IREM Clermont-Ferrand
Algorithmique au lycée

Malika More
malika.more@u-clermont1.fr
28 janvier 2011

Proposition d'activité utilisant l'application

Tripatouille

(http://www.malgouyres.fr/tripatouille/)

Il s'agit d'une application permettant :
- De simuler à la main des algorithmes de tri
- De voir des démonstrations pas à pas des trois algorithmes de tri quadratiques classiques (tri par sélection, tri par insertion, tri par bulles).

On peut l'utiliser dans un navigateur internet à l'adresse indiquée ou la télécharger à la même adresse.

1 Description de l'application

- Partie du haut
  - ⋆ Zone `Taille du tableau`
    - ◦ Permet de choisir le nombre d'éléments du tableau
    - ◦ Minimum : 2 - Par défaut : 15 - Maximum : 50
    - ◦ Attention : il est nécessaire de taper sur `Entrée` pour valider la taille choisie
  - ⋆ Bouton `Nouveau tableau`
    - ◦ Crée un tableau de nombres aléatoires compris entre 0 et $2n-1$ (pour un tableau de n éléments)
  - ⋆ Bouton `Ouvrir un fichier`
    - ◦ Attention : ce bouton est désactivé lorsque l'application s'exécute dans le navigateur internet
    - ◦ Permet de charger en guise de tableau un fichier de nombres compris entre 0 et 99 et séparés par des espaces ou des sauts de lignes
    - ◦ Ce fichier doit être muni de l'extension `.tab`
  - ⋆ Bouton `Enregistrer le tableau`
    - ◦ Attention : ce bouton est désactivé lorsque l'application s'exécute dans le navigateur internet
    - ◦ Permet d'enregistrer le tableau courant sous la forme d'un fichier muni de l'extension `.tab`
  - ⋆ Bouton `À propos de Tripatouille`
    - ◦ Contient les informations de copyright et de version
  - ⋆ Tableau central
    - ◦ C'est le tableau initial qu'il s'agit de trier
    - ◦ Le contenu des cases apparaît dans la ligne du haut et les numéros des cases (de 1 à n pour un tableau de n éléments) dans la ligne du bas
    - ◦ Aucune action n'est possible à cet endroit
- Partie du bas
  - ⋆ Onglet `Tableau trié`
    - ◦ Tableau central
      - ∗ C'est le résultat attendu
      - ∗ Aucune action n'est possible à cet endroit
  - ⋆ Onglet `Trier à la main`
    - ◦ Tableau central
      - ∗ C'est le tableau sur lequel les actions de cet onglet sont effectuées
      - ∗ En cliquant sur `Tout cacher`, on cache tout le tableau
      - ∗ En cliquant sur une case cachée, on la rend visible
      - ∗ En cliquant sur une case visible, on la cache

        - Lorsque le tableau a été caché, on ne peut pas avoir plus de deux cases visibles à la fois (sans compter celles qui sont éventuellement écrites en rouge) - sauf en cliquant sur `Tout montrer`
        - Lorsque deux cases exactement sont visibles, on échange leurs contenus en cliquant sur le bouton `Échanger`
        - En double-cliquant (rapidement) sur une case (visible ou non), on signale qu'on estime que le nombre contenu dans cette case est à sa place définitive
            - Si c'est exact, l'élément s'affiche en rouge, et on ne peut plus toucher à cet élément par la suite
            - Si c'est faux, le message `L'élément n'est pas à sa place définitive. Travaillez encore un peu !` s'affiche
    - Bouton `Tout cacher`
        - Permet de cacher toutes les cases de tous les tableaux, sauf celles qui sont éventuellement écrites en rouge
    - Bouton `Échanger`
        - Attention : ce bouton ne fonctionne que lorsque deux cases du tableau exactement sont visibles (sans compter celles qui sont éventuellement écrites en rouge)
        - Échange le contenu des deux cases visibles
    - Bouton `Tout montrer`
        - Permet de rendre visible le contenu de toutes les cases de tous les tableaux
    - Bouton `Recommencer au début`
        - Permet de revenir au tableau initial
- Onglet `Démo d'un algo`
    - Tableau central
        - C'est le tableau sur lequel la démonstration est effectuée
    - Bouton `Démarrer`
        - Permet de commencer la démonstration de l'algorithme sélectionné
        - Dévoile toutes les cases du tableau si elles sont cachées
        - La démonstration est interrompue lorsqu'on appuie sur le bouton `Pause` ou `Un pas en arrière` ou `Un pas en avant` ou lorsqu'on clique sur l'onglet `Trier à la main` ou `Tableau trié`
        - Cliquer de nouveau sur ce bouton pour reprendre la démonstration
    - Bouton `Pause`
        - Permet d'interrompre la démonstration
        - Cliquer sur le bouton `Démarrer` pour reprendre
    - Bouton `Un pas en arrière`
        - Permet de reculer d'un pas dans la démonstration
        - Cliquer sur le bouton `Démarrer` pour reprendre
    - Bouton `Un pas en avant`
        - Permet d'avancer d'un pas dans la démonstration
        - Cliquer sur le bouton `Démarrer` pour reprendre
    - Bouton `Revenir au début`
        - Permet de revenir au tableau initial
    - Bouton `Tri par bulles`
        - Permet de sélectionner la démonstration du tri par bulles
        - Interrompt la démonstration en cours s'il y a lieu
        - Revient au tableau initial
        - Dévoile toutes les cases du tableau si elles sont cachées
    - Bouton `Tri par sélection`
        - Permet de sélectionner la démonstration du tri par sélection
        - Interrompt la démonstration en cours s'il y a lieu
        - Revient au tableau initial
        - Dévoile toutes les cases du tableau si elles sont cachées
    - Bouton `Tri par insertion`
        - Permet de sélectionner la démonstration du tri par insertion
        - Interrompt la démonstration en cours s'il y a lieu
        - Revient au tableau initial
        - Dévoile toutes les cases du tableau si elles sont cachées

