

Ce projet se compose de deux parties : une partie **Probabilités** et une partie **Statistiques**. La partie **Statistiques** reprend des concepts introduits dans la partie **Probabilités**, mais les deux peuvent se traiter indépendamment.

Le but de ce projet est de programmer un assistant de jeu pour le poker, plus particulièrement le Texas Hold'em limit¹ (voir [Règles complètes Texas Hold'em](#) et [Règles poker limit](#)).

Vous rendrez une copie par binôme de 5 pages maximum contenant les réponses aux questions posées dans le sujet, ainsi que votre code en python au plus tard pour le :

Vendredi 01 novembre 2019


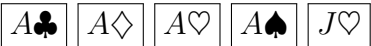

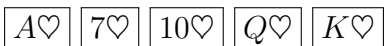
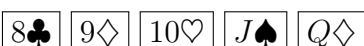
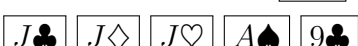

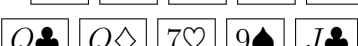
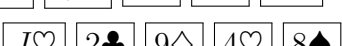
Résumé des règles

Le but du poker est de remporter les jetons des adversaires en constituant la meilleure combinaison de cinq cartes ou en leur faisant abandonner le coup.

Les séquences de jeu alternent distribution de cartes et tours d'enchères. À la fin du dernier tour d'enchères, tous les joueurs encore en jeu ont misé la même quantité de jetons et les cartes sont dévoilées : le gagnant de la manche est celui avec le meilleur jeu, il remporte les jetons.

Dans le cas où plus de 5 cartes peuvent constituer un jeu, on gardera le meilleur possible.

Les mains possibles sont, de la meilleure à la plus faible :

1. Quinte flush 
2. Carré 
3. Full 
4. Couleur 
5. Suite 
6. Brelan 
7. Deux paires 
8. Une paire 
9. Carte Haute 

Texas Hold'em

Dans le cas du Texas Hold'em, chaque joueur possède 2 cartes qui lui sont propres et 5 cartes communes sont placées sur la table (les joueurs allant jusqu'au dernier tour d'enchères devront donc constituer leur jeu parmi 7 cartes).

Les tours de jeu s'enchaînent de la manière suivante :

1. Distribution de deux cartes cachées à chaque joueur

1. on se restreint au 'limit' pour diminuer le nombre de coups possibles dans la partie **Statistiques**

2. 1^{er} tour d'enchères
3. Distribution de trois cartes communes (le *flop*)
4. 2^{ème} tour d'enchères
5. Distribution de la quatrième carte commune (le *turn*)
6. 3^{ème} tour d'enchères
7. Distribution de la cinquième carte commune (la *river*)
8. 4^{ème} tour d'enchères
9. Abattage des cartes des joueurs encore en jeu

Limit

Dans la variante *limit* du Tepad Hold'em, au moment du tour d'enchères, un joueur peut :

- se coucher (*fold*) : il abandonne la manche et perd les jetons misés précédemment ;
- suivre (*call*) : il mise la même somme que le joueur précédent ;
- faire "parole" ("*check*") : si aucun joueur avant lui n'a misé de jeton, il continue sans miser ;
- relancer (*raise*) : il augmente la mise (d'un montant prédéfini).

Le tour d'enchère se poursuit jusqu'à ce que tous les joueurs aient misé la même somme ou aient abandonné.

La différence entre la version *limit* et *no limit* vient du fait que les relances soient bornées ou non.

1 Probabilités

Dans cette partie, on cherche à construire un assistant qui, étant donné un état du jeu (cartes en main et cartes sur la table), nous donne notre probabilité de gagner, perdre ou faire égalité.

On se place dans le cas où :

- on joue contre un seul joueur
- il y a 3, 4 ou 5 cartes sur la table (pour une question de complexité)

Le but est d'avoir des informations au moment des 2^{ème}, 3^{ème} et 4^{ème} tours d'enchères.

