La traversée de la rivière

Fiche professeur

Quand on joue à un jeu, on souhaite maximiser sa probabilité de gagner. Si cette probabilité est trop difficile à calculer, on peut obtenir une estimation, en moyennant les résultats d'un grand nombre de parties simulées. Certains programmes d'intelligence artificielle fonctionnent selon ce principe. C'est sur ce principe que l'ordinateur Deep Blue a été programmé et a réussi à gagner face au champion d'échecs Kasparov en 1997. En effet, dans le cas du jeu d'échec, à chaque coup, le calcul exact de la probabilité de gagner est inaccessible, car il y a trop de possibilités à envisager.

- Niveau : 1ère générale ou technologique
- Durée : 1 heure ou 2 heures
- Matériel à prévoir : des dés et des pions

1. Description de l'activité

1.1 La situation problème

Les élèves découvrent un jeu simple possédant plusieurs stratégies de jeu. Ils expérimentent les différentes stratégies en jouant. Ils doivent ensuite calculer les probabilités de réussite de chacune des stratégies : la valeur exacte en calculant à l'aide d'un arbre ou la valeur approchée en écrivant un programme qui simule un grand nombre de parties de jeu.

1.2 Organisation pédagogique

Compétences travaillées :

- Représenter une situation par un arbre pondéré de probabilités
- Arbres pondérés et calcul de probabilités : règle du produit, de la somme
- Comprendre et utiliser une simulation numérique
- Concevoir et écrire une instruction conditionnelle
- Concevoir et écrire une boucle for

Pré-requis :

- Les élèves doivent avoir travaillé sur les calculs de probabilités avec les arbres pondérées
- Les élèves doivent être à l'aise avec l'écriture d'un programme Python, notamment les instructions conditionnelles et les boucles for.

Consignes et réalisations attendues

- Partie A : Travail à faire à la maison
- Partie B : Les élèves jouent en groupe de 4 avec une stratégie différente pour chaque élève du groupe
- Partie C : Les élèves calculent la probabilité de gagner pour chaque stratégie
- Partie D : Bilan avec le professeur

Consignes pour la partie programmation :
Les élèves peuvent programmer en local avec un éditeur de Python de type Thonny ou Edupython, ou en ligne en utilisant les liens Capytale ci-dessous :
Lien Capytale - Stratégie des sauts moyens (fiche énoncé)
Lien Capytale - Stratégie des sauts moyens (corrigé)
Lien Capytale - Stratégie des petits sauts (fiche énoncé)
Lien Capytale - Stratégie des petits sauts (corrigé)

Les aides possibles à apporter

- Fournir des arbres de probabilités à compléter
- Pour aider les élèves à programmer, on peut leur proposer de lire le code des fonctions Python pour petit saut et grand saut. Dans les activités en ligne Capytale, on propose aux élèves de commencer par lire et exécuter la fonction `sauter()` qui simule un petit saut.

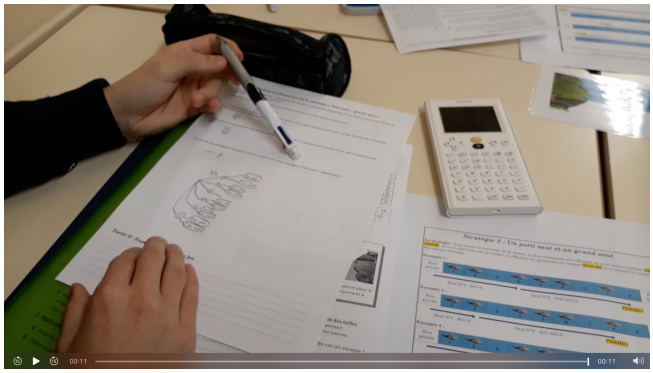
Différenciation possible

- Programmer la stratégie « trois petits sauts »est plus difficile.

