Victor Maguer Benjamin Hollande Gabriel Guy–Soulard

S1.02
Comparaison d’approches algorithmiques

Table des matières

1 - Introduction

L’objectif de cette SAE est de finaliser la SAE S1.01 ou nous avons réalisé un menu nous permettant de jouer à certains jeux. Dans cette SAÉ nous devons réaliser plusieurs modes de jeux nous permettant de jouer contre l’ordinateur ou de faire s’affronter deux ordinateurs pour cela nous devons réaliser des stratégies de jeux adapter pour les différents jeux en prenant en compte la vitesse de l'algorithme, sa capacité à gagner et l’expérience de l’utilisateur.

2 - Approche et gestion du projet

1) Menu du choix du mode de jeu

Pour commencer la réalisation de ce projet nous avons créé un menu qui s'affiche après le choix du jeu qui nous permet de choisir le mode de jeu choisi par l'utilisateur.L'utilisateur pourra choisir de jouer en joueur contre un autre joueur, de jouer contre l'ordinateur ou bien de faire jouer deux ordinateurs ensemble.

```
print("-------------------------------------")
print("|                                   |")
print("| 1 - Jouer à 2 joueurs             |")
print("| 2 - Jouer contre l'ordinateur     |")
print("| 3 - Ordinateur contre ordinateur  |")
print("| 4 - Quitter                       |")
print("|                                   |")
print("-------------------------------------")
```

2) Gestion pour les allumette

Ensuite nous avons réalisé un sous menu pour le choix des difficultés pour le joueur contre l'ordinateur avec le deuxième choix pour le jeu des allumettes nous avons quatre difficultés disponible pour le joueur que nous allons traiter.

```
elif saisie == '2' :
    print("")
    print("")
    print("choisissez le niveau de difficulté :")
    print("1 - Facile")
    print("2 - Moyen")
    print("3 - Difficile")
    print("4 - Impossible")
    saisie=input("Faites votre choix :")
```

difficulté facile :

Pour ce niveau de difficulté l'ordinateur n'a pas de stratégie pour pouvoir gagner il ne fait que sélectionner aléatoirement une à trois allumettes et les enlèves. L'algorithme est

extrêmement rapide mais a peu de chances de gagner.

```
if nballu>=0:
    numjoueur = 2
if nballu>0 :# test si il reste des allumettes
    print("tour de l'ordinateur")
    choixIA=randint(1,3)
    while choixIA>nballu:
        choixIA=randint(1,3)
    print("l'ordinateur enlève",choixIA,"allumettes")
    allj2=allj2 + choixIA
    nballu=enlevéallumette(nballu ,choixIA)
    print(nballu*allu)#affichage des allumettes
    print("Il reste ",nballu," allumettes")
```

difficulté moyen :

Pour la deuxième difficulté une série de test permet de faire le choix optimal si il reste peu d'allumettes il fera le meilleur choix si il reste de 2 à 8 allumettes le seul cas ou il n'y aura pas de choix optimal pour gagner sera si il reste 5 allumettes pour ces cas le bot choisira un nombre aléatoire comme pour la difficulté facile. Le programme est encore une fois très rapide mais ne gagnera pas tout le temps car sa stratégie ne commence qu'à la fin ce qui fait qu'il n'est pas très dure à battre.

```
if nballu>0 :# test si il reste des allumettes
    print("tour de l'ordinateur")
    choixIA=randint(1,3)
    if nballu==2:
        choixIA=1
    if nballu==3:
        choixIA=2
    if nballu==4:
        choixIA=3
    if nballu==6:
        choixIA=1
    if nballu==7:
        choixIA=2
    if nballu==8:
        choixIA=3
    while choixIA>nballu:
        choixIA=randint(1,3)
    print("l'ordinateur enlève",choixIA,"allumettes")
    allj2=allj2 + choixIA
    nballu=enlevéallumette(nballu ,choixIA)
    print(nballu*allu)#affichage des allumettes
    print("Il reste ",nballu," allumettes")
```

difficulté difficile :

