Matière à réfléchir. Bibliothèque inutile

Imaginez une bibliothèque scolaire un peu spéciale : les livres n'y sont pas rangés par ordre alphabétique ! Ils sont bien rangés sur des étagères, mais sans aucune logique particulière. Vous entrez dans cette bibliothèque un peu spéciale et vous vous mettez à chercher l'ouvrage *Le Guide du voyageur galactique*.

Pensez-vous pouvoir retrouver ce livre ? Combien de temps cela vous prendra-t-il ?

Y a-t-il des objets chez vous, que vous rangez dans un ordre bien particulier ?

Y a-t-il des objets chez vous, que vous feriez mieux de ranger dans un ordre bien particulier, parce que vous passez beaucoup de temps à les chercher ?

2. Trie, cherche et trouve

Pour l'instant il faut nous croire sur parole, mais si l'on veut pouvoir trouver une information parmi une avalanche d'informations, il faut que ces informations soient bien rangées. L'exemple de la bibliothèque ci-dessus illustre ce besoin de manière intuitive, mais vous allez pouvoir l'expérimenter concrètement dans le chapitre Algorithmique II.

Saviez-vous que le succès fulgurant de Google est surtout dû à sa capacité à bien ranger l'information disponible sur le Web ? Au moment où vous avez besoin d'une information particulière, leurs algorithmes sont capables de la retrouver parce qu'elle est bien rangée. Ce problème qui consiste à ranger les données a un nom, il s'agit du **problème du Tri**. Il est si important qu'il est un des problèmes les plus étudiés en algorithmique.

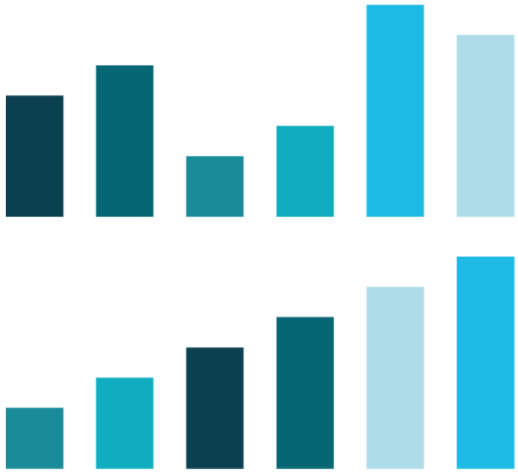


Fig. 5 Problème du tri. Des objets sont triés selon une relation d'ordre, en lien avec une propriété. Sur la ligne du haut, les rectangles sont organisés selon leur couleur (de la plus sombre à la plus claire), alors que sur la ligne du bas, ils sont triés selon leur taille (du plus petit au plus grand).

Algorithmes de tri

Un **algorithme de tri** est un algorithme qui permet de résoudre le problème du tri des données, donc d'organiser les données selon *une relation d'ordre*. Dans la figure ci-dessous, les objets sont organisés

soit par la luminosité de leur couleur (ligne du haut), soit par leur taille (lignes du bas), dans un **ordre croissant**.

Exercice 0. Problème du tri

Trier les rectangles de la ligne du haut de la Figure ci-dessus en fonction de leur taille, pour arriver à la disposition de la ligne du bas. Noter toutes les étapes intermédiaires de vos actions et la disposition des rectangles avant d'arriver à la solution finale. Conseil : remplacer les rectangles par un nombre qui représente leur taille.

En lien avec les ingrédients d'un algorithme, déterminer les données en entrée et le résultat en sortie de l'algorithme.

Quels types d'opérations avez-vous effectuées ?

Solution 0. Problème du tri

Si on remplace les rectangles de la ligne du haut par un nombre qui représente leur taille, on obtient la liste `[3, 4, 1, 2, 6, 5]`. Le plus important est que l'ordre des nombres conserve l'ordre de la taille des rectangles. Après le tri, si l'algorithme est correct, vous devriez vous retrouver avec la liste `[1, 2, 3, 4, 5, 6]`. Les opérations et les dispositions intermédiaires exactes dépendent de l'algorithme que vous avez utilisé.

