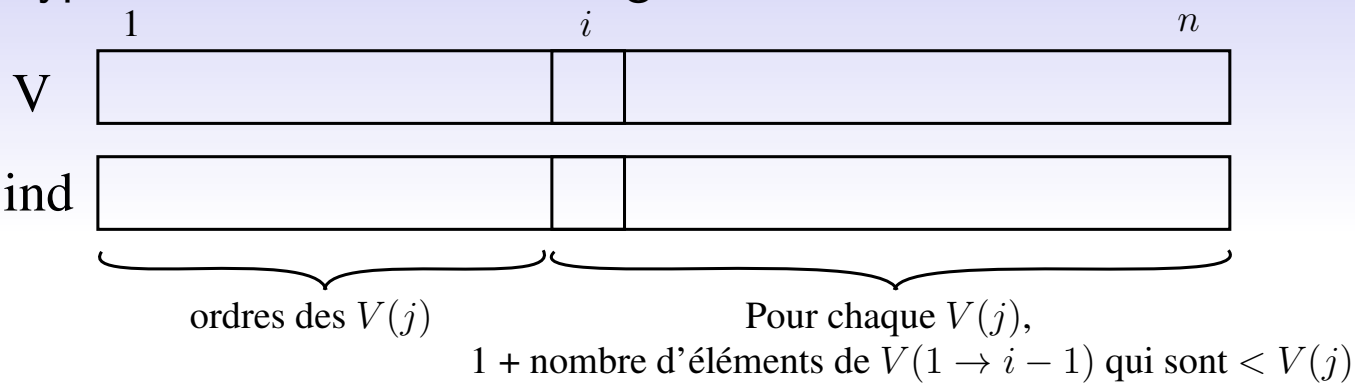
Tri par comptage

- **Principe :** déterminer pour chaque élément de V le nombre d'éléments qui lui sont inférieurs ou égaux
- On construit le vecteur *ind* tel que :
  $ind(i)$ = place de $V(i)$ dans le vecteur trié
- **Méthode :** $\forall\, i \in [\![1, n-1]\!]$, comparer $V(i)$ à tous les $V(j)$, $j \in [\![i+1, n]\!]$.
  - Si $V(j) \leq V(i)$, on incrémente $ind(i)$ de 1
  - Si $V(j) > V(i)$, on incrémente $ind(j)$ de 1

i	1	2	3	4	5	6	7
V	5	15	8	9	2	4	8
ind	3	7	5	6	1	2	4

Pourquoi ça fonctionne ?

- Hypothèse H_i de la situation générale à l'itération i :



- Progression vers la solution. Deux cas sont possibles :
  - $i = n$, c'est fini, *ind* contient les ordres des $V(j)$
  - $i < n$:
    - Pour chaque $V(j)$, avec j de $i+1$ à n,
      - Si $V(j) \leq V(i)$, alors $ind(i)++$
      - Si $V(j) > V(i)$, alors $ind(j)++$
    - A la fin,
      - $ind(i) = 1 +$ nombre d'éléments de $V(1 \to i-1) < V(i)$
        $+$ nombre d'éléments de $V(i+1 \to n) \leq V(i)$
        $=$ ordre de $V(i)$
      - $\forall j \in [\![i+1, n]\!]$, $ind(j) = 1+$ nombre d'éléments de $V(1 \to i) < V(j)$
    - Donc $H_i \Rightarrow H_{i+1}$.
- Condition initiale : $i = 1$ satisfait les hypothèses de la situation générale

