STRUCTURES DE DONNÉES ET ALGORITHMES · SYNTHÈSE

Synthèse des structures de données et algorithmes

Révisez les structures, recherches, tris et parcours de graphes avec leurs hypothèses de coût, leurs invariants et leurs cas limites.

Par Ali ZAINOUL · Cours gratuit

Les fiches par thème

Contrats et coûts

Relier opérations, invariants et modèles de coût.

Lire la fiche · Télécharger le PDF

Tableaux, listes, piles et files

Choisir une organisation selon les accès et les retraits.

Lire la fiche · Télécharger le PDF

Dictionnaires et ensembles

Comprendre hachage, égalité, collisions et déduplication.

Lire la fiche · Télécharger le PDF

Arbres de recherche et tas

Distinguer ordre de recherche, hauteur et priorité minimale.

Lire la fiche · Télécharger le PDF

Recherches et tris

Choisir selon ordre initial, stabilité, mémoire et garanties.

Graphes, BFS, DFS et Dijkstra

Choisir un parcours selon les arêtes et le résultat attendu.

Lire la fiche · Télécharger le PDF

Lire la fiche · Télécharger le PDF

Glouton, programmation dynamique et retour sur trace

Justifier un choix local, réutiliser des sous-problèmes ou explorer des choix.

Lire la fiche · Télécharger le PDF

Lire un coût avec ses hypothèses

La taille n compte les éléments ; pour un graphe, n compte les sommets et m les arêtes. Le modèle suppose des clés et opérations de taille bornée. O donne une borne supérieure asymptotique ; Θ une borne serrée. Pire cas, moyenne et amortissement répondent à des questions distinctes.

Une borne de mémoire doit préciser si elle inclut la sortie et les copies. Les listes Python, vecteurs MATLAB et réalisations pédagogiques ne possèdent pas automatiquement les mêmes garanties.

Comparer les structures

Modèles usuels, sous leurs hypothèses

Structure	Opération favorable	Limite à retenir
Tableau	Accès indicé Θ(1)	Insertion au début Θ(n) ; concaténation et copie à compter

Structure	Opération favorable	Limite à retenir
Liste chaînée	Modification locale de liens Θ(1) après un nœud connu	Recherche du nœud O(n) au pire, allocation séparée
Pile LIFO	Ajout et retrait au sommet O(1) amorti avec une liste dynamique adaptée	Vide, capacité et éventuelle réallocation
File FIFO	Ajout arrière et retrait avant O(1) avec stockage adapté	Retrait pop(0) sur liste Python Θ(n)
Table de hachage	Clé recherchée en O(1) moyen sous hypothèses de distribution et de charge	Collisions, taille des clés et pire cas O(n)
Arbre de recherche	Recherche O(h + 1), donc O(log n) si équilibré	Hauteur Θ(n) possible sans équilibrage
Tas minimum	Minimum Θ(1), restauration après ajout ou retrait O(log n)	Le tableau n'est pas entièrement trié ; croissance du stockage séparée

Comparer recherches et tris

Coûts des méthodes étudiées

Méthode	Temps	Condition ou invariant
Recherche linéaire	Θ(1) au meilleur cas non vide ; Θ(n) au pire	Aucune position déjà examinée ne contient la cible
Dichotomie	O(log n) au pire pour n ≥ 2	Entrée triée, accès central constant, intervalle candidat conservé
Insertion	Θ(n) au meilleur cas ; Θ(n²) au pire	Préfixe trié ; décalage strict pour la stabilité

Méthode	Temps	Condition ou invariant
Fusion	Θ(n log n) pour n ≥ 2	Fusion de deux suites triées ; O(n) mémoire au pic dans les versions étudiées
Tri rapide avec groupes copiés	Θ(n log n) si équilibré ; Θ(n²) au pire	Pivot fixe sans garantie aléatoire ; mémoire copiée pouvant atteindre O(n²)

Un tri correct conserve toutes les occurrences, pas seulement l'ordre. La stabilité conserve en plus l'ordre relatif des éléments de même clé. Vérifiez aussi si l'opération modifie l'entrée ou produit un nouveau résultat.

Choisir un parcours de graphe

Les listes d'adjacence occupent Θ(n + m) emplacements ; la matrice occupe Θ(n²). Le coût d'un parcours dépend de ce choix.

Représentation par listes d'adjacence

Méthode	Besoin	Coût et vigilance
BFS	Distances minimales en nombre d'arêtes	O(n + m), file FIFO ; marquer à la découverte
DFS	Accessibilité et exploration en profondeur	O(n + m) avec marques partagées ; copies et appels à compter séparément
Dijkstra	Distances pondérées, poids non négatifs	O(n² + m) pour la sélection linéaire présentée ; le coût d'une version à tas diffère

Choisir une stratégie

- Glouton : figer un choix local exige une preuve, comme l'argument d'échange des activités par date de fin.

- **Programmation dynamique** : définir un état, un cas de base, une récurrence et un ordre respectant les dépendances.
- **Retour sur trace** : choisir, explorer, restaurer ; copier les solutions avant de modifier l'état partagé.

Vérifier un choix

▼ Une application extrait souvent l'urgence minimale et cherche parfois un identifiant. Un tas suffit-il à rendre les deux opérations logarithmiques ?

Non. Le tas favorise l'accès au minimum et sa suppression, mais une recherche de clé arbitraire peut rester linéaire. Si les deux opérations dominent, il faut envisager un index supplémentaire et maintenir la cohérence des deux représentations à chaque modification.

▼ Pourquoi trier une collection avant une seule recherche peut-il être inutile ?

Le tri ajoute son coût et peut dépasser celui d'un simple parcours. La dichotomie devient intéressante si l'ordre est déjà disponible ou si plusieurs recherches amortissent la préparation.

Revenir au programme complet ou **mettre ces choix en pratique dans le projet de synthèse**.

Version web : https://www.clearcode.fr/courses/programming/dsa/cheatsheet