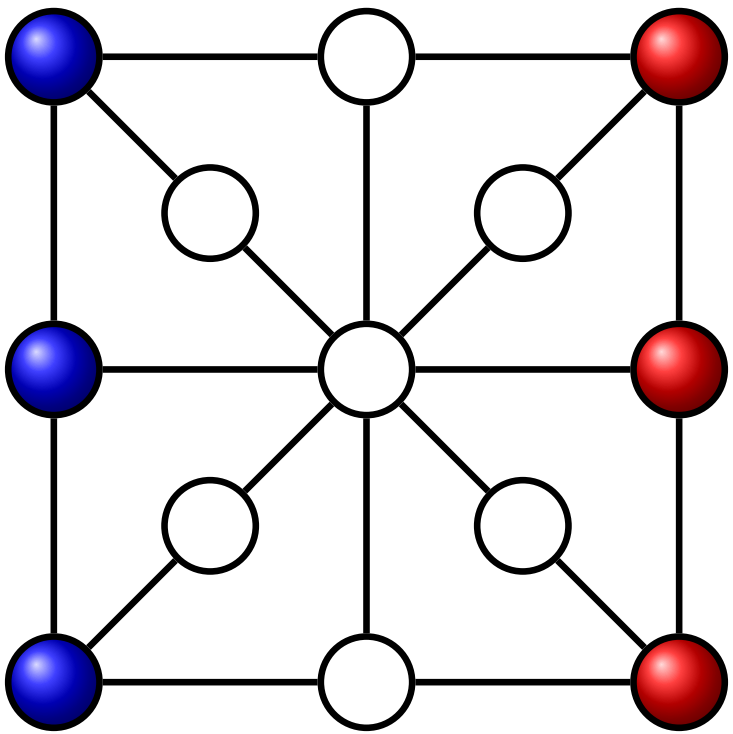
IA et Python
IREM

Alain Busser, Sébastien Hoarau

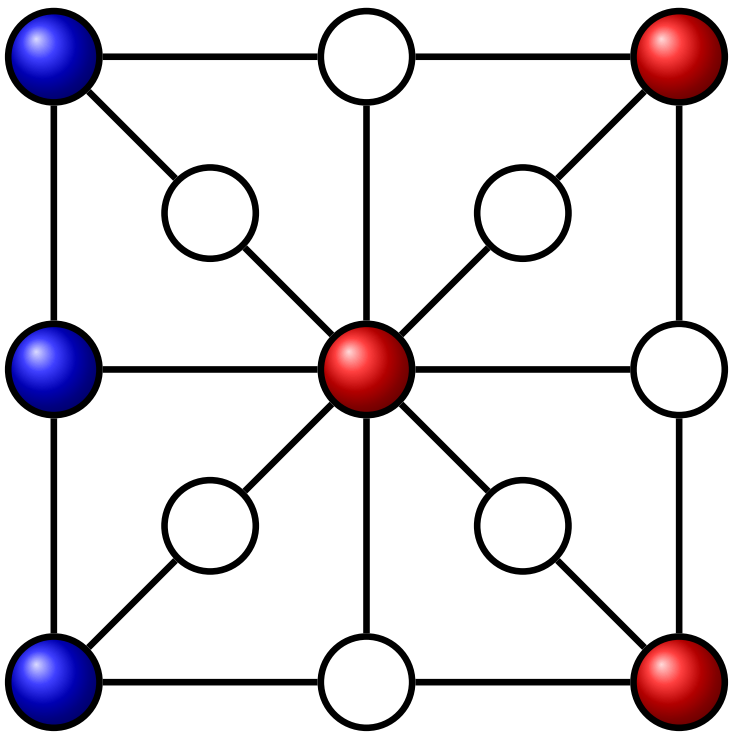
IREMI, LRG, LIM

31 mars 2021

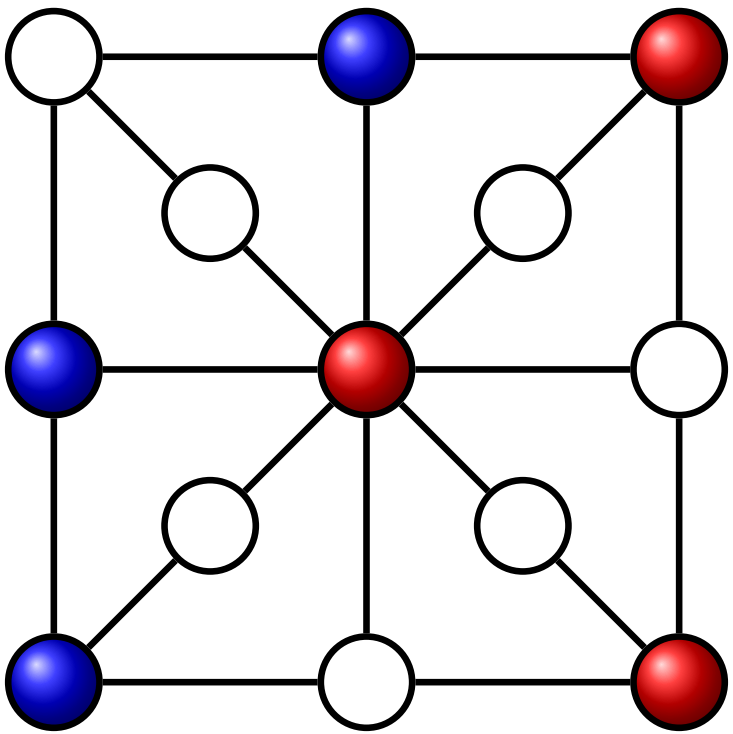
Il était une fois
