# TP1

Auteurs : Karam ELNASORY · Erkin Tunc BOYA· Semih DOYNUK

Code Sur la repositoire Github : https://github.com/ErkinTunc/ARP-Recherche-Non-Inform-



## Exercice 1

## Q1:

- **état**: une position dans la situation (correspond à une situation dans laquelle le solveur peut se trouver)
- **espace des états**: toutes les positions possibles (ensemble de tous les états possibles)
- **action** : correspond à une prise de décision menée par le solveur face à une situation donnée (operation qui permet de choisir le route)
- **fonction de succession**: une fonction de succession nous permet de sauter à l'état suivant (élément de l'espace des états) grâce à l'action choisie.

  Elle permet de modifier la situation courante et d'accéder donc à un nouvel état.
- **graphe d'états**: c'est un graphe orienté où
  - les nœuds sont les états présents dans l'espace des états,
  - il existe un arc entre les états "A" et "B" s'il existe une action reliant A à B.

    Pour montrer tous les états et comment se déplacer entre ces états.

### Q2:

- **état du monde réel** : situations réelles avec beaucoup de détails
- **une description d'état** : représentation abstraite de l'état, on abstrait les détails inutile et qui influent pas la solution.

Cette distinction est utile car elle simplifie la complexité du problème et nous permet de le résoudre sans soucier de détails qui complique le calcule des solutions sans forcément avoir plus de valeur ajouter à la solution. Elle permet de faire le traitement. Enplus, si on rancrontre une problem simulaire on peut appliquer une solution qu'on a deja utilise avant facilement.

## Exercice 2

Question 1 :

Un problème sera défini par les cinq éléments suivant :

0. La description abstraite d'un état s : deux contenaire chaque un a un état soit rempli soit vide et un capacité en plus d'un seau

   Formellement :

(c3,c5) représentant la contenance en litre de la cruche 3L c3 et la cruche 5L c5,

c3 est compris entre 0 et 3 et c5 entre 0 et 5 entiers

1. Un état initial, noté S0=(c3,c5)

2. un ensemble d'actions A = {a0, a1, .. , a5}, on ne regarde que les changement, 0: reste le meme

   a0:remplir c3= (3-c3, 0)

   a1:remplir c5=(0, 5-c5)

   a2:vider c3=(-c3, 0)

   a3:vider c5=(c3,c5)→(c3,0) (0,-c 5)

   a4:transferer c3 à c5 = (-min(c3, 5-c5), min(c3, 5-c5))

   a5:transferer c5 à c3 = (min(c5, 3-c3), -min(c5, 3-c3))

3. Une fonction de successeur, qui définit l'état résultant de l'exécution d'une action dans un état succ(s, a.i) = s+1; i allant de 0 à 5 pour chaque action décrite au dessus et quelque soit les états s

4. Un ensemble d'états buts ou terminaux noté T inclut S : état T={(c3,c5) : c5=4}

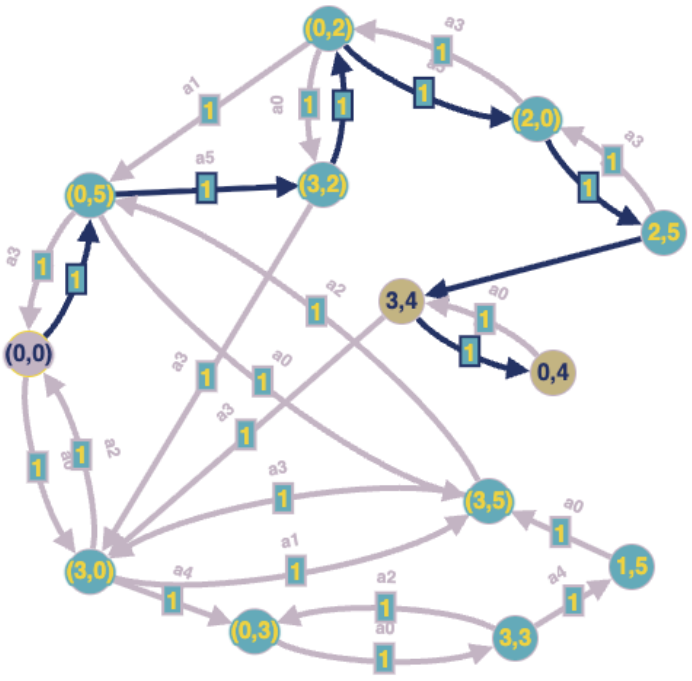
5. Une fonction de Coût, associant à chaque action un nombre non négative

c(s, s') = 1

Question 3.