Algorithme du tri par comptage

Algorithme 8 : Tri par comptage

début

/* ENTRÉES: Un vecteur V de taille n */
/* SORTIE: Un vecteur W contenant V trié */
pour i *de* 1 *à* n **faire**
$ind(i) \leftarrow 1$
pour i *de* 1 *à* $n-1$ **faire**
pour j *de* $i+1$ *à* n **faire**
si $V(j) \leq V(i)$ **alors**
$ind(i) \leftarrow ind(i)+1$
sinon
$ind(j) \leftarrow ind(j)+1$
pour i *de* 1 *à* n **faire**
$W(ind(i)) \leftarrow V(i)$

fin

Complexité du tri par comptage

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Affectation d'une valeur dans le tableau d'indices
- Le nombre de comparaisons et d'affectations ne dépendent pas du vecteur à trier
- Complexité pour les comparaisons :
  - $i = 1$: **(n-1)** comparaisons
  - ⋮
  - $i = n - 1$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- Complexité pour les affectations :
  - Boucle d'initialisations : n affectations
  - Boucles principales : $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ affectations
  - Boucle de remplissage de W : n affectations

La complexité est donc en $\Theta(n^2)$

Complexité du tri par comptage

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Affectation d'une valeur dans le tableau d'indices
- Le nombre de comparaisons et d'affectations ne dépendent pas du vecteur à trier
- Complexité pour les comparaisons :
  - $i = 1$: **(n-1)** comparaisons
  - $\vdots$
  - $i = n - 1$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- Complexité pour les affectations :
  - Boucle d'initialisations : n affectations
  - Boucles principales : $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ affectations
  - Boucle de remplissage de W : n affectations

La complexité est donc en $\Theta(n^2)$

Complexité du tri par comptage

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Affectation d'une valeur dans le tableau d'indices
- Le nombre de comparaisons et d'affectations ne dépendent pas du vecteur à trier
- Complexité pour les comparaisons :
  - $i = 1$: **(n-1)** comparaisons
  - $\vdots$
  - $i = n - 1$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- Complexité pour les affectations :
  - Boucle d'initialisations : n affectations
  - Boucles principales : $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ affectations
  - Boucle de remplissage de W : n affectations

La complexité est donc en $\Theta(n^2)$

Complexité du tri par comptage

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Affectation d'une valeur dans le tableau d'indices
- Le nombre de comparaisons et d'affectations ne dépendent pas du vecteur à trier
- Complexité pour les comparaisons :
  - $i = 1$: **(n-1)** comparaisons
  - $\vdots$
  - $i = n - 1$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- Complexité pour les affectations :
  - Boucle d'initialisations : n affectations
  - Boucles principales : $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ affectations
  - Boucle de remplissage de W : n affectations

La complexité est donc en $\Theta(n^2)$

Complexité du tri par comptage

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Affectation d'une valeur dans le tableau d'indices
- Le nombre de comparaisons et d'affectations ne dépendent pas du vecteur à trier
- Complexité pour les comparaisons :
  - $i = 1$: **(n-1)** comparaisons
  - $\vdots$
  - $i = n - 1$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- Complexité pour les affectations :
  - Boucle d'initialisations : n affectations
  - Boucles principales : $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ affectations
  - Boucle de remplissage de W : n affectations

La complexité est donc en $\Theta(n^2)$

Complexité du tri par comptage

- Opérations significatives :
  - Comparaison de deux éléments de V
  - Affectation d'une valeur dans le tableau d'indices
- Le nombre de comparaisons et d'affectations ne dépendent pas du vecteur à trier
- Complexité pour les comparaisons :
  - $i = 1$: **(n-1)** comparaisons
  - $\vdots$
  - $i = n - 1$: **1** comparaison
- $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ comparaisons. Complexité en $\Theta(n^2)$
- Complexité pour les affectations :
  - Boucle d'initialisations : n affectations
  - Boucles principales : $(n-1) + (n-2) + (n-3) + \ldots + 2 + 1 = n(n-1)/2$ affectations
  - Boucle de remplissage de W : n affectations

  La complexité est donc en $\Theta(n^2)$

Tri par comptage (bis)

Dans le cas où E est un ensemble fini,
tri plus simple via la construction d'un histogramme des valeurs de V
Ex :

- $E = [\![1, 4]\!]$
- $V = [1, 3, 2, 2, 4, 1, 3, 1]$
- Histogramme $H = [3, 2, 2, 1]$
- V trié $= [1, 1, 1, 2, 2, 3, 3, 4]$

Complexité $= 2n$ affectations.