2 Suggestion d'utilisation

- Le document ministériel Ressources pour la classe de seconde en algorithmique de Juin 2009 cite les problèmes de tri parmi les exemples possibles en statistiques (page 4), en liaison avec les calculs de médianes et de quartiles.
- Les problèmes de tri ne présentant aucune difficulté mathématique, leur étude en introduction à l'algorithmique permet d'aborder en douceur quelques notions importantes et spécifiques :
  - ⋆ Les instructions de traitement des données sont limitées et stéréotypées : dans cet exemple, la seule possibilité est d'échanger le contenu de deux cases du tableau
  - ⋆ La mémoire est représentée par des variables en nombre fixé à l'avance : par conséquent, à chaque instant on n'a qu'une vision partielle et locale du tableau, qui correspond ici aux deux cases visibles au maximum
  - ⋆ Des instructions de contrôle (alternatives et boucles) sont nécessaires à la description d'un algorithme, puisqu'il doit fonctionner pour un tableau de nombres quelconques de longueur quelconque
- Par contre, à ce stade, il me semble inutile d'imposer à l'élève un formalisme particulier pour rédiger les algorithmes, au risque de noyer les notions sous la technique
- Concrètement,
  - ⋆ Dans le premier exercice, on propose à l'élève de mettre en place une stratégie de tri, puis d'énoncer un algorithme qui devra être suffisamment clair pour pouvoir être mis en œuvre par un autre élève.
  - ⋆ Dans le second exercice, il s'agit de regarder une des démonstrations proposées, de comprendre la méthode utilisée et d'en tirer un algorithme dont la description devra encore une fois être suffisamment claire pour pouvoir être mise en œuvre par un autre élève.
- On pourrait évidemment faire faire les mêmes exercices avec des cartes portant des nombres sur une seule face, l'ordinateur n'est qu'un cosmétique.

`Fiche activité`

Trier un tableau de nombres, c'est ranger ces nombres dans l'ordre croissant. Par exemple, le tableau

6	3	7	2	3	5

devient une fois trié

2	3	3	5	6	7

. Bien sûr, on a souvent besoin de trier d'autre choses que des nombres (comme des mots par ordre alphabétique, des fichiers par longueur, des messages par date, les résultats d'une recherche par pertinence, des articles par référence, des personnes par date de naissance...), mais les algorithmes utilisés sont les mêmes.
Contrairement à ce qui se passe en général dans la vie courante, en informatique, on trie de très grandes quantités de données (quelques centaines de milliers d'éléments) à tout bout de champ. C'est pourquoi les informaticiens ont inventé de très nombreuses méthodes de tri, plus ou moins rapides et efficaces. Les ordinateurs passent énormément de temps à faire des tris : on considère aujourd'hui que les algorithmes de tri sont ceux qui sont les plus utilisés par les ordinateurs du monde entier !

L'objectif du premier exercice est d'inventer une méthode de tri, puis de la décrire de façon assez précise pour que quelqu'un d'autre puisse l'utiliser.

Exercice 1 (Proposer et décrire un algorithme de tri)**.**

1. Utiliser l'application `Tripatouille`, pour créer un nouveau tableau en cliquant sur `Nouveau Tableau`, puis aller dans l'onglet `Trier à la main` et cacher le tableau en cliquant sur `Tout cacher`.
2. Trier ce tableau, en sachant que les seules actions possibles sont les suivantes :
   - Cliquer sur une case cachée la rend visible et cliquer sur une case visible la cache (deux cases écrites en noir au maximum peuvent être visibles en même temps).
   - Lorsque deux cases écrites en noir exactement sont visibles, on peut échanger leur contenu en cliquant sur le bouton `Échanger`.
   - En double-cliquant (rapidement) sur une case, on peut signaler (mais ce n'est pas obligatoire) qu'on estime que le nombre contenu dans cette case est à sa place définitive.
     - ⋆ Si c'est exact, l'élément s'affiche en rouge, et on ne peut plus toucher à cet élément par la suite.
     - ⋆ Si c'est faux, le message `L'élément n'est pas à sa place définitive. Travaillez encore un peu !` s'affiche.
   - C'est tout !
3. Répéter si nécessaire les deux premières questions, puis passer à la question suivante.
4. Décrire une méthode permettant de trier n'importe quel tableau en utilisant cette application.

5. Échanger sa méthode avec celle de son voisin.
6. Créer un nouveau tableau, puis aller dans l'onglet `Trier à la main`, cacher le tableau et le trier en utilisant la méthode décrite par le voisin.

L'objectif du second exercice est de regarder fonctionner un des algorithmes de tri proposés, puis de le décrire de façon assez précise pour que quelqu'un d'autre puisse le reproduire.