Code Compléter dans le fichier fourni les méthodes `get_probabilities_after_river`, `get_probabilities_after_turn` et `get_probabilities_after_flop`.

Vous pouvez vous aider des tests fournis pour vérifier la correction de vos algorithmes.

Question Dans le cas où la main du joueur est :

10♠ J♥

et les cartes sur la table sont :

3♠ 3♦ 10♦

Calculer la probabilité exacte pour que le joueur ait un full après le *turn*, puis après la *river*.

Vous expliquerez et détaillerez votre calcul dans votre copie.

ATTENTION ! on rappelle que la main du joueur est la meilleure possible, aussi si après la 5^{ème} carte commune le joueur a possibilité d'avoir un carré ou une quinte flush, il n'a pas un full.

2 Statistiques

On considère que l'on possède l'assistant de la partie précédente et que l'on est capable d'évaluer un jeu (comme la probabilité qu'il gagne ou qu'il fasse égalité).

On suppose qu'il existe 4 types de joueurs²

- agressif large : le joueur relance souvent, même avec des jeux faibles ;
- passif large : le joueur suit souvent sans relancer, même avec des jeux faibles ;
- agressif serré : le joueur ne joue presque qu'avec des bons jeux et n'hésite pas à relancer ;
- passif serré : le joueur ne joue presque qu'avec des bons jeux et se contente de suivre.

Selon le type de joueur contre lequel on joue, on souhaite pouvoir affiner notre stratégie.

Le but de ce projet n'est pas de déterminer cette stratégie mais de déterminer à partir des coups qu'il a joués, dans quelle catégorie peut-on placer un joueur.

On considère que les différents profils de joueurs ont les comportements suivants.

TABLE 1 – Actions moyennes du joueur agressif/large

		Valeur de la main			
		[0; 0.3]]0.3; 0.6]]0.6; 0.8]]0.8; 1]
Action	Fold	17%	14%	8%	2%
	Check	45%	32%	13%	4%
	Call	18%	26%	22%	10%
	Raise	20%	28%	57%	84%

TABLE 2 – Actions moyennes du joueur passif/large

		Valeur de la main			
		[0; 0.3]]0.3; 0.6]]0.6; 0.8]]0.8; 1]
Action	Fold	19%	15%	10%	2%
	Check	62%	39%	24%	22%
	Call	17%	40%	52%	55%
	Raise	2%	6%	14%	21%

2. Exemple de classification des [types de joueurs](#)

TABLE 3 – Actions moyennes du joueur agressif/serré

		Valeur de la main			
		[0; 0.3]]0.3; 0.6]]0.6; 0.8]]0.8; 1]
Action	Fold	37%	36%	11%	3%
	Check	45%	40%	13%	4%
	Call	14%	18%	11%	11%
	Raise	4%	6%	65%	82%

TABLE 4 – Actions moyennes du joueur passif/serré

		Valeur de la main			
		[0; 0.3]]0.3; 0.6]]0.6; 0.8]]0.8; 1]
Action	Fold	37%	34%	12%	4%
	Check	44%	50%	16%	3%
	Call	16%	12%	49%	62%
	Raise	3%	4%	23%	31%

Par exemple : Avec une main d'une valeur de 0.659, un joueur de type passif/serré :

- se couchera dans 12% des cas,
- dira "parole" dans 16% des cas,
- suivra dans 49% des cas,
- et relancera dans 23% des cas.

Code Compléter (ou recréer) le fichier fourni, le but étant d'avoir une fonction `get_best_profile`, qui à partir d'actions jouées par un joueur et d'une liste de profils, détermine quel est le profil le plus susceptible de correspondre au joueur.

Question Sur votre copie, présentez en détaillant (hypothèse nulle, hypothèse alternative, valeur du test, etc.) le test qui vous permet de déterminer si le joueur A est passif/large.

Question Sur votre copie, expliquez en détaillant votre programmation et concluez sur les profils des joueurs A,B,C,D,E et F.