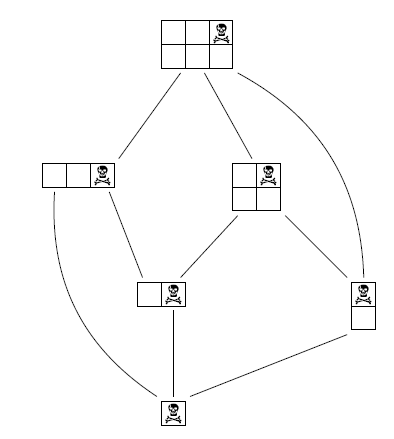
# TP3 ACT Programmation Dynamique

# Le jeu de la tablette de chocolat

Question 1 :



+3

+1

+1

0

+1

-2

Question 2 :

Si tous les fils sont > 0 :  ( max(fils) + 1) \* (-1)

Si au moins 1 fils <= 0 : | (max(fils <=0) -1) |

Question 3 :

|  |  |  |
| --- | --- | --- |
| Configuration | Temps exécution(ms) | Nombre d’appels récursifs |
| F (3, 2, 2, 0) = 3 | **0.090415** | **11** |
| F (10, 10, 7, 3) = 11 | **3030.154975** | **203237663** |

La version naïve a une complexité de l’ordre de l’exponentielle car nous avons beaucoup de cas similaires surtout quand on s’approche du cas où il reste 2 carés. Et que chaque configuration fait appel aux configurations qui ont déjà été utilisé par d’autres configurations et donc il y a énormément d’appels récursifs comme on peut le voir avec la valeur indiquée par le compteur ci dessus.

On voit que suivant la taille de la configuration même pour des petites valeurs par exemple dans l’exemple présenté ci-dessus, que l’écart entre les nombres d’appel récursif est énorme.

Question 4 :

F (100, 100, 50, 50) = -198

F (100, 100, 48, 52) = 191

Question 5 :

Paires i, j pour les configurations (127, 127, i, j) :

* [ 127, 127, 0, 63 ]
* [ 127, 127, 63, 0 ]
* [ 127, 127, 63, 126 ]
* [ 127, 127, 126, 63 ]

Question 6 :

Complexité de la version dynamique : O(m² \* n²)

On a un tableau à dimensions, les 2 premières dimensions représentent les différentes tailles de tablettes et les 2 dernières dimensions représentent les différentes positions de la tête de mort dans ces configurations ainsi nous avons toutes les combinaisons sauvegardés dans un tableau.

Nous allons remplir chaque case de ce tableau 1 et 1 seule fois.

La complexité sera donc : O (m² \* n²) puisque le tableau sera de taille [m][n][m-1][n-1]

Question 7 :

Les 8 configurations ont les mêmes valeurs car elles ont les mêmes tailles, et que peu importe où se trouve la case piège ça ne change pas le nombre de coups possibles et la valeur de la configuration.

Question 8 :

Exemple sur la configuration f (50, 50, 20, 20)

|  |  |  |  |
| --- | --- | --- | --- |
| Fonction | Valeur | Temps exécution(ms) | Nombre d’appels |
| Dynamique | -98 | 236.692482 | 396996 |
| Dynamique & symétrique | -98 | 76.397236 | 88437 |

On remarque que le temps d’exécution est nettement inférieur pour la fonction intégrant les cas de symétrie ceci est due au fait que la fonction intégrant les cas de symétrie fait nettement moins d’appels récursifs que la fonction dynamique.

Question 9 :

On voit que pour une configuration on peut donner une valeur à 8 configuration donc pour chaque appel récursif on va réduire le nombre d’appel par 8.

Ce qui fait que l’exécution est au moins 3 fois plus rapide (nous avons testé que sur des petites valeurs pour le jeu) mais pour des grandes configurations (de l’ordre de 50 \* 50) nous avons observé une exécution au moins 5 fois plus rapide.