Nous allons exposer ici **trois algorithmes de tri simple**, que l'on pourrait utiliser pour trier des objetdans la vie de tous les jours.

Tri par insertion

L'algorithme du **tri par insertion** parcourt la liste d'éléments à trier du deuxième au dernier élément. Pour chaque nouvel élément considéré, il l'insère à l'emplacement correct dans la liste déjà parcourue. A tout moment, la liste d'éléments déjà parcourus (jusqu'à l'élément que l'on considère à un moment donné) est toujours bien triée.

Tri par sélection

L'algorithme du **tri par sélection** commence par rechercher le plus petit élément de la liste et l'échange avec le premier élément de la liste. Il recherche ensuite le plus petit élément de la liste restante (sans le premier plus petit élément). Il sélectionne ainsi le deuxième plus petit élément de la liste et l'échange avec le deuxième élément de la liste. Et ainsi de suite : il recherche le plus petit élément de la liste restante, en excluant les éléments déjà triés, et échange ce plus petit élément avec le premier élément pas encore trié. Il continue de la sorte jusqu'à arriver au dernier élément de la liste.

Tri à bulles

L'algorithme du **tri à bulles** compare les éléments voisins, deux par deux. Il commence par comparer les deux premiers éléments de la liste et les met dans le bon ordre (le plus petit des deux éléments précède le plus grand des deux). Il compare ensuite les deux éléments suivants (le nouveau deuxième

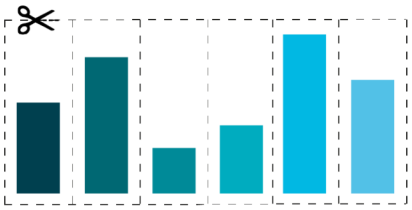
et le troisième élément de la liste) et les met dans le bon ordre. Il continue de la sorte jusqu'à la fin de la liste. Après ce premier parcours de la liste, le plus grand élément se retrouve en dernière position de la liste. L'algorithme parcourt à nouveau la liste, en comparant et en déplaçant les éléments voisins deux par deux (en excluant également le dernier élément qui est déjà bien trié). Après le deuxième parcours de la liste, le deuxième plus grand élément se retrouve en avant-dernière position de la liste. L'algorithme parcourt la liste de la sorte, autant de fois qu'elle possède d'éléments, en excluant les éléments bien triés en fin de la liste.

Exercice 1. Algorithme de tri

Il est fortement recommandé de résoudre cet exercice avant d'avancer dans le chapitre.

Appliquer au moins un des trois algorithmes ci-dessus (tri par insertion, tri par sélection et tri à bulles) pour trier les rectangles de la ligne du haut de la Figure Problème du tri en fonction de leur taille (le résultat est illustré dans la ligne du bas).

Noter l'ordre des éléments à chaque fois qu'il change. Vous aurez besoin d'une grande feuille de papier. Vous pouvez aussi représenter la taille des rectangles par un nombre, cela permet de gagner de la place. Si cela vous aide, vous pouvez découper les rectangles ci-dessous et les manipuler.



Solution 1. Algorithme de tri

La solution est donnée dans la suite du chapitre et est illustrée dans la Figure Algorithmes de tri.

Anecdote

Vous passez trop de temps à chercher vos affaires ? Pensez à mieux les trier. Le temps perdu à ranger vos affaires sera bien inférieur à celui que vous passerez à les chercher plus tard.

La prochaine figure détaille les étapes intermédiaires des trois algorithmes de tri vus précédemment. Dans le **tri par insertion** à gauche, on parcourt la liste dans l'ordre, un élément après l'autre (dénoté par une étoile). A chaque étape, on cherche à *insérer* le rectangle considéré à la bonne place dans la liste précédemment triée. La flèche rouge montre la position à laquelle le rectangle sera inséré après comparaison avec l'élément précédent. Si l'élément est déjà bien trié, aucune action n'est requise dans ce cas et la flèche est remplacée par un point rouge. Notez que la liste qui précède le rectangle considéré (celui avec l'étoile) est toujours bien triée.

