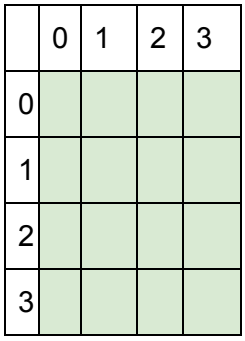
Eight_puzzle_ai.py - lab4

- **calculeaza_succesori(self,nod)**
  - Determina toate mutarile posibile (sus,jos,stanga,dreapta) care pot fi facute de la ind_gol




```
- if linie_gol >= 1:
- # daca linia este >=1, atunci sigur putem face o mutare in
  sus.
-             poz_mutare = (linie_gol - 1) * Nod.NR_COLOANE +
  coloana_gol
-             reprez_noua = self.interschimba(ind_gol,
  poz_mutare, nod.info)
-             h_nod = self.calculeaza_h(reprez_noua)
-             l_succesori.append((Nod(reprez_noua, h_nod), 1))
-         if linie_gol <= 1:
- # daca linia este <=1, atunci sigur putem face o mutare in
  jos.
-             pass #...
-         if coloana_gol >= 1:
- # daca coloana este >=1, atunci sigur putem face o mutare la
  stanga.
-             pass #...
-         if coloana_gol <= 1:
- # daca coloana este <=1, atunci sigur putem face o mutare
  ladreapta.
-
-             pass #...
-         return l_succesori
```

Maze_ai.py - lab5

- **Calculeaza_h(self, reprez_nod)**
    - Reprez_nod este configuratia curenta a labirintului. Prin self avem acces la reprez_scop. O varianta de euristica e aceea de a calcula distanta intre locatia cifrei 1 in reprez_nod si locatia cifrei 1 in reprez_scop.
- **Calculeaza_succesori(self, nod)**
    - Nod este o configuratie curenta de labirint. Pornind de la locatia in care se afla cifra 1, se poate stabili toate mutarile posibile (sus, jos, stanga, dreapta) si vor fi adaugate in l_succesori.

Misionari_canibali_a_star.py - lab6

- **calculeaza_succesori(self, nod)**
    - Nod este configuratia curenta de joc.
    - Determinati un numar de misionari si de canibali care sa fie mutati de pe malul curent pe celalalt avand in vedere faptul ca:
        - numarul de misionari de pe maluri (daca acestia mai exista) trebuie sa fie >= decat numarul de canibali. Si in barca, daca exista misionari, numarul lor trebuie sa fie >= cu cel al canibalilor

Lab_8-x_and_zero_human_vs_computer.py

- mutari(self,jucator_opus)
    - Prin self avem acces la matr care este o lista de “#”,”0” si “x”. Peste tot unde avem “#” vom pune jucator_opus si vom obtine obiecte de tip Joc. Toate aceste obiecte vor fi adaugate in l_mutari
- linie_deschisa(self, lista, jucator)
    - Lista are 3 elemente (luate de pe linii, coloane, diagonale din tabla de x si 0).
    - Verificam daca, plasand jucator (care poate avea valoarea x sau 0) in aceasta lista, aceasta ar putea oferi o sansa de castig jucatorului care joaca cu simbolul jucator. O modalitate de a verifica acest lucru consta in a verifica daca lista contine doar “#” si simbolul jucator. Daca da, returnati 1, altfel returnati 0