Prolongements possibles

- Inventer une nouvelle stratégie avec une probabilité de gagner encore plus grande. Par exemple, on peut estimer que la stratégie « saut moyen + grand saut »aura une probabilité de gagner de 0,67.
- Proposer aux élèves de retirer un des rochers et de refaire les calculs.

Analyse du dispositif Les élèves ont apprécié le jeu. Les aides à la programmation étaient les bienvenues, et nous conseillons l'utilisation des pages Capytale pour la partie programmation.

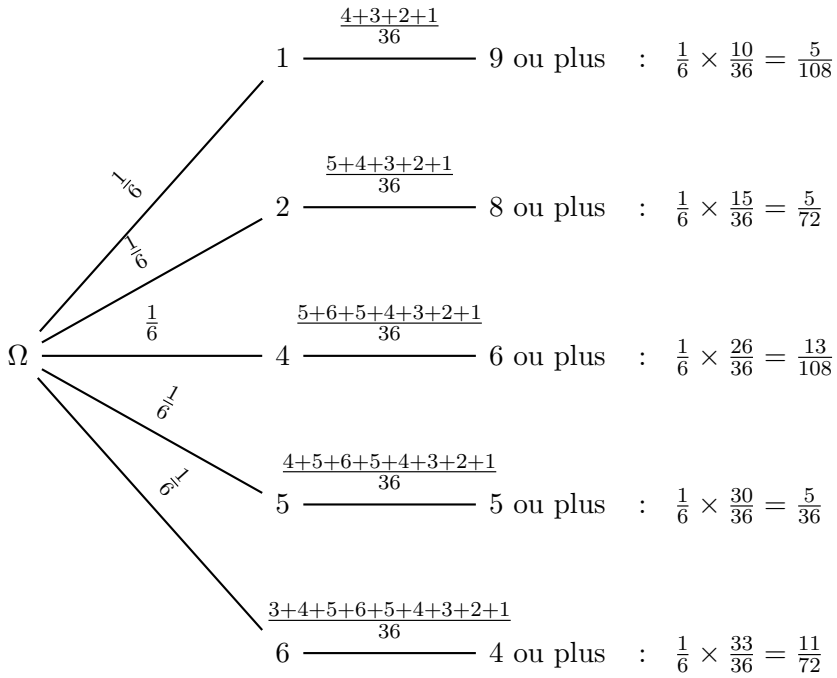


2. Correction de l'activité

2.1 Partie A : Loi de probabilité de la somme de deux dés

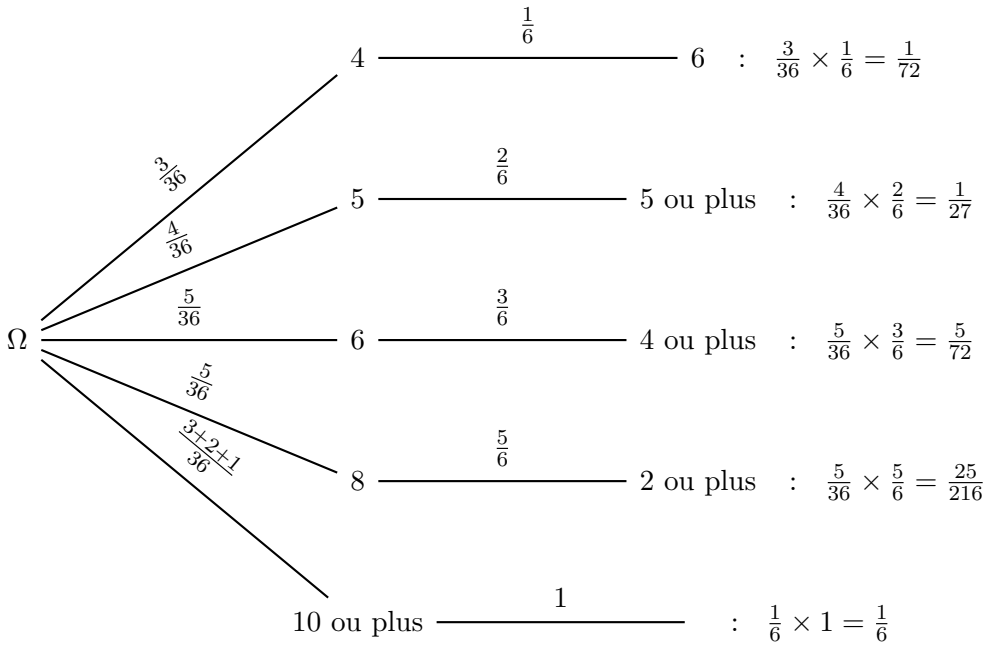
Issues	2	3	4	5	6	7	8	9	10	11	12
Probas	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