Exercice 2 (Comprendre et décrire un algorithme de tri)**.**

1. Utiliser l'application `Tripatouille`, pour créer un nouveau tableau en cliquant sur `Nouveau Tableau`, puis aller dans l'onglet `Démo d'un algo` et choisir l'une des démonstrations d'algorithmes de tri en cliquant sur le bouton correspondant, puis sur le bouton `Démarrer`.
2. Observer ce qui se passe.
   - Le bouton `Pause` permet d'interrompre la démonstration
   - Les boutons `Un pas en avant` et `Un pas en arrière` permettent d'avancer (ou de reculer) d'une seule étape
   - La démonstration reprend quand on clique de nouveau sur `Démarrer`
3. Répéter si nécessaire les deux premières questions, puis passer à la question suivante.
4. Décrire la méthode utilisée de façon à permettre de trier n'importe quel tableau en utilisant l'onglet `Trier à la main`.
5. Échanger sa méthode avec celle de son voisin.
6. Créer un nouveau tableau, puis aller dans l'onglet `Trier à la main`, cacher le tableau et le trier en utilisant la méthode décrite par le voisin.

3 Quelques notions sur les algorithmes de tri classiques

Objectifs de ce paragraphe

- Présenter aux enseignants le problème du tri
- Décrire des algorithmes de tri classiques plus ou moins sophistiqués
- Dire quelques mots de leur complexité
- Attention : Il ne s'agit pas d'un cours exhaustif sur les tris !

3.1 Qu'est-ce-que trier ?

Prototype

- Entrée : un tableau de nombres

6	3	7	2	3	5

- Sortie : le même tableau contenant les mêmes nombres, mais rangés dans l'ordre croissant

2	3	3	5	6	7

Remarque. On convient que les indices des tableaux sont numérotés à partir de 1, et on note $T[i]$ l'élément d'indice i du tableau T. Autrement dit, les tableaux précédents peuvent être écrits plus précisément

Indice	1	2	3	4	5	6
Valeur	6	3	7	2	3	5

et

Indice	1	2	3	4	5	6
Valeur	2	3	3	5	6	7

mais on ne le fera généralement pas pour ne pas alourdir les exemples.

Pourquoi parler des tris ?

C'est un sujet incontournable en algorithmique :

- Premiers algorithmes vraiment utiles dans un cursus classique d'algorithmique (par exemple en DUT Informatique)
- Culture générale : utilisés de façon visible ou non pour l'utilisateur un peu partout en informatique (en particulier en programmation système, en bases de données, en programmation web, etc.)
- Cas d'école en algorithmique : plusieurs algorithmes très différents pour le même problème, illustrant des méthodes classiques
- Occasion idéale pour parler de l'efficacité des algorithmes, c'est-à-dire de *complexité algorithmique*

3.2 Les algorithmes de tri classiques

Il existe deux grandes familles d'algorithmes de tri généralistes :

- Algorithmes simples mais lents
  - ⋆ Tri par sélection
  - ⋆ Tri par insertion
  - ⋆ Tri à bulles
- Algorithmes plus compliqués mais plus rapides
  - ⋆ Tri rapide
  - ⋆ Tri fusion
  - ⋆ Tri par tas
  - ⋆ Etc.

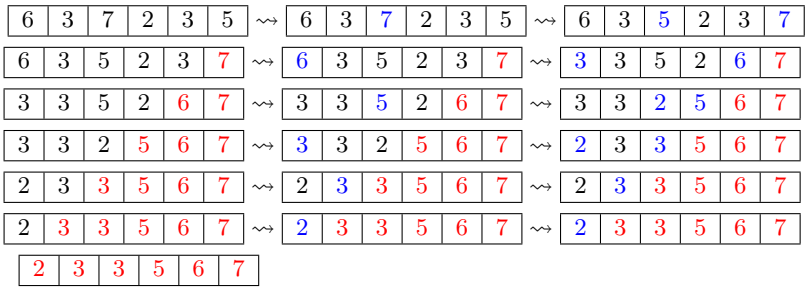
Chacun de ces algorithmes possède des champs d'application privilégiés. Par exemple, le tri par sélection et le tri par insertion sont très simples et efficaces lorsqu'on doit trier un petit nombre d'éléments (jusqu'à 10 environ). Pour des données plus nombreuses, le tri rapide est le plus utilisé actuellement. Mais s'il est rapide en moyenne, il est par contre très lent lorsque le tableau d'entrée est déjà presque trié. Si on sait à l'avance que c'est le cas, on préfèrera utiliser le tri par tas. Il existe aussi de nombreux algorithmes de tri plus spécialisés pour des types de données particuliers, comme le tri par base (radix sort), dont il ne sera pas question ici.

3.2.1 Tri par sélection

Idée

- Sélectionner le plus grand élément du tableau
- Le positionner en dernière position par un échange
- Recommencer sur le reste du tableau

Exemple.