Pour cette difficulté l'approche est très différente on va utiliser un modulo 4 pour pouvoir toujours avoir le nombre optimal d'allumettes qui est le modulo 4 du nombre d'allumettes restante est égale à 1 donc sur 20 allumettes il faut qu'il reste a chaque tours du joueurs 17,13,9 et 5 allumettes pour pouvoir gagner. L'algorithme est encore une fois très rapide et elle est efficace : si le joueur fait une seule erreur il perd la partie. Mais le joueur peut gagner s' il connait la bonne stratégie.

```
if nballu>0 :# test si il reste des allumettes
    print("tour de l'ordinateur")
    if nballu%4==0:
        choixIA=3
    elif nballu%4==1:
        choixIA=randint(1,3)
        while nballu<choixIA:
            choixIA=randint(1,3)
    elif nballu%4==2:
        choixIA=1
    elif nballu%4==3:
        choixIA=2
    print("l'ordinateur enlève",choixIA,"allumettes")
    allj2=allj2 + choixIA
    nballu=enlèveallumette(nballu ,choixIA)
    print(nballu*allu)
    print("Il reste ",nballu," allumettes")
```

difficulté Impossible :

Sur cette difficultées le bot commence a jouer avant le joueur ce qui fait que le joueur ne pourra jamais gagner.

Victor Maguer Benjamin Hollande Gabriel Guy–Soulard

Ordinateur contre ordinateur :

Pour l'ordinateur contre ordinateur nous avons fait jouer deux ordinateurs de niveau facile l'un contre l'autre pour pouvoir garder un équilibre dans les victoires des deux ordinateurs. L'algorithme est rapide et comme tout est aléatoire, les deux ont approximativement les mêmes chances de gagner.

```
elif saisie == '3' :
        numjoueur=1
        pseudo1="Bot 1"
        numjoueur=2
        pseudo2="Bot 2"
        print(nballu*allu)
        print("Il reste ",nballu," allumettes")
        while nballu>0:
            numjoueur = 1
            print("tour de l'ordinateur1")
            choixIA=randint(1,3)
            while choixIA>nballu:
                choixIA=randint(1,3)
            sleep(1)
            print("l'ordinateur1 enlève ",choixIA," allumettes")
            allj2=allj2 + choixIA
            nballu=enlèveallumette(nballu ,choixIA)
            print(nballu*allu)#affichage des allumettes
            print("Il reste ",nballu," allumettes")
            if nballu>=0:
                numjoueur = 2
            sleep(1)
            if nballu>0 :# test si il reste des allumettes
                print("tour de l'ordinateur2")
                choixIA=randint(1,3)
                while choixIA>nballu:
                    choixIA=randint(1,3)
                print("l'ordinateur2 enlève",choixIA,"allumettes")
                allj2=allj2 + choixIA
                nballu=enlèveallumette(nballu ,choixIA)
                print(nballu*allu)#affichage des allumettes
                print("Il reste ",nballu," allumettes")
                if nballu>=0:
                    numjoueur = 1
```

3) Gestion pour les devinettes

Pour les devinettes il y aura donc le mode joueur contre ordinateur de disponible la différence avec les autres jeux est que les deux joueur n'ont pas le même rôle donc nous avons créer un menu avant le choix des difficulté qui nous permet de choisir si l'ordinateur doit deviner le nombre ou si il doit nous le faire deviner.

```
elif choix==2:
    print("")
    print("")
    print("---------------------------------")
    print("|                               |")
    print("| 1 - L'ordinateur devine       |")
    print("| 2 - L'ordinateur fait deviner |")
    print("| 3 - Quitter                   |")
    print("|                               |")
    print("---------------------------------")
    saisie=input("Faites votre choix :")
```

L'ordinateur devine :

