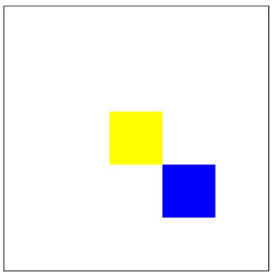
Domáca úloha 2

Našiel som schelling.pdf cez fulltextsearch v bookkite učebnice, a prečítal som si ho.

Čiže potrebujem:

- Vytvoriť tabuľku/matrix, ktorá reprezentuje „UV mapu“ torusu
- $Rat(io) = \frac{N° \; agentov \; vlastnej \; farby}{N° \; \forall \; agentov}$
- Čiže, ak chcem aby *agent* bol *šťastný*, tak v jeho okruhu (v tomto prípade v tabuľke je to 3x3 s agentom v strede musí byť pomer agentov jeho farby väčší ako jeho tolerancia
- Mám pár otázok:
    - Ako chcem robiť toleranciu, na jedno pole alebo všade rovnaká?
        - Odpoveď: Skúsim najprv všade rovnakú
    - Akú veľkú tabuľku urobiť?
        - Ešte neviem ani kresliť v pythone, skúsim najprv 5x5

No o 15 minút neskôr som trochu zabojoval s grafom a mám toto:



Chcel som sa dozvedieť viac o Schellingovom modeli a narazil som na toto video.

Vidím že mi treba oveľa viac buniek ako 5x5 aby som videl „segregáciu“, tak urobím 25x25 maticu. Mením deklaráciu z

rep = [['E', 'E', 'E', 'E', 'E'], ['E', 'E', 'E', 'E', 'E'], ['E', 'E', 'Y', 'E', 'E'], ['E', 'E', 'E', 'B', 'E'], ['E', 'E', 'E', 'E', 'E']]	na	matrix = [['E' for _ in range(25)] for _ in range(25)]

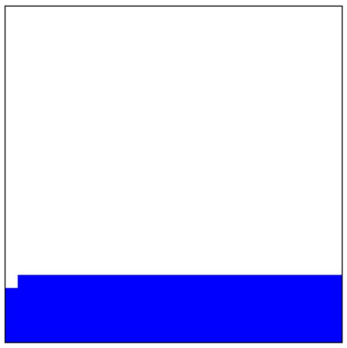
Mal som problémy s jupyter notebook tak prechádzam na VSCode ako editor.

Hm, ale mám prázdne pole, skúsim nejako urobiť pseudo-náhodnú generáciu žltých a modrých členov.

Snažil som sa vytvoriť funkciu pomocou času, ale kód sa spracúva príliš rýchlo:

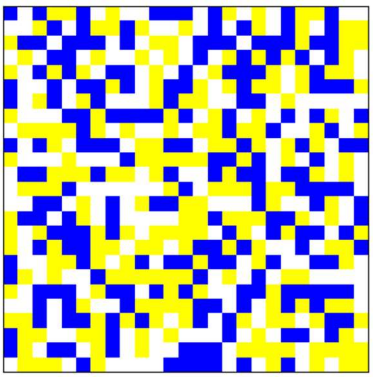
Trvalo mi to cca 15 pokusov, dokým som dostal toto:

```
def whatColor():
   cur_time = time.time()
   ms = int(cur_time * 1000) % 10
   if ms < 3:
      return 'E' #biela
   elif ms < 6:
      return 'Y' #zlta
   else:
      return 'B' #modra
```



Back to the drawing board:

rep = [[random.choice(['E','Y','B']) for _ in range(25)] for _ in range(25)]

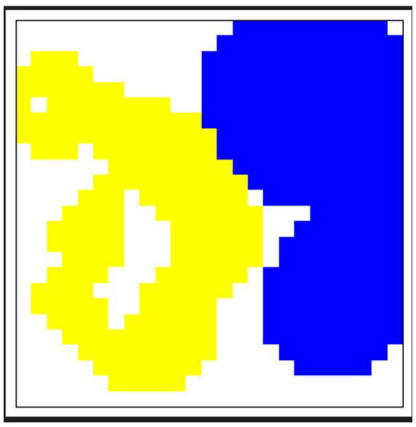


Tak tentokrát to už vyzerá dobre, teraz idem zaimplementovať schellingov model.