une princesse qui dessinait des graphes



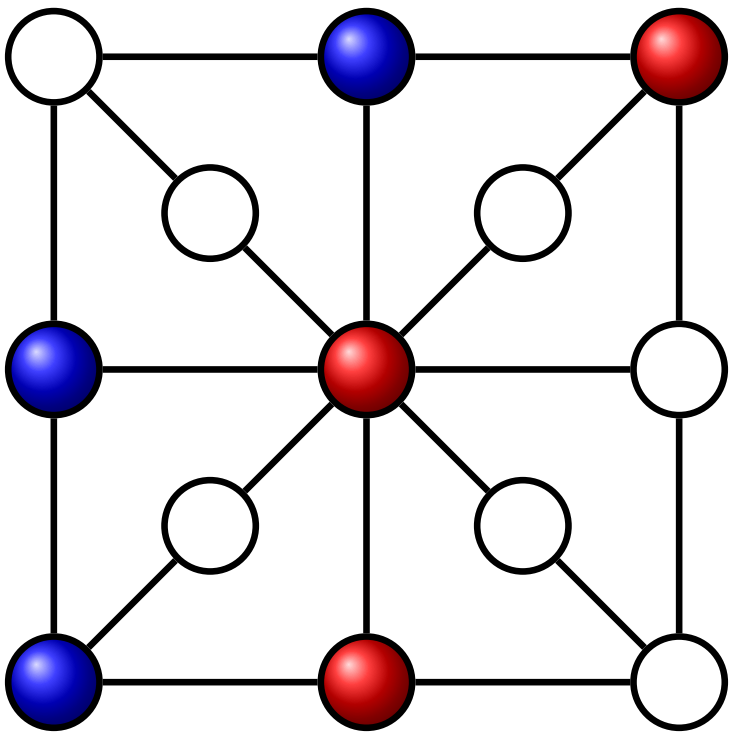
Il était une fois
une princesse qui dessinait des graphes



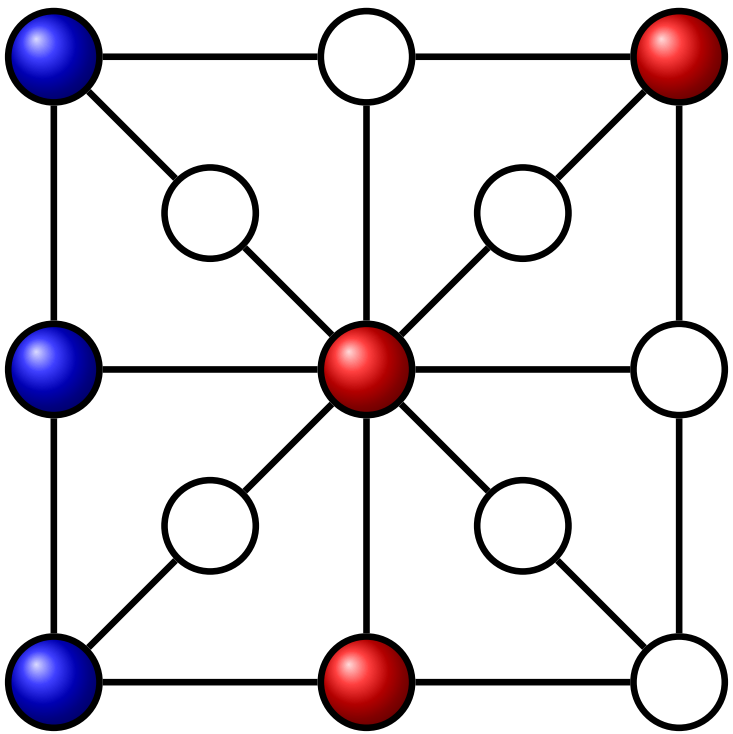
Il était une fois
une princesse qui dessinait des graphes



