Dossier : jeu de dominos

Nicolas Poulain

18 mars 2012

Table des matières

1 Règles du jeu de dominos

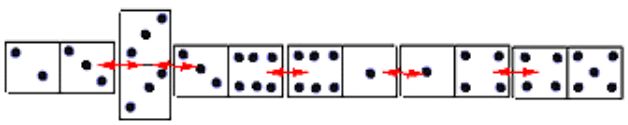
Le jeu de dominos est un jeu de société d'origine chinoise, utilisant 28 pièces (dans le cas d'un jeu "double-six"), les dominos. On peut adopter une des règles suivantes :

Règle 1

- Étaler les dominos sur la table, points cachés. Distribuer 10 dominos à chaque joueur.
- Celui qui a le double le plus fort commence et pose celui-ci, son voisin pose à l'une des extrémités (les bouts) un domino dont l'une des parties a le même nombre de points.
- Chaque joueur joue à son tour et l'on constitue ainsi une chaîne dont les parties voisines ont le même nombre de points.
- Le joueur qui ne peut pas jouer passe son tour, et on continue à jouer jusqu'à ce qu'un des joueurs se soit débarrassé de tous ses dominos, ou que le jeu soit complètement bloqué.
- À la fin du jeu, celui qui totalise le moins de points (la somme des points de l'ensemble des dominos) est le gagnant. On a donc tout intérêt à se débarrasser en premier des dominos valant beaucoup de points.

Règle 2

- Étaler les dominos sur la table, points cachés. Distribuer le même nombre de dominos à chaque joueur, en laissant un talon de quelques pièces.
- Le premier joueur pose son plus fort domino, son voisin pose à l'une des extrémités un domino dont l'une des parties a le même nombre de points.
- Chaque joueur joue à son tour et l'on constitue ainsi une chaîne dont les parties voisines ont le même nombre de points.
- Lorsqu'un joueur n'a pas de domino qui convienne, il pioche en prenant une pièce du talon et passe son tour, c'est le suivant qui joue.
- Le vainqueur est celui qui a placé le premier tous ses dominos.



2 Modélisation

représenté par un vecteur colonne $\begin{pmatrix} x \\ y \end{pmatrix}$ ou par un nombre $10x+y$ Ainsi le jeu d'un joueur peut être représenté par la matrice (le tableau)

$J = \begin{pmatrix} 6 & 5 & 5 & 3 & 3 & 3 & 1 \\ 1 & 4 & 0 & 3 & 1 & 0 & 1 \end{pmatrix}$ ou $J = (61, 54, 50, 33, 31, 30, 11)$.

La partie la plus intéressante du programme est l'élaboration de la stratégie de l'ordinateur.

2.1 Premières fonctions

On commencera par écrire les fonctions :

1. Écrire la fonction `creation_jeu` capable de donner l'ensemble des dominos de la boîte de jeu.

$$\begin{pmatrix} 6 & 6 & 6 & 6 & 6 & 6 & 6 & 5 & 5 & 5 & 5 & 5 & 5 & 4 & 4 & 4 & 4 & 4 & 3 & 3 & 3 & 3 & 2 & 2 & 2 & 1 & 1 & 0 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 & 5 & 4 & 3 & 2 & 1 & 0 & 4 & 3 & 2 & 1 & 0 & 3 & 2 & 1 & 0 & 2 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

2. Écrire une fonction `distribue` qui tire au hasard (et sans remise) une main contenant un nombre donné de dominos.
3. Écrire une fonction `est_avant` qui admet comme paramètres d'entrée deux dominos et qui renvoie `True` ou `False` selon que les deux dominos sont ou pas classés dans l'ordre décroissant. Par exemple `est_avant([6,2],[5,0])` renverrait `True`.
4. Écrire une fonction `tri_decr` qui trie un ensemble de dominos par ordre décroissant.
5. Écrire une fonction `doubles` qui renvoie en sortie l'ensemble des doubles d'une main ainsi que leur indice dans la main. Dans l'exemple du début, la fonction `doubles` renverrait en sortie $\begin{pmatrix} 3 & 1 \\ 3 & 1 \\ 4 & 7 \end{pmatrix}$ car le double 3 apparaît en 4-ème position dans la main et le double as en 7-ème.
6. Écrire une fonction `histogramme` qui calcule le nombre de fois où apparaît chacune des valeurs 0,1...dans un jeu. Dans notre exemple $H = [2, 4, 0, 4, 1, 2, 1]$.
7. Écrire une fonction `max_presence` qui renseigne sur le (ou les) nombre qui apparaît le plus souvent dans un jeu. Dans notre exemple `maxpresence` renverrait $[1, 3]$.
8. `possibilites` qui à partir d'une main donnée et des deux "bouts" se trouvant sur le tapis, donne les choix possibles du joueur. Dans notre exemple , `possibilites(J,5,0)` donnerait $\begin{pmatrix} 5 & 5 & 3 \\ 4 & 0 & 0 \\ 2 & 3 & 6 \end{pmatrix}$