Okej, trvalo mi to asi 1,5h ale nakoniec som vymyslel checkovanie satisfaction tak, aby kontrolovalo vsetkych 9 clenov, ale ak je out of bounds tak proste tam doplni E ako prazdne.

Ok, tam kde je prazdne je satisfakcia 1

Zaujimavy vysledok pocas debugovania:



Počas behov som narazil na to, že ak mám funkcie step(rep) v jednej funkcii tak je výpočet do 10 sekúnd a ak ich rozdelím na funkciu, ktorá berie tieto údaje z step(rep) a iteruje cez nich tak 10+ minut potom som to vzdal.

Zistil som že som mal zle poskladané funkcie a prepočítavalo matice spokojnosti viackrát, aj keď nemuselo. Optimalizoval som kód dokým beh na matici 25x25 netrval menej ako minútu, nechce sa mi dlho čakať. Taktiež som nahradil inkrementovanie premien funkciami sum() na miestach, kde sa to dalo.

Celé som to prepísal, boli tam funkcie, ktoré som vôbec nepoužíval ako kontrola rohov.

Otázky a odpovede

Otázky

1. Aký je vplyv parametru **tol?**
2. Vývoj N° spokojných jednicov v čase?
3. U jakého **tol** nastáva segregácia?
4. Vplyv veľkosti okolia?
5. Možné rozšírenia modelu?

Odpovede

1: Parameter **tol** ovplyvňuje zgrupovanie agentov, t.j. ako veľmi im bude vadiť, že majú suseda inej farby. Čím väčší **tol**, tým viac im to vadí.

V prípade môjho algoritmu je to pomer: $satisfaction = \frac{počet\ buniek\ rovnakej\ farby}{počet\ všetkých\ buniek}$

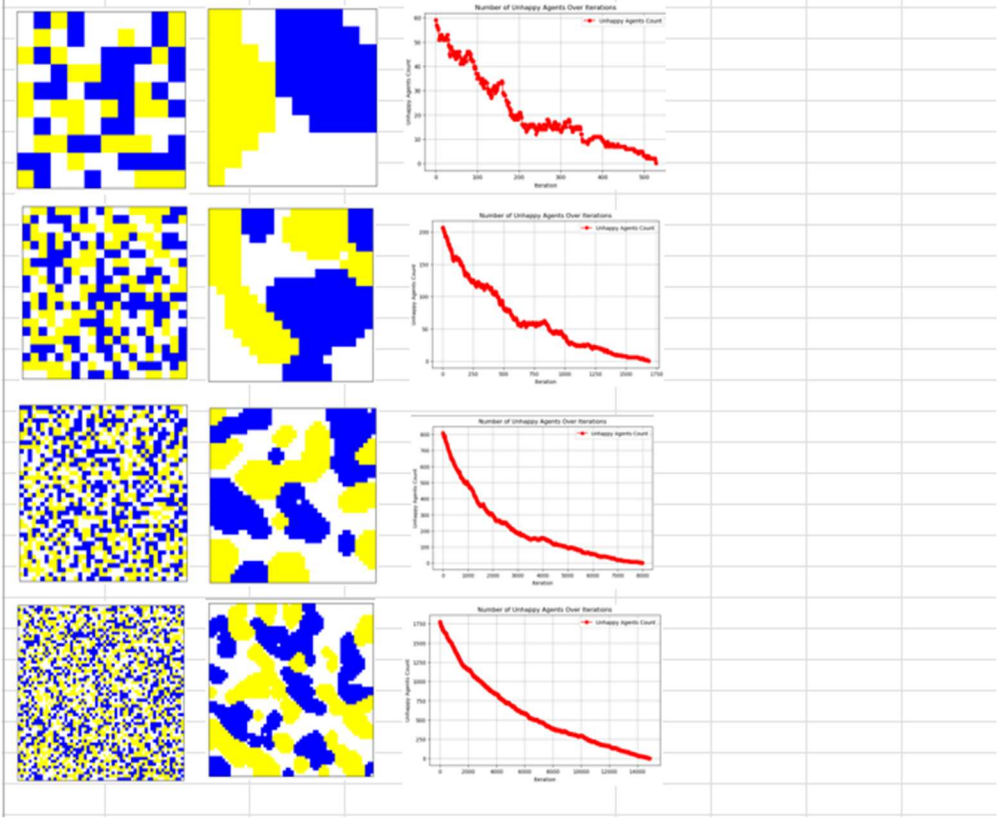
Kde sa porovnáva **tol** voči **satisfaction.**