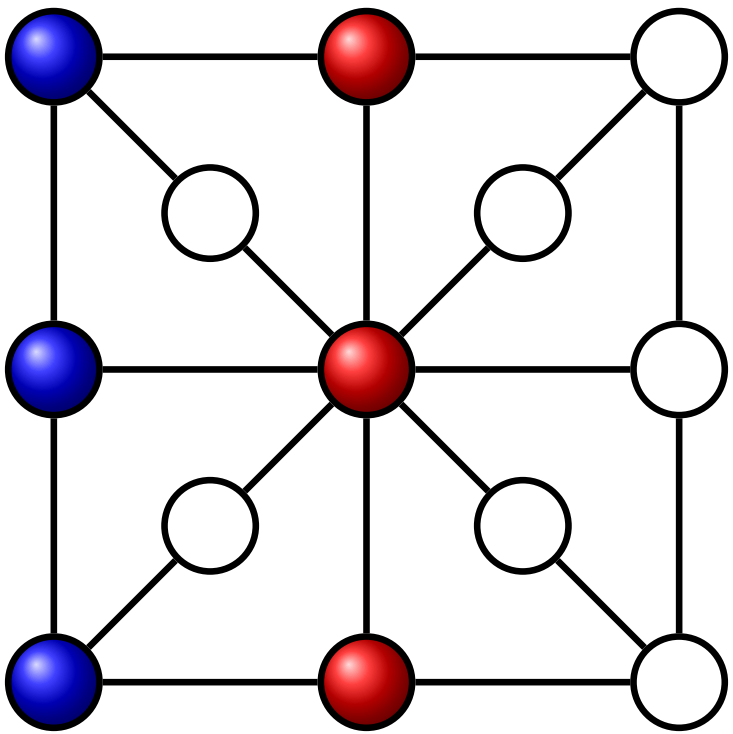
Il était une fois
une princesse qui dessinait des graphes



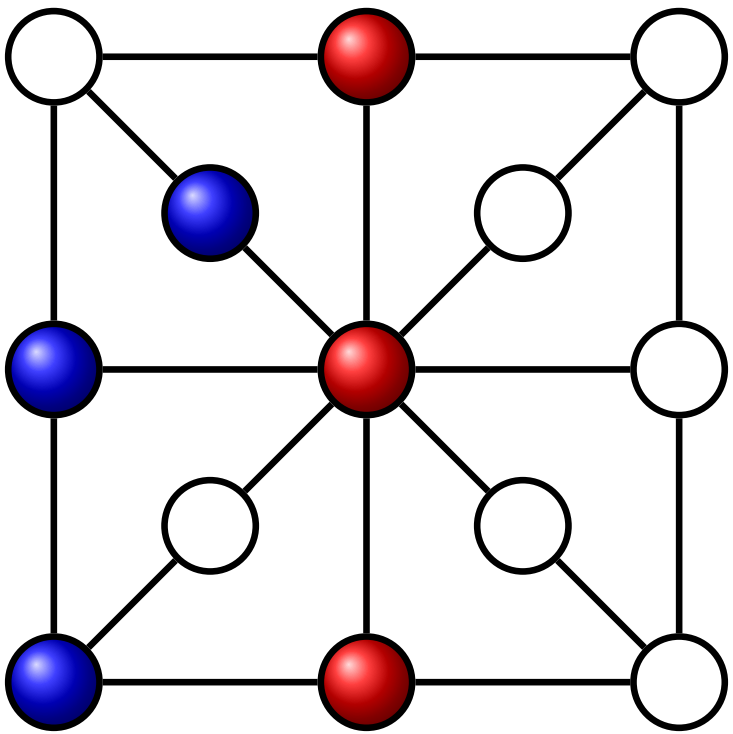
Il était une fois
une princesse qui dessinait des graphes