Dans le **tri par sélection** au milieu, on parcourt la liste pour *sélectionner* son plus petit élément, et on le met à la bonne position. La ligne noire au-dessous des rectangles montre la liste parcourue pour rechercher le plus petit élément. Le plus petit élément de cette liste est désigné par l'étoile. Finalement, la flèche rouge montre les éléments échangés : le premier élément de la liste non triée et le plus petit élément. Ainsi, le plus petit élément sélectionné (avec étoile) se retrouve à la fin de la liste déjà triée (liste non soulignée). Si l'élément est déjà bien trié et qu'aucune action n'est requise, la flèche bidirectionnelle est remplacée par un point rouge.

Dans le **tri à bulles** à droite, les lignes en dessous des rectangles montrent les éléments voisins qui sont comparés à chaque étape. Lorsque cette ligne est grise, les éléments sont déjà bien ordonnés et aucune action n'est requise. Lorsque la ligne est noire, les éléments ne sont pas dans le bon ordre et doivent être intervertis (flèche rouge). Après un passage complet de la liste, l'élément le plus grand se retrouve en dernière position, il remonte comme une *bulle* (voir la 4e ligne). Le point rouge ici indique les éléments triés. Dans ce cas, la liste est triée après deux parcours complets de la liste.

Notez que même si tous les algorithmes arrivent à la même solution finale, ils y arrivent de manière très différente et avec plus ou moins de calculs.

