Rappels sur les algorithmes de Première

1. L'algorithme de **recherche dichotomique** permet de rechercher plus efficacement un entier dans une liste triée.

Le principe consiste à comparer la valeur recherchée `elm` avec la valeur au milieu de la liste `m` :

- Si `elm == m`, la recherche est terminée et on renvoie `True` ;
- Sinon, si `elm < m`, on recommence la recherche dans la partie gauche de la liste ;
- Sinon, on recommence la recherche dans la partie droite de la liste.

Si `elm` n'est pas dans la liste, on renvoie `False`.

L'efficacité de cet algorithme réside dans le fait que l'on ne parcourt pas toute la liste.

→ Dans un algorithme "naïf", où l'on parcourt toute la liste, dans le pire des cas :

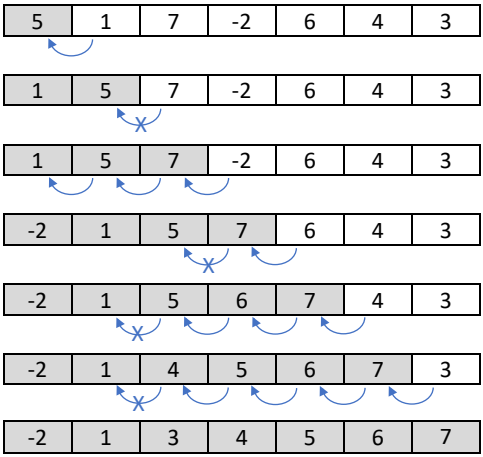
Taille de la liste	10	100	1 000	10 000
Nombre de tours de boucle	10	100	1 000	10 000

→ Dans un algorithme de recherche dichotomique, dans le pire des cas :

Taille de la liste	10	100	1 000	10 000
Nombre de tours de boucle	4	7	10	14

2. Les algorithmes **tri par insertion** et **tri par sélection** permettent de trier des listes de nombres dans l'ordre croissant ou décroissant.

Le principe du tri par insertion consiste en l'insertion d'un nombre dans une liste triée.



On considère que le premier nombre est trié (zone grise = zone triée).

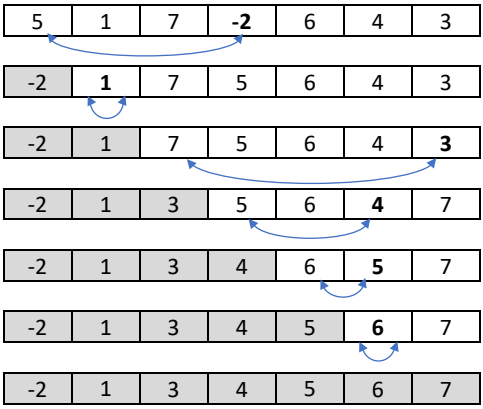
a. On prend le nombre d'indice `j` à droite de la partie triée de la liste `L`.

b. On compare le nombre à l'indice `j` avec le nombre à sa gauche (`j-1`) :

- Si `L[j] < L[j-1]`, on échange leur valeur dans la liste et on recommence l'étape **b.** ;
- Sinon, le nombre à l'indice `j` est bien placé et on recommence l'étape **a.** .

Quand toute la liste a été parcourue, on considère que la liste est triée.

Le principe du tri par sélection consiste en la sélection de la plus petite valeur d'une liste.



(zone grise = zone triée)

a. On cherche dans la partie non triée le nombre le plus petit.

b. On échange la position dans la liste de ce nombre avec le premier nombre de la partie non-triée ;

c. On recommence l'étape **a**. Jusqu'à l'avant-dernier nombre.

Quand tous les nombres de la liste ont été parcourus (sauf le dernier), on considère que la liste est triée.

3. Les **algorithmes gloutons** permettent d'obtenir un résultat optimal à un problème de décision.

Le principe est de faire une sélection de données avec les meilleures caractéristiques possibles sans dépasser un maximum et/ou jusqu'à atteindre un objectif.

Les principales étapes :

a. Les données sont triées par ordre décroissant en fonction de leurs caractéristiques (de la donnée la plus pertinente à la moins pertinente).

b. On sélectionne la 1ère donnée : Si elle ne dépasse pas le maximum, elle est utilisée.

c. On recommence avec cette donnée ou la donnée suivante jusqu'à atteindre l'objectif.

Les exemples les plus connus :

- Le rendu de monnaie : Rendre le moins de pièces/billets pour une somme donnée.
  L'algorithme consiste à rendre la pièce/billet avec la valeur la plus élevée et inférieure à la somme à rendre. On répète le processus tant que la valeur de cette pièce est inférieure à la somme, sinon on répète avec une pièce d'une valeur plus faible.
- Le problème du sac à dos : Remplir un sac avec le plus d'objets ayant les meilleurs ratios valeur/poids.
  Le principe reste le même que le rendu de monnaie, en remplaçant la valeur des pièces par le ratio des objets et la somme à rendre par le poids maximum du sac.

Un algorithme glouton est plus rapide qu'un algorithme "naïf" qui teste toutes les solutions possibles. Cependant, il ne trouve pas toujours le meilleur résultat, mais l'un des meilleurs (tout dépend des données et du problème à résoudre).

4. Les **algorithmes k plus proches voisins** permettent une forme d'apprentissage automatique.

Ces algorithmes utilisent la proximité d'un élément avec d'autres pour faire des classifications ou des prédictions.

Le principe est de récupérer les `k` données dont les caractéristiques sont les plus proches d'une donnée non-classée `nc` pour faire des déductions.

Les principales étapes :

a. On calcule une distance entre la donnée `nc` et les données connues, en fonction de leurs caractéristiques communes.

b. On tri ensuite les données en fonction de leur distance avec `nc`, dans l'ordre croissant, ce qui permet de sélectionner les k plus proches données.

c. Ensuite, on peut faire des déductions de `nc`.

Un exemple connu :

- Le problème des iris : Connaître l'espèce d'un nouvel iris en fonction des caractéristiques de ces pétales et de celles des iris déjà connus.

  On supposera alors que l'espèce du nouvel iris est l'espèce majoritaire parmi les k plus proches iris.

Tout comme l'algorithme glouton, les algorithmes k plus proches voisins ne permettent pas toujours d'obtenir le meilleur résultat.

Plusieurs facteurs sont à prendre en compte :

→ Le nombre de données connues ✓ Beaucoup de données ✗ Peu de données

→ La qualité de ces données ✗ Une catégorie surreprésentée ✓ Equilibre

→ Erreurs dans ces données

→ Le k utilisé :