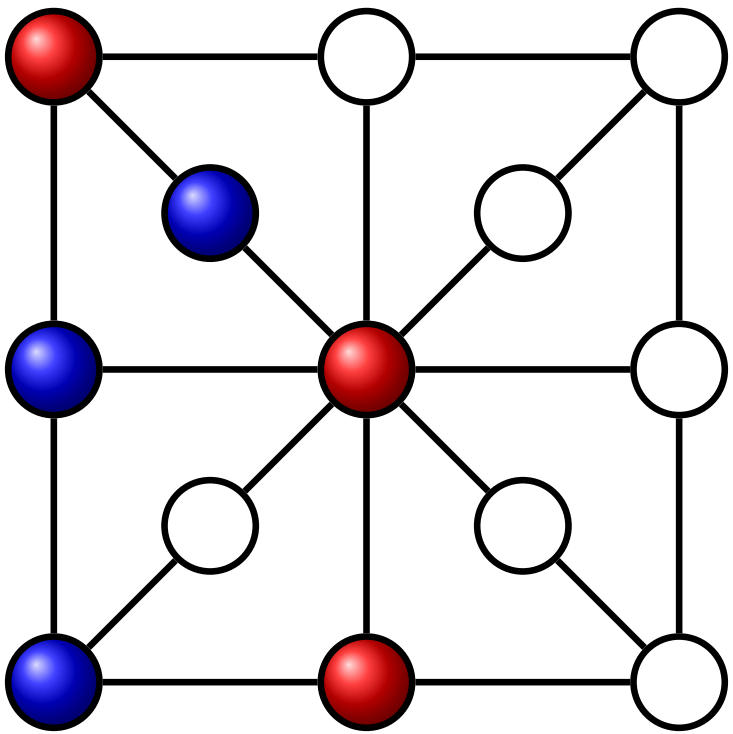
Il était une fois
une princesse qui dessinait des graphes



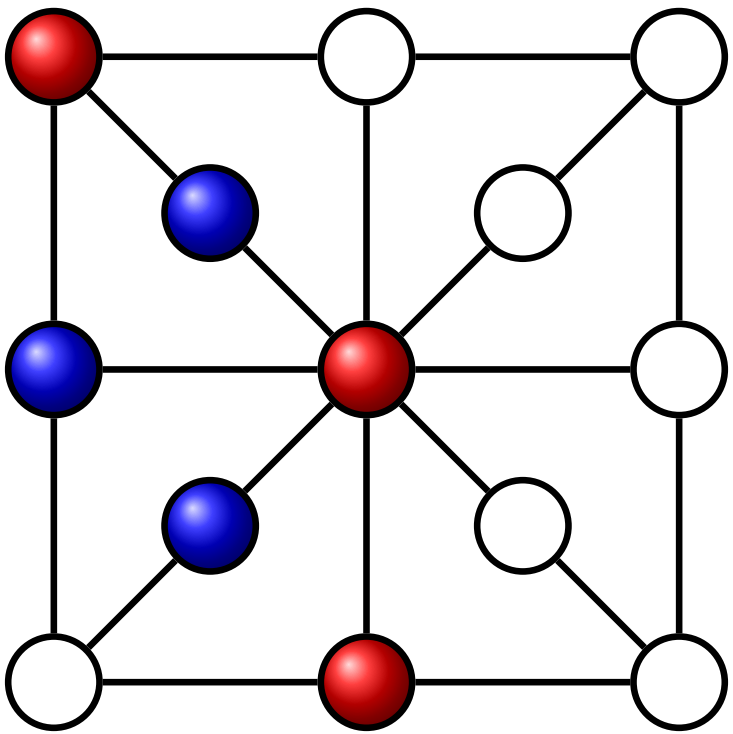
Il était une fois
une princesse qui dessinait des graphes