Algorithme

Algorithme 1 : Tri par sélection

Entrée : Un tableau T de n entiers
Résultat : Le tableau T trié

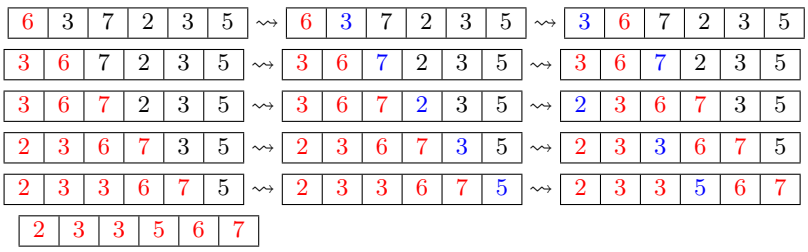
début
  variables locales : Des entiers k, i, $imax$, $temp$
  pour k **de** n **à** 2 **par pas de** -1 **faire**
    `% recherche de l'indice du maximum : %`
    Donner à $imax$ la valeur 1
    pour i **de** 2 **à** k **par pas de** 1 **faire**
      si $T[imax] < T[i]$ **alors**
        Donner à $imax$ la valeur i
      fin
    fin
    `% échange : %`
    Donner à $temp$ la valeur $T[k]$
    Donner à $T[k]$ la valeur $T[imax]$
    Donner à $T[imax]$ la valeur $temp$
  fin
fin

3.2.2 Tri par insertion

Idée

- Le premier élément du tableau est déjà trié
- Insérer le deuxième élément à sa place pour obtenir une liste triée à deux éléments
- Insérer le troisième élément à sa place pour obtenir une liste triée à trois éléments
- Et ainsi de suite

Exemple.



Algorithme

Algorithme 2 : Tri par insertion

Entrée : Un tableau T de n entiers
Résultat : Le tableau T trié

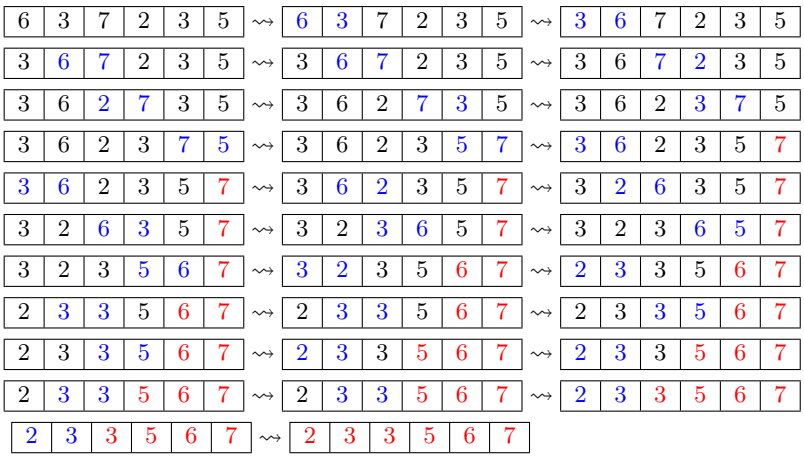
début
  variables locales : Des entiers k, i, v
  pour k **de** 2 **à** n **par pas de** 1 **faire**
    Donner à v la valeur $T[k]$
    Donner à i la valeur $k-1$
    `% décalage des éléments pour l'insertion : %`
    tant que $i \geq 1$ **et** $v < T[i]$ **faire**
      Donner à $T[i+1]$ la valeur $T[i]$
      Donner à i la valeur $i-1$
    fin
    `% insertion proprement dite : %`
    Donner à $T[i+1]$ la valeur v
  fin
fin

3.2.3 Tri par bulles

Idée

- Échanger les deux premiers éléments s'ils ne sont pas dans l'ordre
- Échanger le deuxième et le troisième éléments s'ils ne sont pas dans l'ordre
- Ainsi de suite
- La première passe amène le maximum à sa place définitive
- La deuxième passe amène l'élément juste inférieur au maximum à sa place définitive
- Ainsi de suite

Exemple.



Remarque. Les bulles montent le long du tableau...

Algorithme

Algorithme 3 : Tri par bulles

Entrée : Un tableau T de n entiers
Résultat : Le tableau T trié

```
début
    variables locales : Des entiers k, i, temp
    % pour chaque passe : %
    pour k de n à 2 par pas de −1 faire
        % on fait remonter le plus grand : %
        pour i de 2 à k par pas de 1 faire
            si T[i] < T[i − 1] alors
                % échange de T[i] et de T[i − 1] : %
                Donner à temp la valeur T[i]
                Donner à T[i] la valeur T[i − 1]
                Donner à T[i − 1] la valeur temp
            fin
        fin
    fin
fin
```

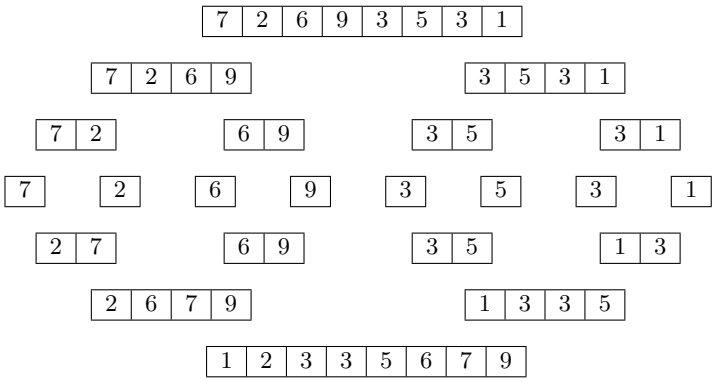
3.2.4 Tri fusion

Idée

- Le tri fusion fonctionne suivant la méthode *diviser pour régner*.
- Trier séparément les deux moitiés du tableau, puis interclasser les deux demi-tableaux déjà triés
- Pour trier chaque demi-tableau, trier séparément les deux quarts de tableau, puis interclasser les deux quarts de tableau déjà triés
- Ainsi de suite

- Un tableau constitué d'une seule case est déjà trié
- Il s'agit d'un algorithme récursif (qui s'appelle lui-même)
- Le tri fusion utilise un tableau auxiliaire de même taille que le tableau initial pour stocker des résultats intermédiaires.