Tri par comptage (bis)

Algorithme 9 : Tri par comptage bis

début

/* ENTRÉES: Un vecteur V de taille n dont les éléments sont dans un ensemble E fini de cardinal p*/
/* $\forall x \in E$, $\mathrm{rang}(x) =$ rang de x dans E */
/* $\forall j \in [\![1, p]\!]$, $\mathrm{element}(j) = j^{\text{ème}}$ élément de E */
/* SORTIE: Le vecteur V trié */
$H \leftarrow$ vecteur de p zéros
pour *i de 1 à n* **faire**
  $H(\mathrm{rang}(V(i)) \leftarrow H(\mathrm{rang}(V(i)) + 1$
$i \leftarrow 0$
pour *j de 1 à p* **faire**
  pour *k de* 1 *à* $H(j)$ **faire**
    $i \leftarrow i + 1$
    $V(i) \leftarrow \mathrm{element}(j)$

fin

Algorithmes de tri

Tri par insertion

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![2, n]\!]$

  On insère $V(i)$ à son rang parmi les $i - 1$ éléments précédents, déjà triés

= Méthode "du joueur de cartes"

1	2	3	4	5
12	9	3	43	18

Tri par insertion

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![2, n]\!]$

  On insère $V(i)$ à son rang parmi les $i - 1$ éléments précédents, déjà triés

$=$ Méthode “du joueur de cartes”

1	2	3	4	5
12	9	3	43	18

Tri par insertion

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![2, n]\!]$

  On insère $V(i)$ à son rang parmi les $i - 1$ éléments précédents, déjà triés

$=$ Méthode “du joueur de cartes”

1	2	3	4	5
9	12	3	43	18

Tri par insertion

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![2, n]\!]$

  On insère $V(i)$ à son rang parmi les $i - 1$ éléments précédents, déjà triés

$=$ Méthode "du joueur de cartes"

1	2	3	4	5
9	12	3	43	18

Tri par insertion

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![2, n]\!]$

  On insère $V(i)$ à son rang parmi les $i - 1$ éléments précédents, déjà triés

$=$ Méthode "du joueur de cartes"

1	2	3	4	5
3	9	12	43	18

Tri par insertion

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![2, n]\!]$

  On insère $V(i)$ à son rang parmi les $i - 1$ éléments précédents, déjà triés

$=$ Méthode "du joueur de cartes"

1	2	3	4	5
3	9	12	43	18

Tri par insertion

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![2, n]\!]$

  On insère $V(i)$ à son rang parmi les $i - 1$ éléments précédents, déjà triés

$=$ Méthode "du joueur de cartes"

1	2	3	4	5
3	9	12	43	18

Tri par insertion

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![2, n]\!]$

  On insère $V(i)$ à son rang parmi les $i - 1$ éléments précédents, déjà triés

$=$ Méthode “du joueur de cartes”

1	2	3	4	5
3	9	12	43	18

Tri par insertion

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié. On veut trier V.
- Parcourir le vecteur : pour tout $i \in [\![2, n]\!]$

  On insère $V(i)$ à son rang parmi les $i - 1$ éléments précédents, déjà triés

$=$ Méthode "du joueur de cartes"

1	2	3	4	5
3	9	12	18	43

Tri par insertion

- Hypothèse de la situation générale
  - les $i-1$ premiers éléments de V sont triés, $1 \leq i \leq n$
- Progression vers la solution. Deux cas sont possibles :
  - $i = n+1$, c'est fini, le vecteur V est trié
  - $i \leq n$:
    - On cherche le rang r de $V(i)$ dans le sous-vecteur $V(1 \ldots i-1)$
    - On insère $V(i)$ à la position r tout en décalant $V(r \ldots i-1)$ vers la droite
    - $i \leftarrow i+1$, on retrouve la situation générale
- Condition initiale : $i = 2$ satisfait les hypothèses de la situation générale