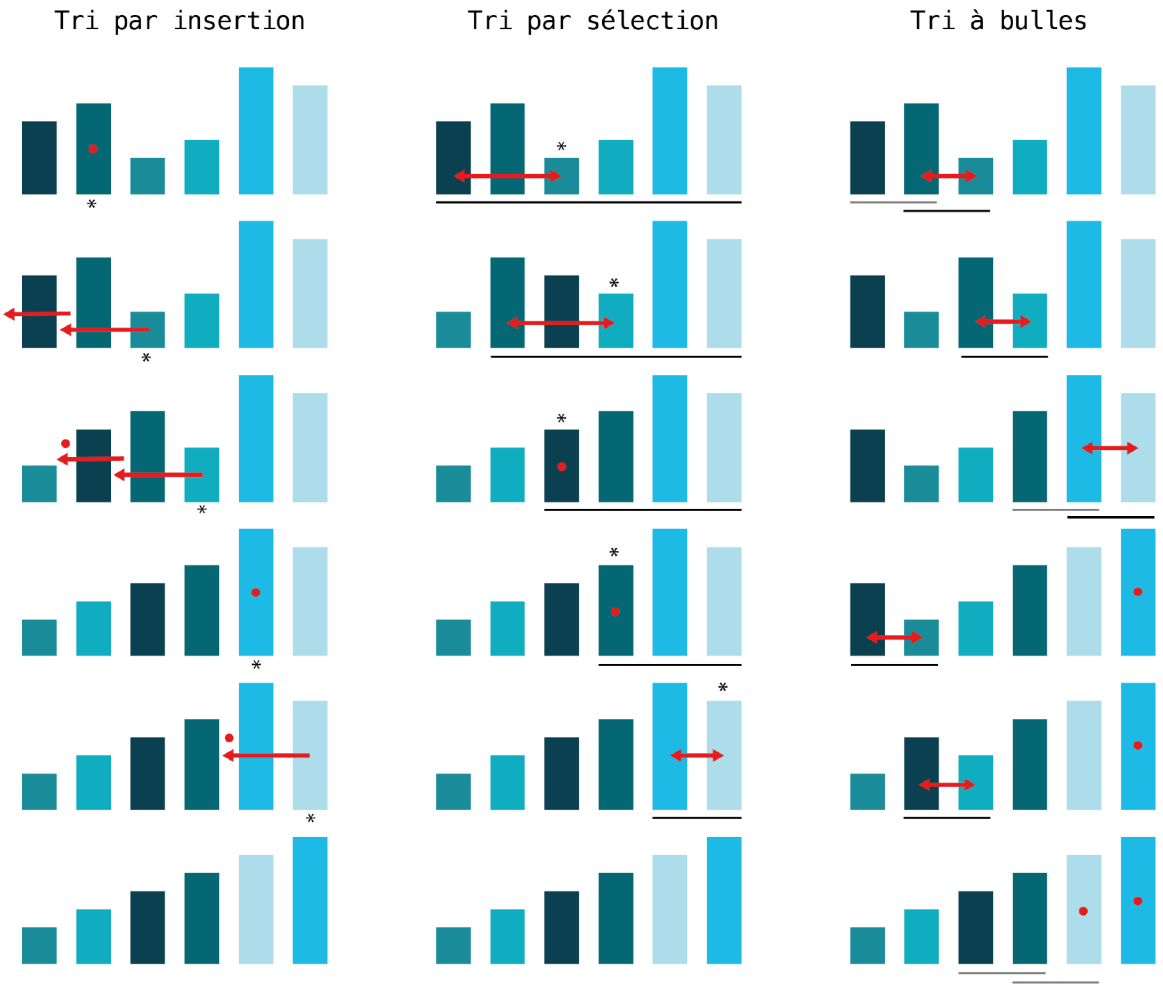


Fig. 6 Algorithmes de tri. Etapes intermédiaires lors de l'application des différents algorithmes de tri. La flèche rouge montre les mouvements des éléments suite à une opération. Si l'élément ne bouge pas, la flèche rouge est remplacée par un point rouge. A gauche, le tri par insertion. L'étoile dénote l'élément considéré à un moment donné. Au milieu, le tri par sélection. L'étoile désigne le plus petit élément de la liste non triée. A droite, le tri à bulles. Ici le point rouge signale les éléments triés.

Exercice 2. Votre algorithme de tri

Rappelez-vous quelle méthode vous avez utilisée pour résoudre l'exercice 0. De quel algorithme de tri se rapproche-t-elle le plus ?

Solution 2. Votre algorithme de tri

Cela dépend de votre solution de l'exercice 0. Vous avez probablement utilisé la méthode du tri par sélection ou du tri à bulles.

Exercice 3. Opérations

Pour chaque algorithme de tri, compter le nombre de *comparaisons* de la taille de deux rectangles, ainsi que le nombre de *déplacements* (le nombre de fois que deux rectangles échangent leur place).

Imaginons que ce qui prend le plus de temps est une **comparaison**. Dans ce cas précis, quel algorithme de tri parmi les trois algorithmes présentés est le plus lent ?

Imaginons que ce qui prend le plus de temps est un déplacement. Dans ce cas précis, quel algorithme de tri est le plus lent ? Quel algorithme est le plus rapide ?

Solution 3. Opérations

Le décompte des opérations effectuées, en se référant à la Figure Algorithmes de tri est comme suit :

Tri par insertion : 9 comparaisons deux par deux (flèches et points rouges) et 5 déplacements deux par deux (flèches rouges). Notez que pour insérer un élément en première position, il faut tout d'abord l'échanger avec l'élément juste devant, puis avec l'élément avant, et ainsi de suite jusqu'à arriver à la première position.

Tri par sélection : 15 comparaisons deux par deux (lignes en dessous) et 3 déplacements deux par deux (flèches rouges).

Tri à bulles : 9 comparaisons deux par deux (lignes en dessous) et 5 déplacements deux par deux (flèches rouges).

Si ce qui prend beaucoup de temps est la comparaison de la taille de deux rectangles, il ne faudrait pas utiliser le tri par sélection, car il comporte le plus grand nombre de comparaisons et il serait le plus lent. Si c'est le déplacement de deux rectangles qui coûte beaucoup de temps, cette fois-ci le tri par sélection serait le plus rapide (avec 3 rectangles qui échangent leur position). Donc, selon l'implémentation sur la machine, le tri par sélection serait le plus lent ou le plus rapide des trois algorithmes.

Ces résultats sont valables pour cette configuration en particulier. Si on trie un autre tableau, la performance des trois algorithmes pourrait changer. Le choix du meilleur algorithme dépend donc de l'implémentation et de la situation initiale. Notez finalement qu'il existe des algorithmes de tri bien plus rapides que les trois algorithmes considérés ici.