Exemple.



Algorithme

Le cœur de l'algorithme est la fonction d'interclassement de deux parties contigües du tableau qu'on suppose chacune déjà triée :

- Parcourir simultanément les deux parties du tableau principal de gauche à droite
- À chaque instant, choisir le plus petit des deux éléments courant, le copier dans le tableau auxiliaire et se déplacer d'une case dans le tableau auxiliaire et dans la partie du tableau où on vient de choisir un élément.
- Arrêter quand on a parcouru entièrement les deux parties du tableau : on a alors écrit tous les éléments dans l'ordre croissant dans le tableau auxiliaire
- Recopier le résultat dans le tableau principal

Fonction `Interclassement(`T,n,*debut*,*milieu*,*fin*`)`

Entrée : Un tableau T de n entiers, des entiers *debut*, *milieu* et fin
Résultat : Le tableau T interclassé entre *debut* et fin

début
  variables locales : Des entiers i, j, k, un tableau *temp* de n entiers
  Donner à i la valeur *debut*
  Donner à j la valeur $milieu + 1$
  pour k **de** *debut* **à** fin **par pas de** 1 **faire**
    si *(*$j > fin$ **ou** *(*$i \leq milieu$ **et** $T[i] < T[j]$*))* **alors**
      Donner à $temp[k]$ la valeur $T[i]$
      Donner à i la valeur $i + 1$
    sinon
      Donner à $temp[k]$ la valeur $T[j]$
      Donner à j la valeur $j + 1$
    fin
  fin
  `% copier le tableau résultat` *temp* `%`
  `% à sa place dans le tableau` T `%`
  pour k **de** *debut* **à** fin **par pas de** 1 **faire**
    Donner à $T[k]$ la valeur $temp[k]$
  fin
fin

Il ne reste plus qu'à écrire la fonction (récursive) qui effectue le tri fusion :

Fonction `TriFusion(`T,n,$debut$,fin`)`

Entrée : Un tableau T de n entiers et des entiers *debut* et fin
Résultat : Le tableau T trié entre *debut* et fin

début
  variables locales : Un entier *milieu*, un tableau *temp* de n entiers
  si $debut < fin$ **alors**
    Donner à *milieu* la valeur $\lfloor \frac{debut+fin}{2} \rfloor$
    `TriFusion(`T,n,$debut$,$milieu$`)`
    `TriFusion(`T,n,$milieu+1$,fin`)`
    `Interclassement(`T,n,$debut$,$milieu$,fin`)`
  fin
fin

Remarque. Finalement, on trie le tableau T de n entiers numérotés de 1 à n en appelant `Tri fusion(`$T, n, 1, n$`)`.

3.2.5 Tri rapide

Idée

- Le tri rapide fonctionne aussi suivant la méthode *diviser pour régner*.
- La méthode consiste à placer un élément du tableau (appelé pivot) à sa place définitive, en permutant tous les autres éléments de telle sorte que tous ceux qui lui sont inférieurs soient à sa gauche et que tous ceux qui lui sont supérieurs soient à sa droite. Cette opération s'appelle le *partitionnement*.
- Pour chacun des sous-tableaux, on définit un nouveau pivot et on répète l'opération de partitionnement.
- Ce processus est répété jusqu'à ce que l'ensemble des éléments soit trié.
- Il s'agit encore une fois d'un algorithme récursif.

Exemple.
On choisit arbitrairement le dernier élément du tableau comme pivot :

7	2	6	9	3	5	3	1	4	⇝	7	2	6	9	3	5	3	1	4
7	2	6	9	3	5	3	1	4	⇝	1	2	6	9	3	5	3	7	4
1	2	6	9	3	5	3	7	4	⇝	1	2	3	9	3	5	6	7	4
1	2	3	9	3	5	6	7	4	⇝	1	2	3	3	9	5	6	7	4
1	2	3	3	4	9	5	6	7	⇝	1	2	3	3	4	9	5	6	7

Le pivot se trouve à sa place définitive : tous les éléments à sa gauche sont plus petits, tous les éléments à sa droite sont plus grands. On recommence sur les deux demi-tableaux à droite et à gauche (en accéléré...) :

1	2	3	3	4	9	5	6	7	⇝	1	2	3	3	4	6	5	7	9

Et ainsi de suite :

1	2	3	3	4	6	5	7	9	⇝	1	2	3	3	4	5	6	7	9
1	2	3	3	4	5	6	7	9	⇝	1	2	3	3	4	5	6	7	9
1	2	3	3	4	5	6	7	9	⇝	1	2	3	3	4	5	6	7	9

Algorithme de partitionnement

- On veut partitionner le tableau T entre les indices *imin* et *imax* sans toucher au reste du tableau
- On choisit une valeur pivot, par exemple $v = T[imax]$
- On veut séparer dans le tableau les éléments inférieurs à v et les éléments supérieurs à v
- Pour faire cela, on introduit un indice i, initialisé à *imin*, et un indice j, initialisé à $imax - 1$
- On fait ensuite remonter l'indice i jusqu'au premier élément supérieur à v, et simultanément on fait redescendre j jusqu'au premier élément inférieur à v

