ALGO 1

Tris

- ▷ Tri par insertion : $O(n^2)$
  Considérer chaque élément un à un pour l'insérer à sa bonne place (penser à un jeu de cartes)
- ▷ Tri fusion : $O(n \log n)$
  Paradigme diviser pour régner, diviser en deux sous-problèmes
- ▷ Tri Shell : $O(n^2)$
  Suite de tris par insertion sur chaque constituant d'une partition du tableau
- ▷ Tri par tas : $O(n \log n)$
  Utiliser une structure de file de priorité, ici un tas
- ▷ Optimalité : $\Omega(n \log n)$ nécessaire, arbres de décision

Structures de données

- ▷ Files de priorité : implémentées par exemple avec un tas
  Enfiler, défiler un élément maximal, est vide ?, construire file vide
- ▷ Tables de hachage
  Ajout, suppression, contient ?, risque de collision
- ▷ Structure Union-Find
  créer partition, fusionner deux classes (union), obtenir un représentant (find)
  Implémentation par une forêt d'arbres, compression de chemin

Arbres binaires

- ▷ Arbre binaire : $1 + h \leq n \leq 2^{h+1} - 1$
- ▷ Arbre binaire presque complet : $2^h \leq n \leq 2^{h+1} - 1$
- ▷ Tas
- ▷ Arbre binaire de recherche (ABR)
  La recherche d'un élément ne suit qu'une branche, problème si arbre non équilibré
- ▷ Arbre AVL
  Rééquilibrage d'un arbre par des rotations : $\log_2(n+1) \leq h \leq 1.44 \log_2 n$

Graphes

- ▷ Graphes orientés, pondérés

- ▷ Implémentations par liste d'adjacence ou matrice d'adjacence
- ▷ Parcours en profondeur
  Valeurs de `Pre` et `Post` traitement, types d'arc, détection de cycles, tri topologique, composantes fortements connexes (*Algorithme de Kosaraju*), graphe quotient
- ▷ Parcours en largeur
  Recherche d'un plus court chemin (*Algorithme de Dijkstra*, *algorithme* A^*)
- ▷ Arbre courant de poids minimal (*Algorithme de Kruskal*, *Algorithme de Prim*)

Algorithmes gloutons

- ▷ Prendre un choix localement meilleur
- ▷ Algorithmes de Kruskal, de Prim
- ▷