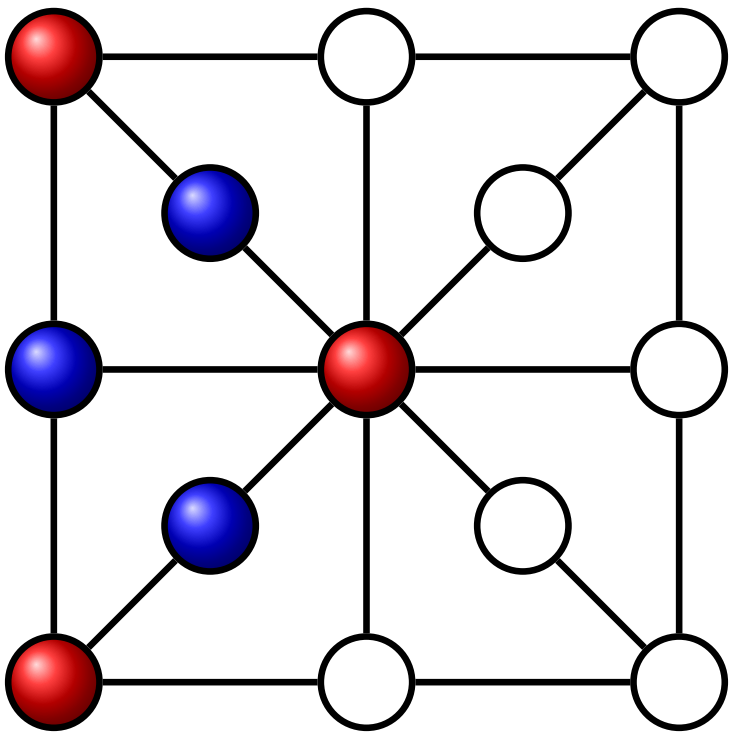
Il était une fois
une princesse qui dessinait des graphes



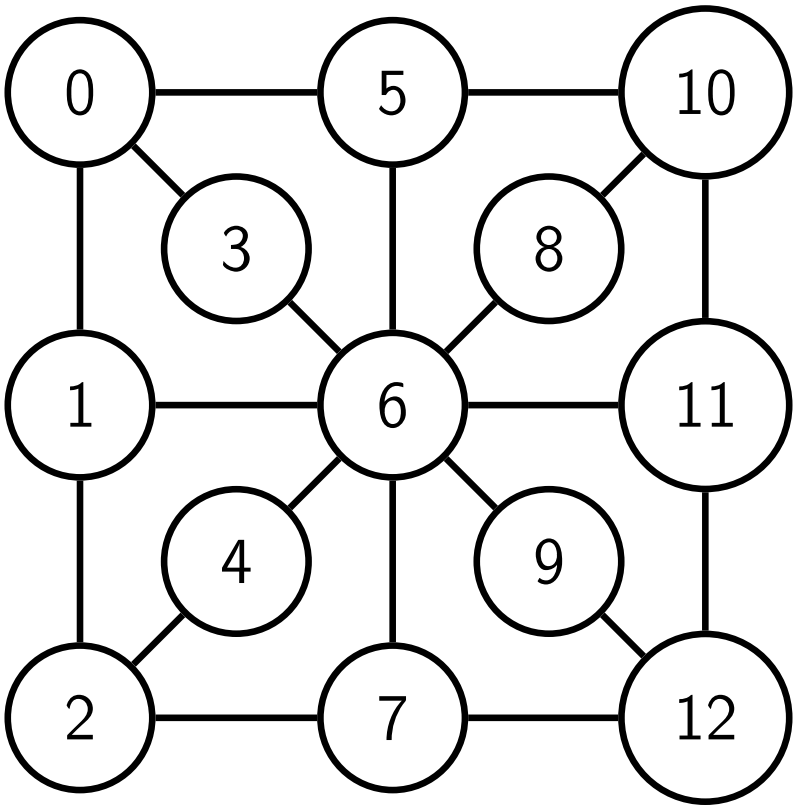
Il était une fois
une princesse qui dessinait des graphes