2: Vývoj počtu spokjných jedincov pri tolerancii 0.5 to vyzerá približne ako funkcia

$y = -log(x)$

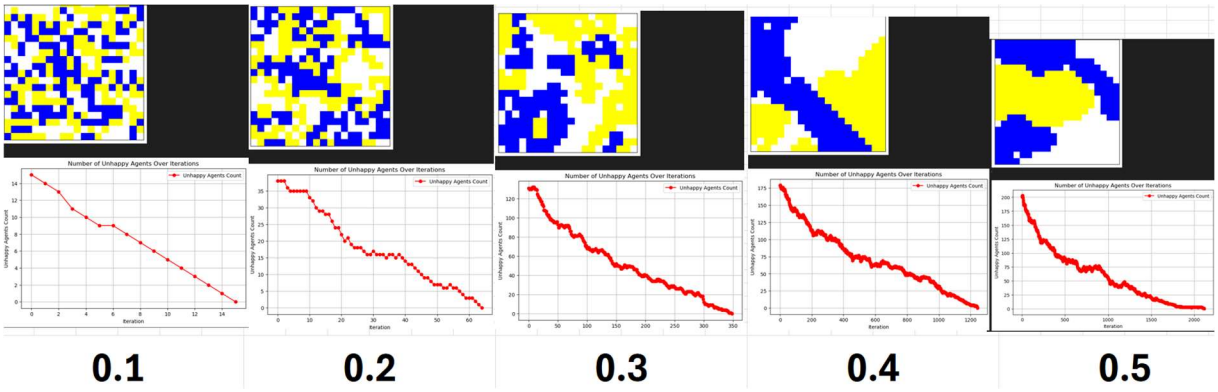
Počet Iter.	Veľkosť Matice	Čas v s	nasobok oproti predošlemu
370	10	0,30	N/A
1955	20	3,30	11
7162	40	52,40	15,87878788
14865	60	216,50	4,131679389

tolerancia 0.5 všade

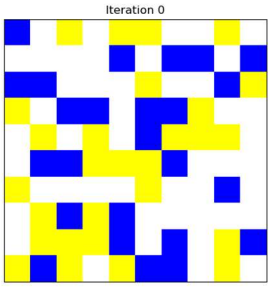


Keď som dal model nad toleranciu 0.5 tak mi nenašlo výsledok (infinite loop)

3: Segregácia u **tol** nastáva už okolo 0.3, kedy vidieť že sa bunky dávajú do skupín.



4: Vplyv väčšieho prostredia: Pri 2x viac prázdneho miesta sa mi dialo toto: (gif v pdf nefunguje, je priložený v zipku)



Následne ak som dal 3x väčšie prostredie a väčší počiatočný matrix tak všetky bunky boli „šťastné“.

Čím väčšie okolie, resp. čím viac voľného miesta, tak tým „šťastnejšie“ bunky.

5: Možné rozšírenie modelu by som si predstavoval, najmä nad **tol** 0.5, že bunky budú dohľadávať väčšie skupiny, a pripojovať sa k ním,
alebo jednoduchšie nechodiť znova na prázdne miesto, kde už bunka bola