Algorithme de tri par insertion

Algorithme 10 : Tri par insertion

début
    /* ENTRÉES: Un vecteur V de taille n */
    /* SORTIE: Le vecteur V trié */
    pour *i de 2 à n* **faire**
        $r \leftarrow i$
        $x \leftarrow V(i)$
        tant que $V(r-1) > x$ *et* $r > 1$ **faire**
            $V(r) \leftarrow V(r-1)$
            $r \leftarrow r-1$
        $V(r) \leftarrow x$
    retourner V
fin

Complexité du tri par insertion

- Opérations significatives :
  - Comparaison $V(r-1) > x$
  - Affectations depuis ou vers V
- Dans tous les cas, $n-1$ itérations sur i. Pour chacune,
  - Dans le meilleur des cas (vecteur trié),
    1 comparaison et 2 affectations
  - Dans le pire des cas (vecteur trié en ordre inverse),
    i comparaisons et $i+1$ affectations
  - En moyenne,
    autant de comparaisons que dans la recherche séquentielle dans un vecteur trié, soit $\frac{i-1}{2}$
    autant d'affectations que de comparaisons, plus 1, soit $\frac{i+1}{2}$

Complexité du tri par insertion

- Opérations significatives :
  - Comparaison $V(r-1) > x$
  - Affectations depuis ou vers V
- Dans tous les cas, $n-1$ itérations sur i. Pour chacune,
  - Dans le meilleur des cas (vecteur trié),
    1 comparaison et 2 affectations
  - Dans le pire des cas (vecteur trié en ordre inverse),
    i comparaisons et $i+1$ affectations
  - En moyenne,
    autant de comparaisons que dans la recherche séquentielle dans un vecteur trié, soit $\frac{i-1}{2}$
    autant d'affectations que de comparaisons, plus 1, soit $\frac{i+1}{2}$

Complexité du tri par insertion

- Opérations significatives :
  - Comparaison $V(r-1) > x$
  - Affectations depuis ou vers V
- Dans tous les cas, $n-1$ itérations sur i. Pour chacune,
  - Dans le meilleur des cas (vecteur trié),
    1 comparaison et 2 affectations
  - Dans le pire des cas (vecteur trié en ordre inverse),
    i comparaisons et $i+1$ affectations
  - En moyenne,
    autant de comparaisons que dans la recherche séquentielle dans un vecteur trié, soit $\frac{i-1}{2}$
    autant d'affectations que de comparaisons, plus 1, soit $\frac{i+1}{2}$

Complexité du tri par insertion

- Opérations significatives :
  - Comparaison $V(r-1) > x$
  - Affectations depuis ou vers V
- Dans tous les cas, $n-1$ itérations sur i. Pour chacune,
  - Dans le meilleur des cas (vecteur trié),
    1 comparaison et 2 affectations
  - Dans le pire des cas (vecteur trié en ordre inverse),
    i comparaisons et $i+1$ affectations
  - En moyenne,
    autant de comparaisons que dans la recherche séquentielle dans un vecteur trié, soit $\frac{i-1}{2}$
    autant d'affectations que de comparaisons, plus 1, soit $\frac{i+1}{2}$

Complexité du tri par insertion

- Complexité en nombre de comparaisons :
  - Meilleur des cas : $n-1$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i = \frac{n(n-1)}{2} - 1 = \frac{n^2-n-2}{2} \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i-1}{2} = \frac{1}{2}\sum_{i=1}^{n-2} i = \frac{(n-2)(n-1)}{4} \sim \frac{n^2}{4}$$

- Complexité en nombre d'affectations :
  - Meilleur des cas : $2(n-1)$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i+1 = \frac{n(n-1)}{2} - 1 + (n-2) = \frac{n^2}{2} + \frac{n}{2} - 3 \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i+1}{2} = \frac{n^2}{4} + \frac{n}{4} - \frac{3}{2} \sim \frac{n^2}{4}$$