Pour la difficulté moyenne, l'ordinateur prendra un nombre aléatoire entre 1 et le maximum choisis par le joueur et suivant la réponse du joueur va changer le maximum ou le minimum pour affiner la recherche. Cet algorithme peut prendre beaucoup de temps à trouver le nombre si la limite est grande.

```
min=0
max=lim+1
if choix==1:
    while trouvé==False :
        essai=randint(min+1,max-1)
        print(pseudo2," essayez de deviner le nombre :")
        print(essai)
        nbessai=nbessai+1 #compte le nombre d'essaie pour le calcul du score
        choix=reponse(pseudo1, n, essai)
        if choix==1 and n<essai :
            print("trop grand")
            max=essai
        elif choix==2 and n>essai :
            print("trop petit")
            min=essai
        elif choix==3 and n==essai :
            print("c'est gagné")
            trouvé=True
```

Il y a ensuite une difficulté difficile qui fonctionne par dichotomie donc l'ordinateur additionne le minimum et le maximum puis le divise par deux pour pouvoir toujours couper la liste en deux.Cette algorithme est le plus efficace pour pouvoir retrouver un élément rapidement dans un intervalle.

```
if choix==2:
    while trouvé==False :
        essai=int((min+max)/2)
        print(pseudo2," essayez de deviner le nombre :")
        print(essai)
        nbessai=nbessai+1 #compte le nombre d'essaie pour le calcul du score
        choix=reponse(pseudo1, n, essai)
        if choix==1 and n<essai :
            print("trop grand")
            max=essai
        elif choix==2 and n>essai :
            print("trop petit")
            min=essai
        elif choix==3 and n==essai :
            print("c'est gagné")
            trouvé=True
```

L'ordinateur fait deviner :

On pourra sélectionner trois niveaux de difficulté qui vont définir l'intervalle dans laquelle il faudra deviner le nombre et il répondra comme un joueur avec trop petit ou trop grand avec un test sur la saisie du joueur.

```
if choix==1:
    lim=randint(2,50)
if choix==2:
    lim=randint(50,500)
if choix==3:
    lim=randint(500,10000)
print("Limite :",lim)
n=randint(1,lim)
while trouvé==False : #test si le nombre saisie est égale au nombre cherché
    essai=int(saisiessai(pseudo2,lim))
    nbessai=nbessai+1 #compte le nombre d'essaie pour le calcul du score
    if n<essai :
        print("trop grand")
    elif n>essai :
        print("trop petit")
    elif n==essai :
        print("c'est gagné")
        trouvé=True
```

Victor Maguer Benjamin Hollande Gabriel Guy–Soulard

Ordinateur contre ordinateur :

Pour l'ordinateur contre ordinateur nous avons pris la difficulté moyenne de l'ordinateur qui devine et l'ordinateur qui fait deviner avec une limite qui est comprise entre 2 et 500. L'ordinateur qui devine fera une saisie aléatoire pour garantir l'équilibre des victoires entre les deux ordinateurs.

```
elif choix==3:
    pseudo1='Bot 2'
    pseudo2='Bot 1'
    lim=randint(2,500)
    n=randint(1,lim)
    print("la limite est ",lim)
    min=0
    max=lim+1
    while trouvé==False :
        essai=randint(min+1,max-1)
        print(pseudo2," essayez de deviner le nombre :")
        print(essai)
        nbessai=nbessai+1
        if n<essai :
            print("trop grand")
            max=essai
        elif n>essai :
            print("trop petit")
            min=essai
        elif n==essai :
            print("c'est gagné")
            trouvé=True
        sleep(1)
```

4) Gestion pour le morpion

Pour le morpion nous avons accès à un menu avec deux possibilité pour la difficulté pour joueur contre ordinateurs.

```
elif saisie == '2' :
    print("1 - Facile")
    print("2 - Moyen")
    choixdif = input("Choisissez votre difficulté !")
```

Difficulté facile :

La difficulté facile va permettre une sélection aléatoire des cases à remplir avec une fonction morpion bot.

```
elif tour%2==1 : #tour du deuxième joueur
    print(pseudo2," : choisissez le numéro de la ligne ")
    x = str(random.randint(0, 2))
    print(pseudo2," : choisissez le numéro de la colonne : ")
    y = str(random.randint(0, 2))
    val = morpionbot(x,y,val,tour,pseudo2)
```