2.2 Programme Python

```
#!/usr/bin/python
import random


def creation_jeu(max=6):
  """Cree la boite de jeu avec l'ensemble des dominos
  qui seront distribues
  """
  jeu=[]
  for i in range(max,-1,-1):
    for j in range(i,-1,-1):
      jeu = jeu + [[i,j]]
  return jeu

def distribue(jeu,n=10):
  """Tire dans jeu (sans remise) une main de n dominos"""
  main = []
  for i in range(n):
    r = random.randint(0,len(jeu)-1)
    main = main + [ jeu[r] ]
    jeu.pop(r)
  return main

def est_avant(d,e):
  """Verifie si deux dominos sont dans l'ordre lexicographique
  - Exemples :
  >>> est_avant( [2,3], [1,2] ); est_avant( [2,3], [2,4] )
  True
```

```
  False
  """
  if d[0]>e[0] or ( d[0]==e[0] and d[1]>e[1] ):
    return True
  return False

def tri_decr(player):
  """Trie les dominos du joueur dans l'odre
  decroissant lexocographique
  - Exemple :
  >>> tri_decr([ [1,2], [3,4], [1,3], [2,0] ])
  [[3, 4], [2, 0], [1, 3], [1, 2]]
  """
  player = sorted(player)
  player.reverse()
  return player

def is_player1_first(pl1,pl2):
  """Le joueur A a-t-il une meilleure main que le joueur B ?"""
  if est_avant(pl1[0],pl2[0]):
   return True
  return False

def possibilites(table,pl):
  """Donne, la liste des dominos de pl qui peuvent
  etre places sur la table
  >>> possibilites([[3,4],[4,4]], [[2,3],[1,5],[4,6]])
  [0, 2]
  """
  possbl = []
  x=table[0][0]
  y=table[-1][1]
  for i in range(len(pl)):
    if pl[i][0]==x or pl[i][1]==x or pl[i][0]==y or pl[i][1]==y:
      possbl = possbl + [ i ]
  return possbl

def un_tour_de_jeu(table,player,passe):
  """Tente de placer sur un des deux bouts de la table un
  domino de la maon du player. Si ce n'est pas spossible,
  on incremente passed_tours.
  """
  possbl = possibilites(table,player)
  if len(possbl)==0:
    return table,player,passe + 1
  table = positionne(player[ possbl[0] ], table)
  player.pop(possbl[0])
  return table,player,0

def positionne(domino,table):
  """Positionne le domino correctement sur la table"""
  if domino[0]==table[0][0]:
    table = [ [ domino[1],domino[0] ] ] + table
  elif domino[1]==table[0][0]:
    table = [ [ domino[0],domino[1] ] ] + table
  elif domino[0]==table[-1][1]:
    table = table + [ [ domino[0],domino[1] ] ]
  else:
    table = table + [ [ domino[1],domino[0] ] ]
  return table

if __name__ == "__main__":
  import doctest
  doctest.testmod()

  jeu = creation_jeu()
  player1 = tri_decr(distribue(jeu))
  player2 = tri_decr(distribue(jeu))
  print "Joueur␣1␣:␣",player1
  print "Joueur␣2␣:␣",player2
  print ""
  # initialisation de la partie
  if is_player1_first(player1,player2):
    print "Joueur1␣commence"
    table = [ player1[0] ]
    player1.pop(0)
```

```
    a_qui_le_tour = 2
  else:
    print "Joueur2 commence"
    table = [ player2[0] ]
    player2.pop(0)
    a_qui_le_tour = 1
  print "Table : ", table

  # c'est parti
  passed_tours = 0
  while passed_tours<2 and len(player1)>0 and len(player2)>0:
    p = passed_tours
    if a_qui_le_tour==1:
      table,player1,passed_tours = un_tour_de_jeu(table,player1,passed_tours)
      a_qui_le_tour = 2
      if p!=passed_tours:
        print "Joueur 1 passe"
    else:
      table,player2,passed_tours = un_tour_de_jeu(table,player2,passed_tours)
      a_qui_le_tour = 1
      if p!=passed_tours:
        print "Joueur 2 passe"
    if p==passed_tours:
      print "Table : ", table

  # Fin de partie
  print "Joueur 1 : ",player1
  print "Joueur 2 : ",player2
```

voir http ://code.google.com/p/npoulain pour le code