Rapport TP2 ACT

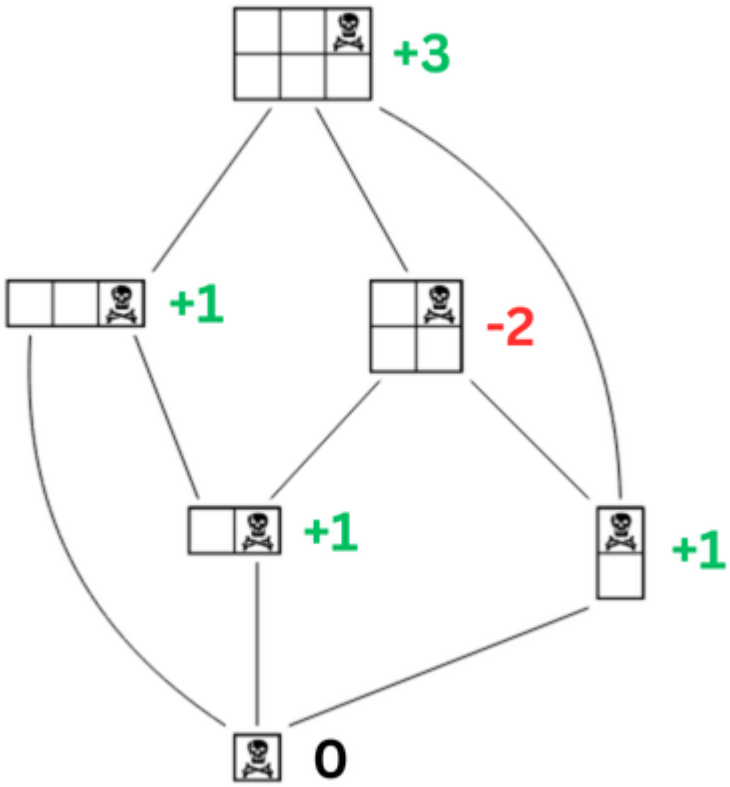
Gaspar Henniaux - Marwane Ouaret

Lien Github : https://github.com/pargass/ACT-TP/tree/main/tp2

Question 1

Pour définir une grille de manière unique il faut le nombre de lignes, le nombre de colonnes et les coordonnées de la case piégée

Question 2



à justifier...

Question 3

étant donnée donnée une configuration (m, n, i, j), pour obtenir tous ses successeurs on peut appliquer les règles suivantes :

à écrire... (voir algo)

Question 4

Pour calculer la valeur d'une configuration à partir des valeurs de ses successeurs, on peut appliquer la formule suivante :

- si parmis les successeurs il y a des valeurs négatives, alors la valeur de la position s'obtient en prenant la valeur absolue de la plus haute valeur négative et d'ajouter 1
- sinon, la valeur de la position s'obtient en prenant l'oppoée de la plus haute valeur positive (dans l'idée de retarder la défaite au maximum)

Question 5

```
function position_value(m, n, i, j):
    si il ne reste qu'une case alors
        return 0
    sinon
        valeur_successeurs = valeur de tous les successeurs de la position
(m, n, i, j)
        si il y a des valeurs négatives dans valeur_successeurs ou 0 alors
            return abs(valeur négative la plus haute) + 1
        sinon
            return -valeur positive la plus haute
```

Question 6

pour la configuration (10, 7, 7, 3), le temps d'exécution est de 192 secondes tandis que pour la configuration (10, 7, 5, 3) le temps d'exécution est de 414 secondes

Question 7

Cette différence s'explique par le fait que dans le second cas plein de positions sont recalculée plusieurs fois, ce qui n'est pas le cas dans le premier cas

la complexité de cet algorithme est exponentielle car dans la boucle principale on appelle la fonction position_value m + n - 2 fois

Question 8

Nous avons décidés de choisir un dictionnaire pour stocké les valeures déjà caculées. On met en clé les configurations sous forme de tuple et en valeur la valeur de la configuration.

Question 9

Pour la configuration (100,100,50,50), la valeur est -198 et la confirguration (100,100,48,52), la valeur est 191.

Question10

Les configurations possibles pour obtenir la valeur 127 dans une tablette de taille (127, 127) sont (127,127,i,j) tel que (i,j) :

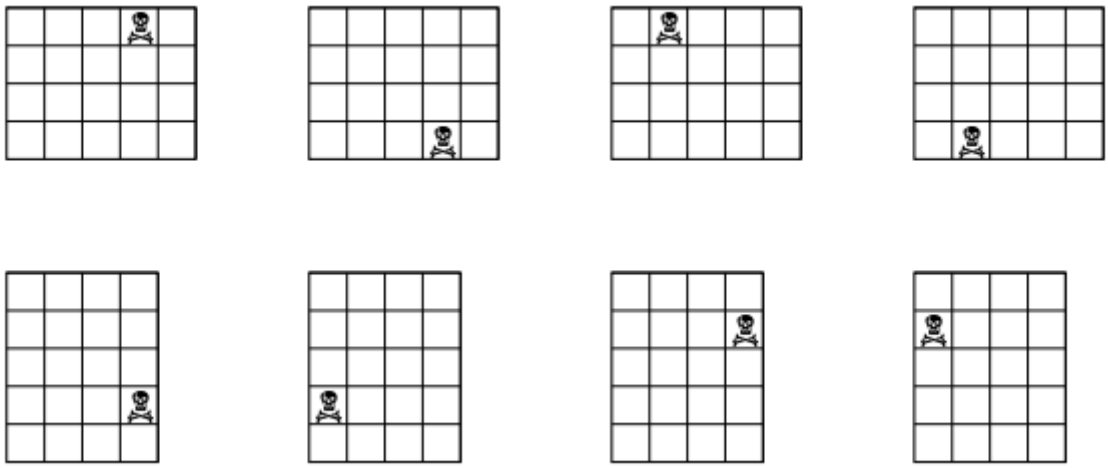
- **(0,63)**
- **(63,0)**
- **(126, 63)**

