Chapitre 26

Stratégies Algorithmiques

Sommaire.

Les propositions marquées de ★ sont au programme de colles.

1 Stratégies.

Définition 1: Force brute.

idée: Explorer tous les candidats possibles.
contraintes: Univers dénombrable, on sait tester si une entrée est solution.

Définition 2: Backtracking.

idée: Construire une solution pas à pas et revenir sur le dernier choix en cas d'impasse.

Définition 3: Algorithmes gloutons.

idée: Parier sur les maxima locaux.
contraintes: La solution n'est pas toujours optimale.

Définition 4: Programmation dynamique.

Cette méthode est envisageable si :
1. Sous-problèmes optimaux: La solution pour une entrée donnée s'exprime en fonction des solutions pour des entrées strictement plus petites.
2. Chevauchement de sous-problèmes: La solution naïve mène à calculer plusieurs fois les mêmes solutions.

Approche de bas en haut: Calculer les résultats dans l'ordre pour avoir les solutions quand on en a besoin. On peut les garder en mémoire (tableau...)
Mémoïsation: On crée une structure stockant les solutions. Au moment du calcul, on vérifie si la solution a déjà été calculée, sinon on la rajoute.

2 Exemples.

Exemple 5: Rendu de monnaie.

Problème: On dispose d'un nombre illimité de pièces de valeurs $n_1 > n_2 > ... > n_k$. Comment arriver à une certaine somme S avec le moins de pièces ?

L'algorithme glouton naturel consiste à puiser dans les pièces par ordre décroissant de valeurs sans dépasser S.

Exemple 6: Placement d'activités.

Problème: attribuer des salles pour le plus de cours possibles.
Théorème: Le choix du cours terminant le plus tôt est optimal.

Solution :

Supposons que ce choix mène à la liste de cours $(c_1, ...c_n)$. Montrons par l'absurde qu'on ne peut pas avoir de suite plus longue par récurrence descendante sur la longueur du plus long préfixe commun.
Cas de base: $(d_1, ..., d_n, ...)$ avec $d_i = c_i$ pour $i \in [\![1, n]\!]$.
On peut alors construire $(c_1, ..., c_n, c_{n+1})$, car d_{n+1} est plaçable : contradiction.
Hérédité: Pour $k < n$, il ne peut pas exister une suite de cours plaçable de longueur $n + 1$ ayant un préfixe commun de longueur au moins $k + 1$ avec $(c_1, ..., c_n)$.
Supposons qu'il existe une suite de cours plaçables $(c_1, ..., c_k, d_{k+1}, ..., d_{n+1})$, c_{k+1} est plaçable entre c_k et d_{k+1}, ce qui est absurde.

Exemple 7: Distance de Levenshtein

C'est une distance sur l'ensemble des mots d'un dictionnaire. Elle représente le nombre minimal d'opérations élémentaires pour passer d'un mot à un autre : substitution, délétion et insertion.
Soient deux mots u et v. On note $d[i, j]$ la distance de Levenshtein entre le préfixe de longueur i de u et le préfixe de longueur j de v.
On a :

$$d[i,j] = \min \begin{cases} d[i, j-1] + 1 \\ d[i-1, j] + 1 \\ d[i-1, j-1] + \delta_{u[i] \neq v[j]} \end{cases}$$

De plus, $d[0, j] = j$ et $d[i, 0] = i$.
Complexité de l'algorithme naïf:
$T(m, n) = \alpha + T(m-1, n) + T(m, n-1) + T(m-1, n-1)$.
En simplifiant: $T(m, n) \geq \alpha + 3T(m-1, n-1)$.
On pose $U(n) = T(n, n) \geq \alpha + 3U(n-1)$ donc $U(n) \geq \alpha n + 3^n U(0) = \Omega(3^n)$.

Complexité de l'algorithme dynamique:
On fait un nombre constant d'opérations pour chaque axe du tableau donc une complexité en $O(mn)$.

Exemple 8: Typographie

Hypothèses: Police de longueur fixe, un seul espace entre deux mots.
Entrée: Une suite de n mots de longueurs $l_1, ..., l_n$ et la largeur M de la ligne tels que $\forall i,\ l_i \leq M$.
Contraintes:

- Aucune ligne ne dépasse : si une ligne contient les mots i à j, on a : $\sum_{k=i}^{j} l_k - i + j \leq M$
- L'espacement est harmonieux sur l'ensemble des lignes : on minimise la somme cubes des espaces finaux sur chaque ligne, sauf la dernière.

Montrons que la propriété de sous-problèmes optimaux est vérifiée.
Soit une solution optimale sur h lignes, alors la composition est optimale sur les $h - 1$ premières lignes.
Notons $\mathcal{E}(i)$ la somme des cubes des espaces finaux de lignes, sauf la dernière à partir du mot i jusqu'au dernier, en supposant que le mot i est en début de ligne.
⊚ Si on peut écrire tous les mots restants sur une seule ligne, il suffit de le faire :

$$E(i) = 0 \quad \text{si} \quad \sum_{k=i}^{n} l_k + n - i \leq M$$

⊚ Sinon, il faut trouver j tel que :

$$E(i) = \min\{(M - (\sum_{k=i}^{j} l_k + j - i))^3 + E(j+1) \mid j \in [\![1, n]\!] \ \wedge\ \sum_{k=i}^{j} l_k + j - i \leq M\}.$$