- On échange alors les éléments d'indices i et j
- On itère ce procédé tant que $i \leq j$.
- À la fin on place la valeur pivot en position i, qui est sa place définitive
- Tous les éléments inférieurs à v ont alors un indice inférieur à i, et tous les éléments supérieurs à v ont un indice supérieur à i

Fonction `Partitionnement(`T,n,$imin$,$imax$`)` $\longrightarrow$ *ipivot*

Entrée : Un tableau T de n entiers, des entiers $imin$ et $imax$
Résultat : Le tableau T partitionné
Sortie : La position définitive du pivot

```
début
    variables locales : Des entiers v, i, j, temp
    Donner à v la valeur T[imax]
    % v valeur pivot %
    Donner à i la valeur imin
    Donner à j la valeur imax − 1
    tant que i ≤ j faire
        tant que i < imax et T[i] < v faire
            Donner à i la valeur i + 1
        fin
        tant que j ≥ imin et T[j] ≥ v faire
            Donner à j la valeur j − 1
        fin
        si i < j alors
            % échange %
            Donner à temp la valeur T[i]
            Donner à T[i] la valeur T[j]
            Donner à T[j] la valeur temp
        fin
    fin
    % on place la valeur pivot en position i %
    Donner à T[imax] la valeur T[i]
    Donner à T[i] la valeur v
    % on renvoie la position définitive du pivot %
    retourner i
fin
```

Algorithme de tri rapide

- Après le partitionnement d'un tableau, le pivot v est à sa place définitive dans le tableau trié
- De plus, tous les éléments qui se trouvent à la gauche du pivot dans le tableau trié sont ceux qui se trouvent à sa gauche après partitionnement
- Le tri rapide consiste à effectuer le partitionnement du tableau en deux parties, puis à trier chacune des deux parties (gauche et droite) suivant le même principe

Fonction `TriRapide(`T,n,$imin$,$imax$`)`

Entrée : Un tableau T de n entiers, des entiers $imin$ et $imax$
Résultat : Le tableau T trié entre $imin$ et $imax$

```
début
    variables locales : Un entier i
    si imin < imax alors
        Donner à i la valeur Partitionnement(T,n,imin,imax)
        % tri de la partie gauche %
        TriRapide(T,n,imin,i-1)
        % tri de la partie droite %
        TriRapide(T,n,i,imax)
    fin
fin
```

Remarque. Finalement, on trie le tableau T de n entiers numérotés de 1 à n en appelant `TriRapide`$(T, n, 1, n)$.

3.2.6 Tri par tas

Idée

- Le tri par tas est basé sur la structure d'arbre binaire.
- On considère que l'élément d'indice 1 du tableau est la racine de l'arbre, et que les deux fils de l'élément d'indice i sont les éléments d'indices $2i$ et $2i + 1$ s'ils existent. Les éléments d'indice 2 et 3 sont donc les fils de la racine.
- Un tas est une arbre binaire tel que :
  - ⋆ Tous les niveaux sont complets sauf éventuellement le dernier, qui peut s'arrêter avant le bout
  - ⋆ La valeur d'un nœud donné est supérieure ou égale à celle de ses deux fils.
- La première étape consiste à transformer le tableau en tas, par insertion successive des éléments.
  - ⋆ Pour $i = 2$ à n, on veut insérer l'élément d'indice i dans un tas représenté par les éléments d'indices 1 à $i - 1$ du tableau
  - ⋆ Le père de cet élément se trouve en position $j = \lfloor \frac{i}{2} \rfloor$
  - ⋆ Si $T[i] > T[j]$, on échange ces deux éléments et on recommence avec le père et le grand-père
  - ⋆ On s'arrête quand on a atteint la racine ou bien quand il n'y a pas d'échange à faire
- Lorsque c'est fait, le tableau est un tas et donc le maximum du tableau se trouve dans la case d'indice 1.
- On permute alors la première et la dernière case du tableau. De ce fait, le maximum du tableau se retrouve à sa place définitive, et le tableau restant n'est plus un tas, mais seule la racine peut être mal placée
- On rétablit la structure de tas sur le tableau restant :
  - ⋆ Si la racine n'est pas supérieure ou égale à ses deux fils, on la permute avec son plus grand fils
  - ⋆ On recommence à la nouvelle position atteinte
  - ⋆ On arrête quand on n'a pas besoin de permuter
  - ⋆ Le tableau restant est de nouveau un tas
- Et on recommence

Exemple.
Construction du tas initial :

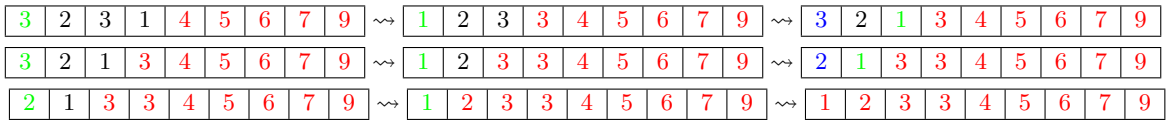
7	2	6	9	3	5	3	1	4	$\rightsquigarrow$	7	2	6	9	3	5	3	1	4
7	2	6	9	3	5	3	1	4	$\rightsquigarrow$	7	2	6	9	3	5	3	1	4
7	2	6	9	3	5	3	1	4	$\rightsquigarrow$	7	2	6	9	3	5	3	1	4
7	2	6	9	3	5	3	1	4	$\rightsquigarrow$	7	9	6	2	3	5	3	1	4
9	7	6	2	3	5	3	1	4	$\rightsquigarrow$	9	7	6	2	3	5	3	1	4
9	7	6	2	3	5	3	1	4	$\rightsquigarrow$	9	7	6	2	3	5	3	1	4
9	7	6	2	3	5	3	1	4	$\rightsquigarrow$	9	7	6	2	3	5	3	1	4
9	7	6	2	3	5	3	1	4	$\rightsquigarrow$	9	7	6	2	3	5	3	1	4
9	7	6	2	3	5	3	1	4	$\rightsquigarrow$	9	7	6	2	3	5	3	1	4
9	7	6	2	3	5	3	1	4	$\rightsquigarrow$	9	7	6	4	3	5	3	1	2