2.2 Partie C : probabilité de gagner avec la stratégie « Petit saut et grand saut »



La probabilité de gagner est donc $\frac{5}{108} + \frac{5}{72} + \frac{13}{108} + \frac{5}{36} + \frac{11}{72} = \frac{19}{36} \approx 0,53$

2.3 Partie C : probabilité de gagner avec la stratégie « Grand saut et petit saut »



La probabilité de gagner est donc $\frac{1}{72} + \frac{1}{27} + \frac{5}{72} + \frac{5}{36} + \frac{25}{216} + \frac{1}{6} = \frac{29}{72} \approx 0,40$

2.4 Partie C : probabilité de gagner avec la stratégie « Deux sauts moyens »

```
from random import randint
def DeuxSautsMoyens():
    de1=randint(1,6)
    de2=randint(1,6)
    saut1=max(de1,de2)  #le plus grand des deux des
    if saut1==3:
        return "PERDU"
    else:
        de1=randint(1,6)   #de1
        de2=randint(1,6)   #de2
        saut2=max(de1,de2)
        if saut2+saut1>=10:
            return "GAGNE"
        else:
            return "PERDU"
```

```
N=10000
compteur=0
for _ in range(N):
    if DeuxSautsMoyens()=="GAGNE":
        compteur+=1
print("Proba est environ : ",compteur/N)
```

On estime la probabilité de gagner en considérant la fréquence obtenue en simulant un grand nombre de parties. On trouve que la probabilité de gagner est environ 0,43.

2.5 Partie C : probabilité de gagner avec la stratégie « Trois petits sauts »

```
from random import randint
def TroisPetitsSauts():
    de1=randint(1,6)
    position=de1
    if position==3:
        return "PERDU"
    else:
        de2=randint(1,6)
        position=de1+de2
        if position==3 or position==7 or position==9:
            return "PERDU"
        else:
            de3=randint(1,6)
            position = de1+de2+de3
            if position >= 10:
                return "GAGNE"
            else:
                return "PERDU"
```

```
N=10000
compteur=0
for _ in range(N):
    if TroisPetitsSauts()=="GAGNE":
        compteur+=1
print("Proba est environ : ",compteur/N)
```

On estime la probabilité de gagner en considérant la fréquence obtenue en simulant un grand nombre de parties. On trouve que la probabilité de gagner est environ 0,35.

2.6 Partie D : BILAN

Quand on joue à un jeu, pour maximiser ses chances de gagner, on doit calculer la probabilité de gagner et adapter sa stratégie en fonction de cette probabilité.

On a vu dans cette activité 4 stratégies conduisant à des probabilités de gagner différentes. Pour calculer ces probabilités, on a utilisé deux méthodes :

- Le calcul direct de la probabilité qui permet de trouver la valeur exacte de la probabilité de gagner.
- Le calcul algorithmique obtenu en moyennant les résultats d'un grand nombre de partie jouées, et qui permet d'obtenir une estimation de la probabilité de gagner. Certains programmes d'intelligence artificielle fonctionnent selon ce principe. C'est sur ce principe que l'ordinateur Deep Blue a été programmé et a réussi à gagner face au champion d'échecs Kasparov en 1997. En effet, dans le cas du jeu d'échec, le calcul exact de la probabilité de gagner est inaccessible, car il y a trop de possibilités à envisager.