Le saviez-vous ? Tri stupide

Il existe un algorithme, **Tri de Bogo** (ou *Bogosort*), aussi nommé le *tri lent* ou encore le *tri stupide*. C'est un tri qui génère différentes permutations des éléments de la liste et s'arrête lorsque la configuration obtenue par hasard est triée. A votre avis, combien d'opérations prend cet algorithme en moyenne ?

Comparaison d'algorithmes

Toutes les recettes de cuisine ne se valent pas, de la même manière, un algorithme peut aussi être *plus approprié* qu'un autre algorithme pour résoudre le même problème. Il existe des dizaines d'algorithmes qui trient avec des approches différentes (nous en verrons encore quelques-uns). Certains algorithmes sont plus rapides, d'autres plus économes en mémoire ou encore plus simples à coder. Ainsi, selon la situation, il faut choisir le « bon » algorithme.

La qualité d'un algorithme dépend de la propriété que l'on souhaite optimiser (maximiser ou minimiser). Cela pourrait être de maximiser la **vitesse d'exécution** (mesurée par le nombre d'instructions élémentaires exécutées), de minimiser la place occupée en **mémoire**, de minimiser la **consommation d'énergie** ou de maximiser la **précision de la solution**. L'algorithme utilisé devrait être choisi en fonction de ce qui est important.

La vitesse d'un algorithme dépend également des données en entrée. Selon la configuration initiale des données en entrée (correspond à la ligne du haut de la Figure Algorithmes de tri), un algorithme « rapide » peut devenir « lent » et vice versa. Il faut savoir que les algorithmes vus jusqu'ici sont tous des algorithmes lents (nous verrons un algorithme de tri rapide ultérieurement).

Le saviez-vous ? Tri trop lent

Pour trier 1 million d'éléments, selon l'algorithme choisi, cela peut prendre de 20 millions à 1 billion d'opérations. Si chaque opération prenait 1 microseconde (10^{-6} s) à s'exécuter, il faudrait 20 secondes pour trier 1 million d'éléments si l'algorithme est efficace. Par contre, pour un des algorithmes ci-dessus, cela pourrait prendre 11 jours !

Pour aller plus loin

Imaginer que les quatre éléments d'une liste sont triés dans le sens inverse de ce que l'on souhaite (ils sont placés du plus grand au plus petit). Trier la liste selon les trois algorithmes de tri vus précédemment : le tri par insertion, le tri par sélection et le tri à bulles.

Dans cette configuration précise, quel algorithme est le plus rapide (présente le moins d'étapes intermédiaires) ? Quel algorithme est le plus lent ?

Exercices

Exercice 4. L'algorithme de votre journée

Réfléchir à votre journée : y a-t-il des actions qui se retrouvent chaque jour ouvrable ? Arrivez-vous à esquisser un algorithme que vous suivez sans que vous en ayez conscience ?

Exercice 5. Trois algorithmes de tri

Trier la liste `[2, 5, 3, 4, 7, 1, 6]` en utilisant les trois algorithmes de tri vus dans le cours. Représenter l'état de la liste après chaque étape.

Exercice 6. Vérificateur de tri

Ecrire un algorithme qui vérifie si une liste est triée.

Que prend l'algorithme en entrée et que retourne-t-il en sortie ?

Demander ensuite à un autre élève de suivre les opérations décrites par votre algorithme. Est-ce que votre algorithme est correct ?

Comparer vos algorithmes. Sont-ils différents ?

Exercice 7. Mondrian

Analyser les œuvres cubistes de Piet Mondrian. Trouver un algorithme qui permet de créer une œuvre qui pourrait être attribuée à Mondrian.

Ai-je compris ?

1. Je sais qu'il existe plusieurs manières différentes de résoudre un problème.
2. Je sais qu'il faut choisir le meilleur algorithme en fonction de critères objectifs : vitesse de l'algorithme, qualité de la solution, espace utilisé en mémoire ou encore consommation d'énergie.
3. Je sais appliquer les trois algorithmes de tri vus dans le cours.