2ALGO

Justification

SOMMAIRE

I – Stratégie Gloutonne

Cette stratégie est-elle optimale ?

La stratégie gloutonne n'est pas optimale dans tous les cas car elle se base uniquement sur des choix locaux optimaux à chaque étape, sans considérer l'impact de ces choix sur l'ensemble du problème.

Cette stratégie sélectionne à chaque étape l'option qui semble la meilleure localement. Mais cela peut conduire à des situations où une décision apparemment optimale à un certain stade peut entraîner une solution globale moins optimale.

Voici un exemple :

T = [[1, 100],

    [1, 1]]

La stratégie gloutonne sélectionnerait l'essence d'index 1 pour le premier emplacement puis l'essence d'index 2 pour le deuxième emplacement, avec donc un coût total de : 101.

Cependant, la solution optimale serait de choisir l'essence d'index 2 pour le premier emplacement puis l'essence d'index 1 pour le deuxième emplacement, avec un coût total de : 2.

C'est pour quoi la stratégie gloutonne ne garantit pas l'optimalité de la solution dans tous les cas.

II - Récurrence et récursivité naïve

Etablir une formule de récurrence portant sur g[i][j] et justifier son raisonnement.

Estimer sa complexité.

Tout d'abord, pour obtenir la formule de récurrence pour g[i][j] nous devons prendre en compte le coût minimal obtenu en plantant l'essence j à l'emplacement i et aussi prendre en considération les plantations optimales effectuées aux emplacements précédents (1,…, i-1).

La formule de récurrence est donc la suivant :

g[i][j] = T[i][j] + min(g[i-1][k]) pour tout k ≠ j, avec 1 ≤ i ≤ n et 1 ≤ j ≤ m

- T[i][j] représente le coût de plantation de l'essence j à l'emplacement i.
- g[i-1][k] représente le coût minimal obtenu pour les plantations aux emplacements précédents.
- min(g[i-1][k]) pour tout k ≠ j permet de prendre en compte toutes les possibilités d'essences aux emplacements précédents, à l'exception de l'essence j pour l'emplacement i. On cherche donc à minimiser le coût total jusqu'à l'emplacement i en choisissant l'essence la moins coûteuse parmi les essences différentes de j.

Avec cette formule de récurrence, nous pouvons calculer les coûts minimaux g[i][j] pour tous les emplacements i et essences j.

La solution au problème consiste donc à calculer les valeurs de g[n][j] pour toutes les essences j et à choisir la possibilité de chemin avec le coût total minimal, permettant ainsi de minimiser le coût total de la plantation d'arbres.

L'algorithme récursif naïf pour la plantation d'arbres a une complexité exponentielle, car il explore toutes les combinaisons possibles d'essences pour chaque emplacement.

La complexité de l'algorithme dépend du nombre d'emplacements (n) et du nombre d'essences (m). À chaque emplacement, l'algorithme explore m-1 essences différentes (en excluant l'essence déjà utilisée à l'emplacement précédent). Par conséquent, le nombre total de combinaisons d'essences est (m-1)^(n-1).

La récursion se produit n fois, car nous devons décider de l'essence pour chaque emplacement.

Cela donne une complexité de : O((m-1)^(n-1) * n).

III – Approche Top Down

Estimer la complexité de cet algorithme.

L'algorithme Top Down a une complexité exponentielle similaire à l'approche récursive naïve. Cependant, la mémorisation des résultats intermédiaires permet d'éviter de recalculer les mêmes sous-problèmes plusieurs fois, ce qui réduit donc le nombre d'appels récursifs.

La complexité de cet algorithme dépend du nombre d'emplacements (n) et du nombre d'essences (m). L'algorithme explore m-1 essences différentes (en excluant l'essence déjà utilisée à l'emplacement précédent), mais grâce à la mémorisation, les résultats intermédiaires sont stockés pour éviter les recalculs inutiles.

La récursion se produit n fois, car nous devons décider de l'essence minimale pour chaque emplacement.

Cela donne donc une complexité de : O((m-1)*n).

IIII – Approche Bottom Up

Estimer la complexité de cet algorithme.

L'algorithme utilise trois boucles imbriquées pour parcourir les emplacements, les essences et les essences précédentes, respectivement. Chaque boucle parcourt au maximum n, m et m itérations.

À chaque itération de la boucle la plus interne, un calcul de coût est effectué et comparé avec le coût optimal actuellement enregistré. Cela donne une complexité de O(1) pour chaque itération de la boucle la plus interne.

Ainsi, dans le pire des cas, l'algorithme effectue n * m * m opérations, ce qui donne une complexité de O(n * m^2).

La recherche du coût minimal dans la dernière ligne ajoute une complexité de O(m) à l'algorithme.

La reconstruction de la liste des essences utilisées se fait en parcourant les emplacements de la dernière ligne vers la première ligne, ce qui nécessite une itération de taille n. Cela donne une complexité de O(n) pour cette étape.

Donc la complexité totale de l'algorithme est de O(n * m^2 + m + n), ce qui peut être simplifié en :

O(n * m^2) dans le pire des cas, puisque m est généralement inférieur à n.

La complexité de cet algorithme est de O(n * m^2), où n est le nombre d'emplacements et m est le nombre d'essences.