Il était une fois
une princesse qui dessinait des graphes



Il était une fois
une princesse qui dessinait des graphes



Graphe

```
graphe = [[1,3,5],[2,6],[4,7],[6],[6],[6,10],\
    [7,8,9,11],[12],[10],[12],[11],[12],[]]

def voisins(a,b):
  return a in graphe[b] or b in graphe[a]

bleus = [0,1,2]
rouges = [10,11,12]

def vide(sommet):
  return sommet not in bleus \
      and sommet not in rouges
```

choix possibles

```
def bleupeut(a,b):
  return a in bleus \
      and voisins(a,b) \
      and vide(b)

def choixbleus():
  return [(a,b) for a in bleus \
          for b in range(13) \
          if bleupeut(a,b)]

def rougesgagnent():
  return rouges==[0,1,2] \
      or len(choixbleus())==0
```

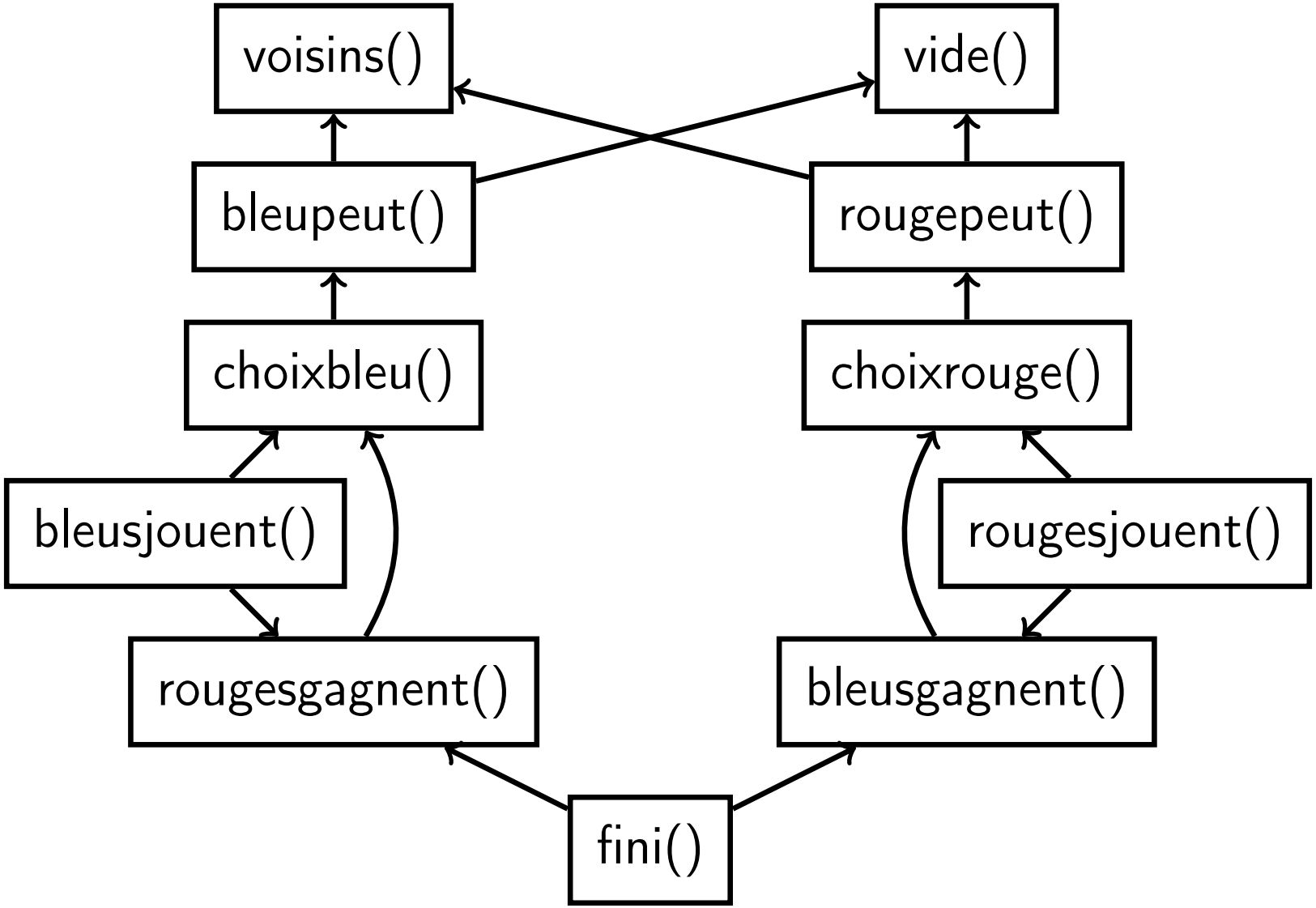
jouabilité

```
from random import choice


def bleusjouent():
    if not rougesgagnent():
        debut,fin = choice(choixbleus())
        bleus.remove(debut)
        bleus.append(fin)
        bleus.sort()


def fini():
    return bleusgagnent() or rougesgagnent()
```