La fonction morpionbot permet de vérifier si le nombre saisie et de renvoyer la variable val qui va être implémenté dans le morpion

Difficulté moyenne :

La difficulté moyenne repose sur une autre fonction verif_menace qui vise à étudier quelle case menace le plus l'autre joueur.

```
def verif_menace(val, signe1, signe2):
    """Fonction qui cherche selon un certain ordre de priorité quelle case menace le plus le joueur.
    Entrée : Le signe du joueur dont on doit analyser les coups
    Sortie : Un tuple d'entiers correspondant à la case la plus menaçante pour l'autre joueur."""
    for i in range(3):
        # Vérification des menaces sur les lignes
        if val[i].count(signe1) == 2 and val[i].count(' ') == 1:
            return i, val[i].index(' ')
        # Vérification des menaces sur les colonnes
        if [val[j][i] for j in range(3)].count(signe1) == 2 and [val[j][i] for j in range(3)].count(' ') == 1:
            return [val[j][i] for j in range(3)].index(' '), i
    # Vérification des menaces sur les diagonales
    if val[0][0] == val[1][1] == signe1 and val[2][2] == ' ':
        return 2, 2
    if val[0][2] == val[1][1] == signe1 and val[2][0] == ' ':
        return 2, 0
    if val[2][0] == val[1][1] == signe1 and val[0][2] == ' ':
        return 0, 2
    if val[2][2] == val[1][1] == signe1 and val[0][0] == ' ':
        return 0, 0
    return None
```

Le programme repose également sur la fonction morpionbot qui vérifie si la saisie colonne ou la ligne et valide ou déjà occupé est implémente la liste avec le choix du joueur.

```python
def morpionbot(x : str, y : str,val : list[list[str]],tour : int,pseudo : str )->list[list[str]]:
    """
    Test si la saisie colonne ou la ligne et valide ou déjà occupé est implémente la liste avec le choix du joueur .
    entrée :
        x(str) : indice de la ligne du morpion entre 0 et 2
        y(str) : indice de la colonne du morpion entre 0 et 2
    sortie : la liste modifier avec un rond ou un carré a l'indice choisit
    """
    while not est1nombre(str(x)) or not est1nombre(str(y)): #validité du choix du joueur
        x=str(random.randint(0, 2))
        y=str(random.randint(0, 2))
    while int(x)<0 or int(x)>2 or int(y)<0 or int(y)>2 or val[int(x)][int(y)]!=' ':
        x=str(random.randint(0, 2))
        y=str(random.randint(0, 2))

    if tour%2==0 : #implémentation de la valeur dans la liste
        val[int(x)][int(y)]='X'
    else:
        val[int(x)][int(y)]='O'
    return val
```

Voici la fonction bot moyen.

```python
while not partiegagnee and not testegalite(val):
    if tour % 2 == 0:  # Tour du joueur
        print(pseudo1, " : choisissez le numéro de la ligne : ")
        x = input()
        print(pseudo1, " : choisissez le numéro de la colonne : ")
        y = input()
        val = morpion(x, y, val, tour, pseudo1)

    elif tour % 2 == 1:  # Tour du bot moyen
        pos_menace = verif_menace(val, 'X', 'O')
        if pos_menace is not None:
            x, y = pos_menace
        else:
            x, y = str(random.randint(0, 2)), str(random.randint(0, 2))
        val = morpionbot(x, y, val, tour, pseudo2)
```

Comme on peut le voir, elle s’appuie sur les fonctions citées précédemment pour faire en sorte que le bot joue selon la menace qui risque le plus de faire gagner le joueur humain.

Ordinateur contre ordinateur

Le mode ordinateur contre ordinateur repose essentiellement sur le hasard avec la fonction randint() du module random.