On multiplie le nombre d'état de c3 et c5, 4x6

Question 4.



Question 5:

le chemin le plus court pour joindre l'étape (3,4) est en bleu sur le graphe, 7 étapes avec un cout de 6.

---

## Exercice 3 :

Problème fondé sur les déplacements du cavalier du jeu d'échecs : un cavalier partant d'une case quelconque (considérons ici la case au coin gauche supérieur) doit visiter chaque case du plateau sans y repasser.

Question 1 :

0. Un état S est la matrice 8*8 représentant l'échequier avec la case $M_{ij}$ = position du cavalier = -1, case visiter = 1, case non visiter = 0

1. $S_0$ : { Matrice M 8*8 | toute les case sont à 0 sauf la case gaucher superieur (0, 0) }
2. Direction = { HG, HD, DH, DB, BD, BG, GH, GB}

   Un ensemble d'actions A = { $a_{direction}$ | $\forall direction \in Direction$ }

   a(S, Direction) → S` $\forall direction \in Direction$ | direction est un déplacement (i , j)

   si $a_{ij}$ = -1 position_chevalier = (i, j) alors $a_{(i,j)+direction}$ = -2 et $a_{ij}$ = 1 .

   HG = (+2, +1)

   HD = (+2, -1)

   DH = (+1, +2)

   DB = (-1, +2)

   BD = (-2, +1)

   BG = (-2, -1)

   GH = (+1, -2)

   GB = (-1, -2)
3. succ(s, a) = a(S, direction)
4. Un états $T \in S$ tel que la matrice dans T avec ses cases $a_{ij}$ = 1 sauf une qui = -1.
5. Côut de l'action s se fait en 3 étapes

---

Question 2 :

Taille de la grille n*m cases, chaque action visite une case, on ne peut pas visiter des cases déjà visiter donc n*m états possible.

---

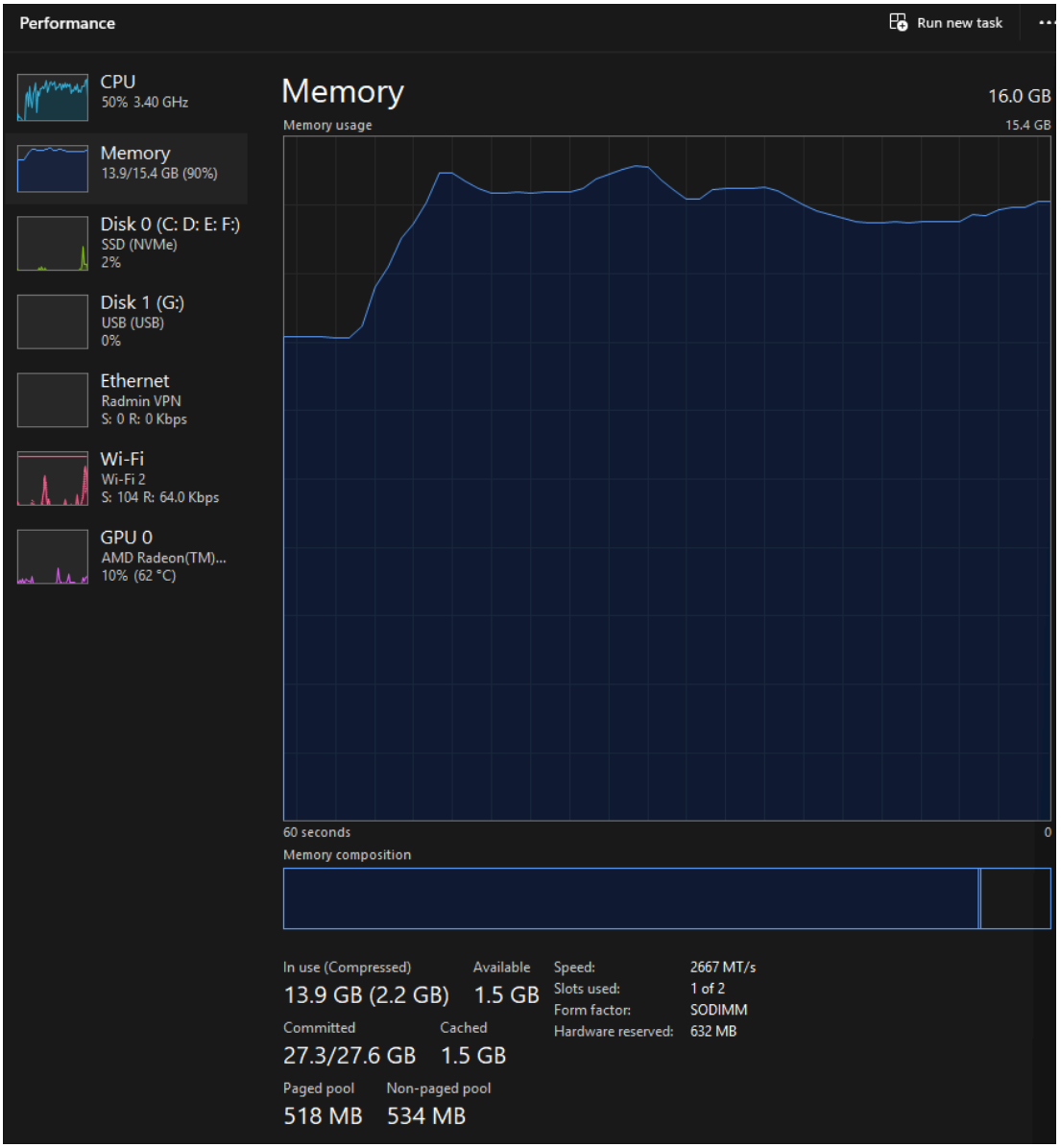
Question 3 : Code Sur la repositoire Github : https://github.com/ErkinTunc/ARP-Recherche-Non-Inform-