Relations entre fonctions



jeu

```
while not fini():
    rougesjouent()
    bleusjouent()
```

Programmation Objet
pour le jeu et l'étude du jeu

```
def a_game(self):
    while not self.gameover:
        self.gameover = self.play()
        self.next_player()
    return 1 - self.player, self.trace


def play(self):
    strategie = self.strategies[self.player]
    new_cfg = strategie()
    self.trace.append(new_cfg)
    self.config = new_cfg
    return self.final_cfg()
```

Stratégies
le squelette commun

- À partir de la configuration `self.config`
- Récupérer la liste des configurations atteignables parmi `self.allcfgs`
- À chacune desquelles on associe un entier (on a donc des couples)
- Parmi ces couples on en choisit un suivant un critère

Stratégies
Au hasard

```
def alea(self):
    cfg = self.config
    possibilities = [(ncfg, 0)
            for ncfg in self.allcfgs[cfg]]
    return random.choice(possibilities)[0]
```

Stratégies
Avancer souvent

```
def oriented(self):
    cfg = self.config
    possibilities = [(ncfg, self.move_played(cfg,ncfg)[1])
                     for ncfg in self.allcfgs[cfg]]
    possibilities.sort(key=lambda e:e[1],\
                       reverse=bool(self.player()))
    nb = len(possibilities)
    index = int(random.triangular(0,nb,0))
    return possibilities[index][0]
```

Stratégies
Bloquer l'adversaire

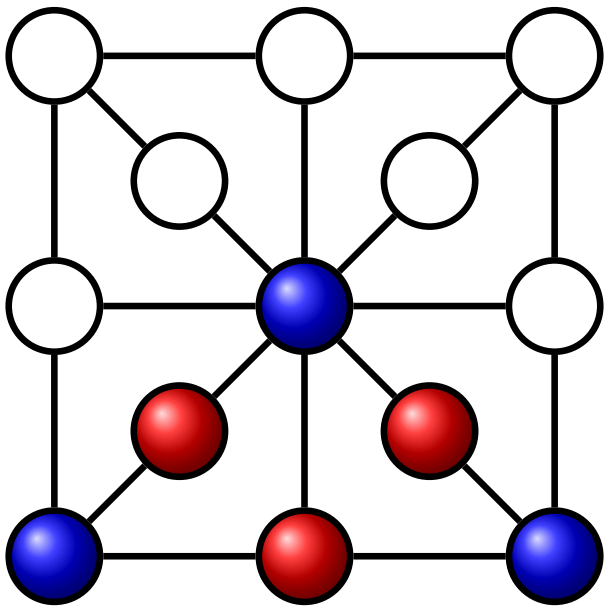
```
def less_liberties(self):
    cfg = self.config
    possibilities = [(ncfg, self.liberties(ncfg,1-p))\
                     for ncfg in self.allcfgs[cfg]]
    best = self.minimum(cfg, possibilities, self.player)
    if best is None:
        best = self.alea()
    return best
```

Stratégies
Bayésienne

```
def bayesien(self):
    cfg = self.config
    memories = self.memories[self.player]
    possibilities = [(ncfg, memories[ncfg])
                     for ncfg in self.allcfgs[cfg]]
    possibilities.sort(key=lambda e:e[1], reverse=True)
    try:
        return tirage(possibilities)
    except:
        return random.choice(possibilities)[0]
```

Théorème

Hoarau, 2018



théorème de Hoarau

Ce genre de configuration est impossible sans tricher.

Conclusion

Pour l'intelligence artificielle,

- Il faut des centaines d'exercices pour acquérir une compétence
- Des exercices trop semblables mènent à un *surapprentissage*
- Apprendre des méthodes par cœur est contre-productif.

Qu'en est-il de l'apprentissage des élèves humains ?