Complexité du tri par insertion

- Complexité en nombre de comparaisons :
  - Meilleur des cas : $n-1$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i = \frac{n(n-1)}{2} - 1 = \frac{n^2-n-2}{2} \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i-1}{2} = \frac{1}{2}\sum_{i=1}^{n-2} i = \frac{(n-2)(n-1)}{4} \sim \frac{n^2}{4}$$

- Complexité en nombre d'affectations :
  - Meilleur des cas : $2(n-1)$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i + 1 = \frac{n(n-1)}{2} - 1 + (n-2) = \frac{n^2}{2} + \frac{n}{2} - 3 \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i+1}{2} = \frac{n^2}{4} + \frac{n}{4} - \frac{3}{2} \sim \frac{n^2}{4}$$

Complexité du tri par insertion

- Complexité en nombre de comparaisons :
  - Meilleur des cas : $n-1$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i = \frac{n(n-1)}{2} - 1 = \frac{n^2-n-2}{2} \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i-1}{2} = \frac{1}{2}\sum_{i=1}^{n-2} i = \frac{(n-2)(n-1)}{4} \sim \frac{n^2}{4}$$

- Complexité en nombre d'affectations :
  - Meilleur des cas : $2(n-1)$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i+1 = \frac{n(n-1)}{2} - 1 + (n-2) = \frac{n^2}{2} + \frac{n}{2} - 3 \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i+1}{2} = \frac{n^2}{4} + \frac{n}{4} - \frac{3}{2} \sim \frac{n^2}{4}$$

Complexité du tri par insertion

- Complexité en nombre de comparaisons :
  - Meilleur des cas : $n-1$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i = \frac{n(n-1)}{2} - 1 = \frac{n^2-n-2}{2} \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i-1}{2} = \frac{1}{2}\sum_{i=1}^{n-2} i = \frac{(n-2)(n-1)}{4} \sim \frac{n^2}{4}$$

- Complexité en nombre d'affectations :
  - Meilleur des cas : $2(n-1)$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i + 1 = \frac{n(n-1)}{2} - 1 + (n-2) = \frac{n^2}{2} + \frac{n}{2} - 3 \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i+1}{2} = \frac{n^2}{4} + \frac{n}{4} - \frac{3}{2} \sim \frac{n^2}{4}$$

Complexité du tri par insertion

- Complexité en nombre de comparaisons :
  - Meilleur des cas : $n-1$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i = \frac{n(n-1)}{2} - 1 = \frac{n^2-n-2}{2} \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i-1}{2} = \frac{1}{2}\sum_{i=1}^{n-2} i = \frac{(n-2)(n-1)}{4} \sim \frac{n^2}{4}$$

- Complexité en nombre d'affectations :
  - Meilleur des cas : $2(n-1)$
  - Pire des cas :

$$\sum_{i=2}^{n-1} i+1 = \frac{n(n-1)}{2} - 1 + (n-2) = \frac{n^2}{2} + \frac{n}{2} - 3 \sim \frac{n^2}{2}$$

  - En moyenne :

$$\sum_{i=2}^{n-1} \frac{i+1}{2} = \frac{n^2}{4} + \frac{n}{4} - \frac{3}{2} \sim \frac{n^2}{4}$$

Complexité du tri par insertion

- Complexité en nombre de comparaisons :
  - Meilleur des cas : $n-1$
  - Pire des cas :
$$\sum_{i=2}^{n-1} i = \frac{n(n-1)}{2} - 1 = \frac{n^2-n-2}{2} \sim \frac{n^2}{2}$$
  - En moyenne :
$$\sum_{i=2}^{n-1} \frac{i-1}{2} = \frac{1}{2}\sum_{i=1}^{n-2} i = \frac{(n-2)(n-1)}{4} \sim \frac{n^2}{4}$$
- Complexité en nombre d'affectations :
  - Meilleur des cas : $2(n-1)$
  - Pire des cas :
$$\sum_{i=2}^{n-1} i + 1 = \frac{n(n-1)}{2} - 1 + (n-2) = \frac{n^2}{2} + \frac{n}{2} - 3 \sim \frac{n^2}{2}$$
  - En moyenne :
$$\sum_{i=2}^{n-1} \frac{i+1}{2} = \frac{n^2}{4} + \frac{n}{4} - \frac{3}{2} \sim \frac{n^2}{4}$$