- **(63, 126)**

Question 11

To Do

Question 12



Toutes ces configurations ont la même valeur car il pour une configuration donnée, la valeur reste la même si on fait une rotation de 90°, 180° ou 270° ou si on fait un retournement horizontal ou vertical

Question 13

Question 14

Question 15

On considère que l'utilisateur rentre que des inputs valides, le code qui gère les mauvaises entrées est disponible sur : https://github.com/pargass/ACT-TP/tree/main/tp2

```
joueur = 1 -> ordinateur
joueur = 2 -> humain

function jeu (nb_colonne, nb_ligne, posX_mort, posY_mort, joueur){

    Si le nb_colonne et nb_ligne = 1
        afficher la configuration
        Si joueur est l'oridnateur (1)
            print "Fin du jeu vous avez perdu"
        sinon
            print "Fin du jeu vous avez gagner"

    Afficher la configuration

    Si joueur est l'oridnateur (1)
        print "Tour de l'ordinateur"
```

```
        configurations = tableau de tous les successeurs directs de la
configuration actuelle (fonction possible_configuration)
        choices = tableau vide

        Pour chaque element dans configurations
            mettre dans choices la valeur de la configuration de element
(fonction acc_position_value)

        Si dans choices la valeur minimal est inferieur ou égale à 0
            index = indice  de choices de la valeur la plus grande
inferieurs ou égal à 0
            jeu (*configuration[index], 2) #Donne à l'adversaire la
configuration qui permet de perdre en le moins de coup possible
        sinon
            index = indice de choices de la valeur la plus grande
            jeu (*configuration[index], 2) #Donne à l'adversaire la
configuration qui permet qu'elle gagne en le plus de coup possible

    Sinon
        print "Votre tour"
        Si m!=1 et n!=1
            direction = input que l'utilisateur choisi entre h et v
(honrizontal et vertical)
        sinon si m==1
            direction = h #car plus possible de couper verticalement
        sinon
            direction = v #car plus possible de couper horizontalement

        Si direction == h
            n2 = input de l'utilisateur qui choisit une valeur entre 1 et
nb_ligne -1
            SI n2 <= posY_mort
                print "L'évaluation de votre coup est :"
acc_position_value(nb_colonne, nb_ligne - n2, posX_mort, posY_mort - n2)
                jeu (nb_colonne, nb_ligne - n2, posX_mort, posY_mort - n2,
1) #Cette configuration permet de décalé la position relative de posY_mort
et donner la main à l'ordinateur
            sinon
                print "L'évaluation de votre coup est :"
acc_position_value(nb_colonne, n2, posX_mort, posY_mort)
                jeu (nb_colonne, n2, posX_mort, posY_mort) #Cette
configuration permet de donner la main à l'adversaire de la coupe choisit

        Si direction == v
            m2 = input de l'utilisateur qui choisit une valeur entre 1 et
nb_colonne -1
            SI m2 <= posX_mort
                print "L'évaluation de votre coup est :"
acc_position_value(nb_colonne - m2, nb_ligne, posX_mort - m2, posY_mort)
                jeu (nb_colonne - m2, nb_ligne, posX_mort - m2, posY_mort,
1) #Cette configuration permet de décalé la position relative de posX_mort
et donner la main à l'ordinateur
            sinon
                print "L'évaluation de votre coup est :"
```

```
acc_position_value(m2, nb_ligne, posX_mort, posY_mort)
                jeu (m2, nb_ligne, posX_mort, posY_mort) #Cette
configuration permet de donner la main à l'adversaire de la coupe choisit
}
```

Question 16

Nous pensons que les 2 jeux ont beaucoup de ressemblance, par exemple :

- La nature des coups ressemble à une tablette de chocolat de taille (m, 1) avec uniquement des coupes de 1 à 3 autorisés.
- dans les 2 jeu il existe des configurations perdantes et gagnantes. Donc on pourrait représentés les configurations sous forme de graphe comme la question 2.
- Il y aura forcement un gagnant et un perdant, si le joueur ayant la configuration gagnante joue parfaitement, il ne pourra jamais perdre (et inversement)

Il serait donc possible de programmer de la même façon le jeu de nim (version dynamique classique) car la configuration est le nombre de batonnêt, cependant nous pense que choisir un tableau à une dimension comme façon de stocké les valeurs des configurations est plus adapté pour le jeu de nim dûe au fait qu'il n'y a qu'une caracteristique pour une configuration.