2 alogo

2.

la stratégie gloutonne ne garantit pas toujours la solution optimale. Un contre-exemple simple est fourni par l'exemple 2 du sujet : T=[3,9,2,7,3,1], C=[2,2,5,4,2,1], A=2, B=−5. Appliquer la stratégie gloutonne à cet exemple en commençant par visiter l'emplacement 1 pour maximiser le gain immédiat ne conduit pas à la solution optimale, qui est de visiter les emplacements 0, 1 et 4.

3.1

$$V[i] = max(A \times T[i] + V[j],\ B \times T[i] + V[i+1])$$

où $V[i]$ représente la somme maximale que l'on peut collecter entre les emplacements d'indices i et n−1 (inclus), Aest le coefficient lorsque l'emplacement possède le même symbole que l'emplacement précédemment visité, B

est le coefficient lorsque l'emplacement est le premier visité ou qu'il possède un symbole différent de l'emplacement précédemment visité, $T[i]$ est la somme que l'on peut collecter en visitant l'emplacement d'indice i, et j est le plus petit indice supérieur à i tel que $C[j] \neq C[i]$ ou j=n si aucun tel indice n'existe.

3.2

L'algorithme récursif naïf explore toutes les possibilités de parcours à chaque emplacement, ce qui entraîne une complexité exponentielle. Pour chaque emplacement, il explore à nouveau toutes les options pour chaque emplacement précédemment visité, doublant ainsi le nombre de sous-problèmes à chaque niveau de la récursion. En conséquence, la complexité temporelle de cet algorithme est approximativement O(2^n), ce qui le rend inefficace pour des entrées de taille importante. Cette inefficacité découle de

sa capacité à examiner toutes les combinaisons possibles, ce qui devient prohibitif lorsque le nombre d'emplacements augmente.

4.1

Dans cette approche, on utilise la récursivité avec mémorisation pour éviter de recalculer les sous-problèmes déjà résolus. La complexité temporelle dépend du nombre de sous-problèmes distincts à résoudre, ce qui est généralement de l'ordre de O(n * m), où n est le nombre d'emplacements et m est le nombre de symboles possibles. Bien que cette approche soit plus efficace que l'approche récursive naïve, elle peut ne pas être aussi efficace que l'approche Bottom Up en termes de complexité temporelle.

4.2

Cette approche résout les sous-problèmes de manière itérative, en commençant par les plus simples et en construisant progressivement la solution complète. Elle évite la récursivité et la mémorisation, ce qui lui permet d'être efficace avec une complexité temporelle généralement de l'ordre de O(n * m), où n est le nombre d'emplacements et m est le nombre de symboles possibles.