Costa da Quinta, Joao Filipe

Algorithmique - Bonus 4

1.1.1)

Le coût optimal pour T [i] [j] est :

T[i] [j] = min (M [i] [j] , T [i - k] [j] + T [i] [i - k])

Avec k = 1, …, i – j , et i , j de 0,…..,n

1.1.2)

Nous devons remplir T de façon dynamique, dans cet exemple, on remplie depuis le haut, ver le bas, et de gauche à droite.

On dit dynamique car on utilise des valeurs calculées par la fonction précédemment pour faire les calculs actuels, d'où l'importance de l'ordre ainsi que de la formule.

1.2.1)

Ici n = len(M), avec M = av.gen_matrix(int) :

Dans mon implémentation je définis une 1ere boucle, qui parcourt chaque ligne (nb de lignes = n), dans cette boucle une 2eme boucle qui pour chaque ligne parcourt chaque colonne (nb de colonnes = n), finalement une 3eme boucle qui calcule (T [i - k] [j] + T [i] [i - k]), pour tout k in range (1 .. i – j).

(i – j) est au plus = n, donc n itérations de 3eme boucle, de n itérations 2eme boucle, de n itérations de 1ere boucle, ce qui donne n*n*n => O(n^3).

Je n'ai pas pris en compte le temps de calcul pour path dans mon approximation précédente, mais path est aussi constitué d'une boucle qui a au plus n-2 itérations, car n-2 est le nombre max d'étapes, donc O(n^3*(n-2)) si on prend en compte path, qu'on peut arrondir à O(n^4).

1.2.2)

Si on décide de résoudre ce problème avec une stratégie type backtracking, on aura de la peine à trouver des conditions implicites ainsi que explicites car on ne sait pas, avant de commencer, à quoi ressemble l'unique solution, ceci n'était pas le cas lors du problème des n-reines.

Sans ces fonctions définies, on va en fait se retrouver à parcourir toutes les branches possibles (dans le pire des cas), et ensuite, trouver le minimum parmi les réponses, ce qui correspond à 1 option au début, ensuite 2, ensuite 3, … jusqu'à n-1, ce qui se traduit en O((n-1)!), plus le calcul du min qui est O(n), ce qui nous donne une complexité totale de O((n-1)!)+O(n).

On peut accélérer la tache en faisant que les branches qui ont possibilité d'être meilleures (moins chères) que la dernière meilleure, imaginons qu'on a une branche qui coute 16, si on en fait une que lors du deuxième choix coute déjà 17, on peut l'abandonner.

De plus, vu que la solution est unique, backtracking qui est capable d'explorer toutes les solutions pour un problème donné, perds tout son intérêt.