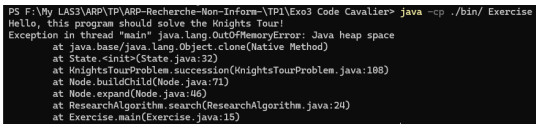
```
# Compiler
javac -d ./bin/ *.java
# Exécuter
java -cp ./bin/ Exercise
```

---

Question 4 :

Après plusieurs essayes, une matrice 10 * 10 j'épuise la mémoire de mon ordinateur sans parvenir à une solution.




Question 6 :

Code modifié avec la `ligne 73` pour un tour ouvert et `ligne 72` pour un tour ouvert

## Exercice 4

- 3 missionaire
- 3 cannibales
- Un bateau → peut transporte 2 personne

> Trouvez un moyen d'amener tout le monde de l'autrecôté sans jamais laisser un groupe de missionnaires dans un endroit en infériorité numérique par rapport aux cannibales de cet endroit

Ce problème est célèbre en IA car il a fait l'objet du premier article abordant la formulation du problème (Amarel, 1968).

---

# Q1: Problematique

1. On a l'état $(c, m, b)$
   - $c$ : **nombre de cannibals** a la rive gauche,
   - $m$ : **nombre de missionnaires** sur la rive gauche
   - $b$ : **position de bateau** : $b \in \{G, D\}$

   donc

   $$c, m \in \{0, 1, 2, 3\}\,, \quad b \in \{G, D\}\,,$$

   ---

   Contriante de securite:
   - pour rive gauche
     - si $m > 0$ alors $m \geq c$
   - Pour rive droite
     - Numero de cannibale en droite: $3 - c$
     - Numero de missionaire en droite: $3 - m$
     - si $3 - m > 0$ alors $3 - m \geq 3 - c$

   Quand il y a au moins 1 missionaire dans un cote , le numero de cannibale ne peut pas etre plus grand de Numero de missionaire.

   ---

2. Etat-Initial - Etat-But

   état initial : $s_0 = (3, 3, G)$

   état but: $s_f = (0, 0, D)$

   |→ Tous le monde est dans le cote droite

3. Actions (operateurs)

   |→ Un action montre combien des gens on peut choisir pour les monter sur le bateau.

   ---

   - $< c, m, g >$

     ---

     - $c, m \in \{-2, -1, 0, 1, 2\}$

       En general
       - soit 1 personne (seulement 1 $M$ ou 1 $C$)
       - soit 2 personnes (2 $M$ , 2 $C$ , ou $1M + 1C$ )

       ---

     - $g = G$ si bateau est en gauche de rive

       $g = D$ si bateau est en droite de rive

       ---

   - Bateau rive Gauche $G$ et les personne il peut ramener(envoyer au rive droite).