Complexité du tri par insertion

- Complexité en nombre de comparaisons :
  - Meilleur des cas : $n-1$
  - Pire des cas :
$$\sum_{i=2}^{n-1} i = \frac{n(n-1)}{2} - 1 = \frac{n^2-n-2}{2} \sim \frac{n^2}{2}$$
  - En moyenne :
$$\sum_{i=2}^{n-1} \frac{i-1}{2} = \frac{1}{2}\sum_{i=1}^{n-2} i = \frac{(n-2)(n-1)}{4} \sim \frac{n^2}{4}$$
- Complexité en nombre d'affectations :
  - Meilleur des cas : $2(n-1)$
  - Pire des cas :
$$\sum_{i=2}^{n-1} i + 1 = \frac{n(n-1)}{2} - 1 + (n-2) = \frac{n^2}{2} + \frac{n}{2} - 3 \sim \frac{n^2}{2}$$
  - En moyenne :
$$\sum_{i=2}^{n-1} \frac{i+1}{2} = \frac{n^2}{4} + \frac{n}{4} - \frac{3}{2} \sim \frac{n^2}{4}$$

Complexité du tri par insertion

On peut réduire le nombre d'affectations en utilisant des listes doublement chaînées :

- Si V implémenté sous forme de tableau : $\sim n^2/4$ affectations
- Si V implémenté sous forme de liste chaînée : $\sim 6n$ affectations

Tri par insertion dichotomique

- Recherche du rang de $V(i)$ $\sim$ recherche séquentielle dans un vecteur trié

$\rightarrow$ Idée : réduire complexité par une recherche dichotomique

  - Recherche séquentielle : de l'ordre de i comparaisons
  - Recherche dichotomique : de l'ordre de $\log_2(i)$ comparaisons

Algorithme de tri par insertion dichotomique

Algorithme 11 : Tri par insertion dichotomique

début

/* ENTRÉES: Un vecteur V de taille n */

/* SORTIE: Le vecteur V trié */

pour i *de* 2 *à* n **faire**

$r \leftarrow \text{trouve\_rang}(V(i), V(1 \rightarrow i-1))$

$x \leftarrow V(i)$

pour $j = i - 1$ *à* r **faire**

$V(j+1) \leftarrow V(j)$

$V(r) \leftarrow x$

retourner V

fin

Complexité de l'insertion dichotomique

- Nombre d'**affectations** inchangé :
  - Meilleur cas : $\sim 2n$
  - Pire cas : $\sim n^2/2$
  - Moyenne : $\sim n^2/4$

  NB : à cause de la recherche dichotomique,
  impossible d'utiliser une liste chaînée pour réduire nombre d'affectations
- Nombre de **comparaisons** :
  - Pour chaque i de 2 à n : $\sim \log_2(i)$ comparaisons (min/max/moy)
  - Total : $\sum_{i=2}^{n} \log_2(i) \sim n \log_2(n)$ comparaisons