Le tableau est maintenant un tas et donc le maximum se trouve dans la première case. On le met à sa place définitive en l'échangeant avec le dernier élément et on rétablit la structure de tas sur le tableau restant (en accéléré) :

9	7	6	4	3	5	3	1	2	$\rightsquigarrow$	2	7	6	4	3	5	3	1	9	$\rightsquigarrow$	7	4	6	2	3	5	3	1	9

Puis on recommence :

7	4	6	2	3	5	3	1	9	$\rightsquigarrow$	1	4	6	2	3	5	3	7	9	$\rightsquigarrow$	6	4	5	2	3	1	3	7	9
6	4	5	2	3	1	3	7	9	$\rightsquigarrow$	3	4	5	2	3	1	6	7	9	$\rightsquigarrow$	5	4	3	2	3	1	6	7	9
5	4	3	2	3	1	6	7	9	$\rightsquigarrow$	1	4	3	2	3	5	6	7	9	$\rightsquigarrow$	4	3	3	2	1	5	6	7	9
4	3	3	2	1	5	6	7	9	$\rightsquigarrow$	1	3	3	2	4	5	6	7	9	$\rightsquigarrow$	3	2	3	1	4	5	6	7	9



Algorithme

Algorithme 8 : Tri par tas

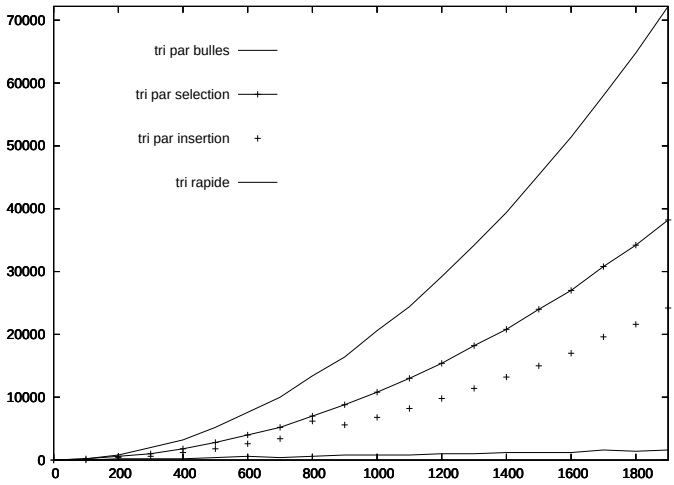
Entrée : Un tableau T de n entiers
Résultat : Le tableau T trié

début
  variables locales : Des entiers k, i, j, j_1, j_2, $temp$, max
  `% Construction du tas initial : %`
  pour k **de** 2 **à** n **par pas de** 1 **faire**
    Donner à i la valeur k
    Donner à j la valeur $\lfloor \frac{k}{2} \rfloor$
    tant que $T[j] < T[i]$ **et** $j \geq 1$ **faire**
      `% échange de` $T[i]$ `et de` $T[j]$ `: %`
      Donner à $temp$ la valeur $T[i]$
      Donner à $T[i]$ la valeur $T[j]$
      Donner à $T[j]$ la valeur $temp$
      `% Mise à jour de` i `et` j `: %`
      Donner à i la valeur j
      Donner à j la valeur $\lfloor \frac{i}{2} \rfloor$
    fin
  fin
  `% Tri proprement dit : %`
  pour k **de** n **à** 2 **par pas de** -1 **faire**
    `% On place le maximum à sa position définitive : %`
    Donner à $temp$ la valeur $T[k]$
    Donner à $T[k]$ la valeur $T[1]$
    Donner à $T[1]$ la valeur $temp$
    `% On rétablit la structure de tas dans le tableau restant : %`
    Donner à i la valeur 1
    Donner à j_1 la valeur $2i$
    Donner à j_2 la valeur $2i + 1$
    tant que *(*$j_1 \leq k$ **et** $T[j_1] > T[i]$*)* **ou** *(*$j_2 \leq k$ **et** $T[j_2] > T[i]$*)* **faire**
      si $j_1 \leq k$ **et** $T[j_1] > T[i]$ **alors**
        Donner à max la valeur j_1
      sinon
        Donner à max la valeur i
      fin
      si $j_2 \leq k$ **et** $T[j_2] > T[i]$ **alors**
        Donner à max la valeur j_2
      fin
      si $max \neq i$ **alors**
        Donner à $temp$ la valeur $T[max]$
        Donner à $T[max]$ la valeur $T[i]$
        Donner à $T[i]$ la valeur $temp$
        Donner à i la valeur max
        Donner à j_1 la valeur $2i$
        Donner à j_2 la valeur $2i + 1$
      fin
    fin
  fin
fin

3.3 Complexité des différents algorithmes de tri

3.3.1 Temps de calcul expérimentaux

Voici des courbes expérimentales qui montrent le temps de calcul (temps d'utilisation du *CPU* en secondes) des algorithmes de tri par bulle, par insertion, par sélection, et du tri rapide, en fonction de la taille n du tableau [1] sur des données aléatoires. Le nombre d'éléments n du tableau varie de 1 à 2000.