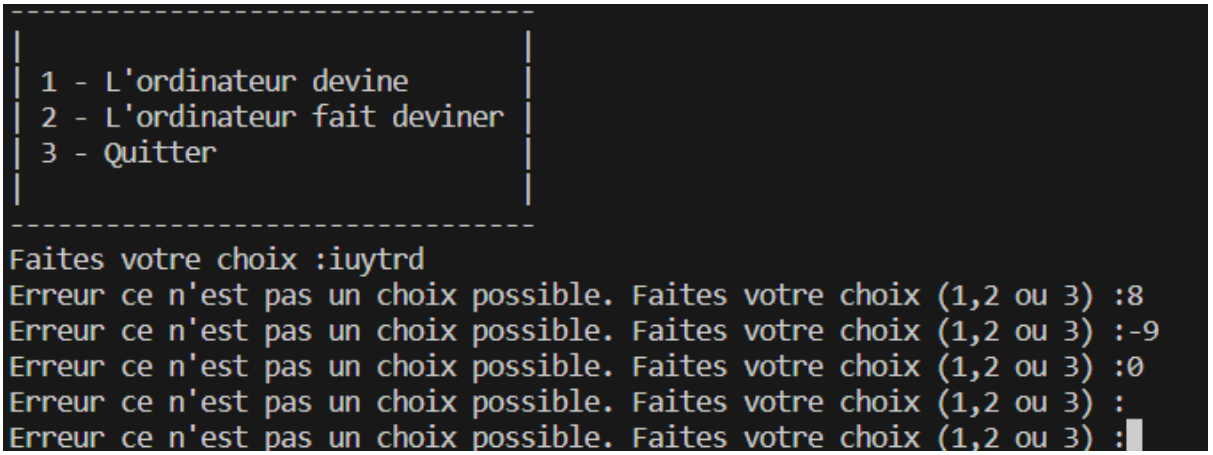
```
while partiegagnée==False and not testegalite(val):#test si la partie et terminer
    if tour%2==0 : #tour du premier joueur
        print(pseudo1," : choisissez le numéro de la ligne : ")
        x = str(random.randint(0, 2))
        print(pseudo1," : choisissez le numéro de la colonne : ")
        y = str(random.randint(0, 2))
        val = morpionbot(x,y,val,tour,pseudo1)
        sleep(1)
        print("    0   1   2  ")
        print("  -------------")
        print("0 |",val[0][0],"|",val[0][1],"|",val[0][2],"|")
        print("  -------------")
        print("1 |",val[1][0],"|",val[1][1],"|",val[1][2],"|")
        print("  -------------")
        print("2 |",val[2][0],"|",val[2][1],"|",val[2][2],"|")
        print("  -------------")
        tour=tour+1
        partiegagnée=test(val)
    elif tour%2==1 : #tour du deuxième joueur
        print(pseudo2," : choisissez le numéro de la ligne ")
        x = str(random.randint(0, 2))
        print(pseudo2," : choisissez le numéro de la colonne : ")
        y = str(random.randint(0, 2))
        val = morpionbot(x,y,val,tour,pseudo2)
```

5) Jeux d’essai

Comme le code des jeux est le même que celui de la SAE 1.01, nous n’allons pas refaire les tests.

Pour les menus, on peut voir que si l’on rentre un nombre autre que 1,2,3 et 4 , rien, 0 , un négatif où des caractères quelconques le programme redemande une saisie.

```
| 1 - Jouer à 2 joueurs                |
| 2 - Jouer contre l'ordinateur        |
| 3 - Ordinateur contre ordinateur     |
| 4 - Quitter                          |
|                                      |
----------------------------------------
Faites votre choix :kojhgf
Erreur ce n'est pas un choix valide. Faites votre choix :12
Erreur ce n'est pas un choix valide. Faites votre choix :
Erreur ce n'est pas un choix valide. Faites votre choix :0
Erreur ce n'est pas un choix valide. Faites votre choix :-998
Erreur ce n'est pas un choix valide. Faites votre choix :
```



3 - Conclusion :

Nous avons réussi à approcher de plusieurs manières la façon de réaliser des algorithmes nous permettant d'offrir une expérience variée aux joueurs avec plusieurs stratégies à comprendre pour pouvoir battre l'ordinateur.