**a1:** envoie 1 *cannibale*: $(1,0)$ , $(c,m,G) \to (c-1,m,D)$

**a2:** envoie 2 *cannibales*: $(2,0)$, $(c,m,G) \to (c-2,m,D)$

**a3:** envoie 1 *missionaire*: $(0,1)$, $(c,m,G) \to (c,m-1,D)$

**a4:** envoie 2 *misionaire* $(0,2)$, $(c,m,G) \to (c,m-2,D)$

**a5**: envoie 1 *missionaire* + 1 *cannibale* $(1,1)$,

$$(c,m,G) \to (c-1,m-1,D)$$

- Bateau rive Droite $D$ et les personne il peut ramener(envoyer au rive Gauche).
  - cannibale droite : $3-c$
  - missionaire droite: $3-m$

**a6:** envoie 1 *cannibale*: $(1,0)$ , $(c,m,D) \to (c+1,m,G)$

**a7:** envoie 2 *cannibales*: $(2,0)$, $(c,m,D) \to (c+2,m,G)$

**a8:** envoie 1 *missionaire*: $(0,1)$, $(c,m,D) \to (c,m+1,G)$

**a9:** envoie 2 *misionaire* $(0,2)$, $(c,m,D) \to (c,m+2,G)$

**a10**: envoie 1 *missionaire* + 1 *cannibale* $(1,1)$,

$$(c,m,D) \to (c+1,m+1,G)$$

Il faut avoir assez gens pour envoyer des gens  au d'autre cote

old version

---

4. Fonction de Successeeur

   Succ(s,mouv)=

   si B=G : s - mouv

   si B=D : s + mouv

   mouv = {(1,0), (0,1), (2,0), (0,2), (1,1)}

---

5. Ensemble d'états but $T \subseteq S$
   - $S$ : tous les transition possible
   - $T \subseteq S$ : sous-ensemble des tous les terminal/but.

   dans cette problem abstrait il y a une seul terminal. c'est le condition ou

   $$T = (0,0,D)$$

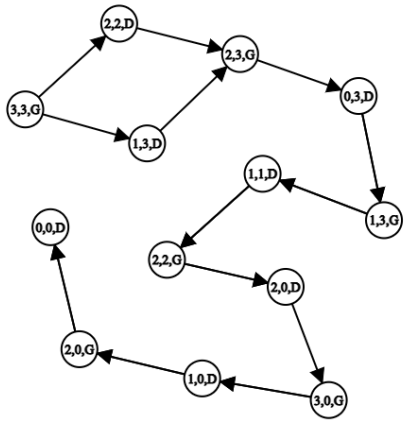
6. Fonction de coût

   On suppose que chaque voyage cout le meme valeur (qui est positive).

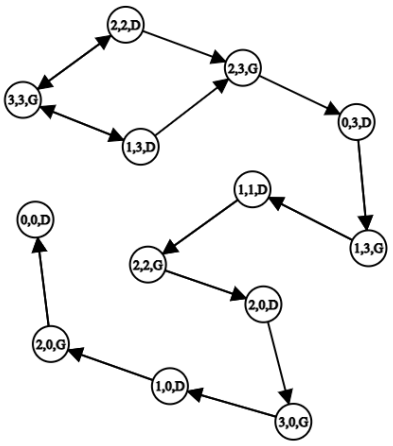
   - Ex:

$cout(s, a, s') = 1$ pour tous $(s, a, s')$

le cout total va nous donner combien de fois bateau faire une voyage au d'autre cote.




2 Chemin de Problematique




3,3,G peut etre acceder par 2,2,D et 1,3,D

Le problem ici c'est en chaque etat on peut retourner l'etat precedent et on peut faire une boucle combien de fois on veut. Donc il cree infini de resultat

ex:

$$1- > 2- > 3- > 2- > 3- > 2- > 3- > 2- > 3- > 4$$

Untitled

# Q2

Malgré le faible nombre d'état (10) les gens ont des difficultés avec cet énigme car il nécessite de l'anticipation afin de ne pas être bloqué, il y a aussi des séquences d'action requise pour atteindre la solution qui ne sont pas intuitifs.