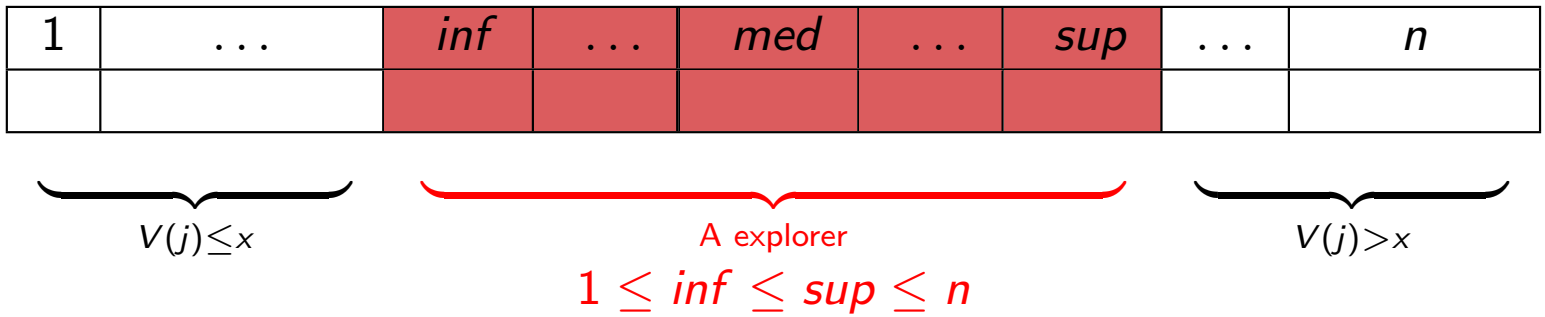
Recherche dichotomique du rang d'insertion max : construction

Soit V trié de taille n. On cherche r tel que :

$$V(r-1) \leq x < V(r)$$

Si $x \geq V(n)$, $r = n+1$. Sinon :

- Situation générale



- 2 cas possibles :
  - $inf = sup$: retourne inf.
  - $inf < sup$: on pose $med = \lfloor \frac{(inf+sup)}{2} \rfloor$
    - $x < V(med)$, $sup \leftarrow med$
    - $x \geq V(med)$, $inf \leftarrow med + 1$
- Conditions initiales : $inf = 1$, $sup = n$.
  Respectent les hypothèses de la situation générale.

Recherche dichotomique du rang d'insertion max

Algorithme 12 : Recherche dichotomique du rang d'insertion maximal d'un élément x dans un vecteur V trié de taille n.

```
début
    /* ENTRÉES: Un vecteur V trié de taille n, un élément x */
    /* SORTIE: rang d'insertion maximal de x dans V */
    si x ≥ V(n) alors
        retourner n + 1
    sinon
        inf ← 1, sup ← n
        tant que inf < sup faire
            med ← (inf + sup) div 2
            si x < V(med) alors sup ← med
            sinon inf ← med + 1
        retourner inf
fin
```

Recherche dichotomique du rang d'insertion max

Pour $x < V(n)$, montrons que l'algorithme renvoie bien r tel que

$$V(r-1) \leq x < V(r)$$

- Hypothèse de récurrence pour l'itération k : H_k : $inf_k \leq r \leq sup_k$
- Si $x < V(med_k)$,
- Si $x \geq V(med_k)$,
- Donc $H_k \Rightarrow H_{k+1}$
- Comme H_1 vraie (avec $inf_1 = 1$ et $sup_1 = n$),
  H_k vraie $\forall\, 1 \leq k \leq p$= nombre d'itérations
- Or $inf_p = sup_p$. Donc $r = inf_p$. Et l'algo retourne r.

Ordres de complexité des algos de tri

Tri	Comparaisons		Transferts	
	moy	max	moy	max
par sélection	n^2	n^2	n	n
à bulles	n^2	n^2	n^2	n^2
par comptage	n^2	n^2	n^2	n^2
par insertion séquentielle	n^2	n^2	n^2	n^2
par insertion dichotomique	$n\log(n)$	$n\log(n)$	n^2	n^2
par ins. seq dans liste chaînée	n^2	n^2	$6n$	$6n$

Algorithmes de tri

Algorithme du drapeau à 3 couleurs

- Soit $V : [\![1, n]\!] \to E$ un vecteur non trié
- Ses éléments sont de 3 types : *R*ouge, *B*leu et *J*aune
- On veut trier V de façon à ce que les premiers éléments soient bleus, les suivants jaunes, puis enfin les derniers rouges.