Observations

- Sur de grands tableaux, les algorithmes ont des temps de calcul assez différents les uns des autres
- L'écart s'accroît avec la taille des tableaux
- En particulier, la courbes du tri rapide est beaucoup plus plate que celles des tris simples
- Les courbes des temps de calcul du tri fusion et du tris par tas n'apparaissent pas sur ce schéma, mais si c'était le cas, elles ressembleraient à celle du tri rapide.

Question

Comment expliquer les deux types de courbes ?

3.3.2 Justification des résultats expérimentaux

On calcule ci-dessous une estimation du nombre d'opérations élémentaires (comparaisons et affectations) dans le pire des cas pour le tri par sélection puis pour le tri fusion.

Tri par sélection

- **Coût des échanges.** Pour trier n nombres : $n-1$ échanges, c.-à-d. $3(n-1)$ affectations
- **Coût des recherches de maximum**
  - ⋆ Rechercher le maximum parmi n éléments : au plus $4(n-1)$ opérations (tests et affectations)
  - ⋆ Puis rechercher le maximum parmi $n-1$ éléments : au plus $4(n-2)$ opérations
  - ⋆ Puis rechercher le maximum parmi $n-2$ éléments : au plus $4(n-3)$ opérations
  - ⋆ etc.
  - ⋆ Le nombre d'opérations pour les recherches de maximum est de :

$$4\left(1+2+3+\cdots+(n-2)+(n-1)\right) = 4\frac{(n-1)n}{2}$$

- Finalement, le nombre total d'opérations est au maximum de $3(n-1)+4\frac{(n-1)n}{2}$

1. schéma extrait de Algorithmique et structures de données en C, Deuxième édition, Rémy Malgouyres, Dunod, Janvier 2008, page 100

- Polynôme de degré 2 en le nombre n d'éléments du tableau
- La partie de degré 1 en n est négligeable devant la partie en n^2 quand n devient grand
- On dit que l'algorithme est *quadratique*, ou encore qu'il est en $\mathcal{O}(n^2)$

Tri à bulles et tri par insertion
On obtiendrait des résultats similaires pour le cas du tri à bulles et du tri par insertion : ce sont aussi des algorithmes quadratiques.

Tri fusion
- **Coût de l'interclassement.** Interclasser deux tableaux triés de m nombres dans un tableau de $2m$ nombres : au plus $2m$ comparaisons et $2m$ affectations, disons $I_m = 4m$ opérations
- **Coût du tri**. Pour simplifier, on suppose que $n = 2^k$
  - ⋆ trier un tableau de n nombres = interclasser deux tableaux de $n/2$ nombres déjà triés + trier deux tableaux de $n/2$ nombres
  - ⋆ trier un tableau de $n/2$ nombres = interclasser deux tableaux de $n/4$ nombres déjà triés + trier deux tableaux de $n/4$ nombres
  - ⋆ etc.
  - ⋆ puisque $n = 2^k$, il y a k étapes
  - ⋆ Le nombre total d'opérations est de :

$$I_{n/2} + 2 \times (I_{n/4} + 2 \times (\ldots + 2 \times I_1)\ldots) = 4(n/2 + 2 \times (n/4 + 2 \times (\ldots + 2 \times 1)\ldots) = 2kn$$

- Au total, le nombre d'opérations est au maximum de $2kn$ pour $n = 2^k$, c'est-à-dire $2n \log_2 n$
- On dit que l'algorithme est en $\mathcal{O}(n \log n)$

Tri par tas
Le tri par tas est aussi un algorithme en $\mathcal{O}(n \log n)$.

Tri rapide
Le cas du tri rapide est un peu différent : c'est un algorithme quadratique dans le pire des cas, mais ce *pire des cas* ne se produit pas souvent et sa complexité *en moyenne* est aussi en $\mathcal{O}(n \log n)$.

Conclusion
Quand n devient grand, $n \log n$ est négligeable devant n^2, ce qui explique que, sur le schéma précédent, la courbe des temps de calculs pour le tri rapide semble complètement plate par rapport à celle obtenue pour les tris quadratiques.

Références

La documentation concernant les algorithmes de tri est particulièrement abondante, tant sous forme de livres que de polycopiés ou de pages web. Vous trouverez ci-dessous la liste des références que j'ai utilisées pour rédiger ce texte.

- Initiation à l'algorithmique et aux structures de données en C, Deuxième édition, Rémy Malgouyres, Dunod, Janvier 2008
- Algorithmique, Troisième édition, Thomas Cormen, Charles Leiserson, Ronald Rivest, et Clifford Stein, Juin 2010, Dunod
- http ://interstices.info/jcms/c_6973/les-algorithmes-de-tri
- http ://fr.wikipedia.org/wiki/Algorithme_de_tri
- Pédagogie de l'informatique, Maurice Nivat (http ://www.epi.asso.fr/revue/articles/a1005b.htm)
- Computer Science Unplugged - L'informatique sans ordinateur (http ://interstices.info/jcms/c_47072/enseigner-et-apprendre-les-sciences-informatiques-a-l-ecole)