Algorithme du drapeau à 3 couleurs

- Hypothèse générale :

1	...	j	...	i	...	r	...	n
B B B B		J J J J					R R R R	

Non testé

- Progression vers la solution, 2 cas possibles :
  - $i = r + 1$. C'est fini, tout le vecteur est traité
  - $i \leq r$, 3 cas possibles
    - $V(i) = J$,

1	...	j	...	i	...	r	...	n
B B B B		J J J J		J			R R R R	

→ On vérifie l'hypothèse générale pour $i \leftarrow i + 1$

    - $V(i) = B$: $Echange(V, j, i)$

1	...	j	...	i	...	r	...	n
B B B B		B	J J J	J			R R R R	

→ On vérifie l'hypothèse générale pour $i \leftarrow i + 1$ et $j \leftarrow j + 1$

    - $V(i) = R$: $Echange(V, r, i)$

1	...	j	...	i	...	r	...	n
B B B B		J J J J				R	R R R R	

→ On vérifie l'hypothèse générale pour $r \leftarrow r - 1$

  Dans les 3 cas, soit i croît, soit r décroît (donc progression vers $i = r + 1$)

- Les conditions initiales $i = 1, j = 1, r = n$ vérifient l'hypothèse générale

Algorithme du drapeau à 3 couleurs

Algorithme 13 : Algorithme du drapeau à 3 couleurs

début

/* ENTRÉES: Un vecteur V de taille n */

/* SORTIE: Le vecteur V trié */

$i \leftarrow 1$; $j \leftarrow 1$; $r \leftarrow n$

tant que $i \leq r$ **faire**

si $V(i) = J$ **alors** $i \leftarrow i + 1$

sinon

si $V(i) = B$ **alors** $Echange(V, j, i)$; $j \leftarrow j + 1$; $i \leftarrow i + 1$

sinon $Echange(V, r, i)$; $r \leftarrow r - 1$

retourner V

fin

Algorithme du drapeau à 3 couleurs

Algorithme 14 : Algorithme du drapeau à 3 couleurs

début

/* ENTRÉES: Un vecteur V de taille n */

/* SORTIE: Le vecteur V trié */

$i \leftarrow 1$; $j \leftarrow 1$; $r \leftarrow n$

tant que $i \leq r$ **faire**

si $V(i) = J$ **alors** $i \leftarrow i + 1$

sinon

si $V(i) = B$ **alors** $Echange(V, j, i)$; $j \leftarrow j + 1$; $i \leftarrow i + 1$

sinon $Echange(V, r, i)$; $r \leftarrow r - 1$

retourner V

fin

Exemple : $V = [J, R, R, B, J, R, R]$

Algorithme du drapeau à 3 couleurs

Algorithme 15 : Algorithme du drapeau à 3 couleurs

début

/* ENTRÉES: Un vecteur V de taille n */
/* SORTIE: Le vecteur V trié */
$i \leftarrow 1$; $j \leftarrow 1$; $r \leftarrow n$
tant que $i \leq r$ **faire**
si $V(i) = J$ **alors** $i \leftarrow i + 1$
sinon
si $V(i) = B$ **alors** $Echange(V, j, i)$; $j \leftarrow j + 1$; $i \leftarrow i + 1$
sinon $Echange(V, r, i)$; $r \leftarrow r - 1$
retourner V

fin

Exemple : $V = [J, R, R, B, J, R, R]$
$\Rightarrow$ Nombre de permutations non optimal.
On peut optimiser cet algorithme.

Algorithme du drapeau à 3 couleurs optimisé

Algorithme 16 : Algorithme du drapeau à 3 couleurs optimisé

début
    /* ENTRÉES: Un vecteur V de taille n*/
    /* SORTIE: Le vecteur V trié */
    $i \leftarrow 1$; $j \leftarrow 1$; $r \leftarrow n$
    tant que $i \leq r$ **faire**
        si $V(i) = J$ **alors** $i \leftarrow i + 1$
        sinon
            si $V(i) = B$ **alors** $Echange(V, j, i)$; $j \leftarrow j + 1$; $i \leftarrow i + 1$
            sinon
                tant que $V(r) = R$ *et* $r > i$ **faire** $r \leftarrow r - 1$
                $Echange(V, r, i)$
                $r \leftarrow r - 1$
    retourner V
fin