# Domáca Úloha 3

Počiatočne som začal s kódom, ktorý som našiel na internete, aby som mal predstavu.

Algoritmus bežal okolo 80min a našiel ofarbenie grafu dsjc125.9 na 50 farieb, s čím som nebol spokojný.

Vlastný algoritmus: moja prvá idea bola, že náhodne ofarbím všetky polia a potom postupne budem meniť farby, dokým graf nebude správne ofarbený (t.j. všetko random walk) ale po znovu-zamyslení sa som si uvedomil že to je nereálne (všetko cez random = extrémne dlhý čas na výpočet + nemusí vždy fungovať).

Môj nový nápad pozostáva z toho, že budem využívať **náhodné ofarbenie,** následne **hill climb** a ak potom uviazne tak **random walk**.

Pre prácu s grafom využívam **networkx**, na zistenie ako funguje syntax pre prácu s knižnicou som si vypísal toto:

```
# zobrazenie bodov a hran
print("Nodes:", list(G.nodes()))  # body
print("Edges:", list(G.edges()))  # hrany

# struktura grafu
print("\nGraph as adjacency list:")

for node in G.nodes():
   neighbors = list(G.neighbors(node))
   print(f"{node}: {neighbors}")
```

No, cez ten hill climb to bolo dosť dlho čakať na výsledok, a vždy mi to nenašlo ten počet farieb aký chcem. Skúšal som to s trochu „ľahším“ grafom: dsjc125.1.col aby som nemusel dlho čakať. Tento graf má mať 5 farieb, ale našlo mi to najlepšie 8.

Vymyslel som to následovne:

- Hill climb ako main vec
    - Kontrola či sa zmenší počet konfliktov
        - Ak áno, super, našli sme novú farbu
        - Ak nie, vráť originálnu
- Random walk každých 10 steps
    - Vyber random node a meň mu nahodne k-krat farbu
        - Každý pokus kontrola či sa zmenší počet konfliktov
            - Ak áno, super
            - Ak nie, vráť originálnu

Ak algo prejde 10k steps, tak mi navýši počet pokusov a farbu. Počet max pokusov som dal na 10. Spustil som to na grafe zo zadania, som zvedavý ako to dopadne (idem sa prejsť :D).

Po 856 minútach som beh grafu vypol, nechcem si brať algoritmus so sebou do hrobu.

Po súperení s mojou naivitou a hlúposťou som sa úplne rozhodol vypustiť chatbotov a robiť to sám, keďže aj mozog potrebuje niekedy námahu a nechcem skončiť tak, že mi Alčko bude hovoriť všetko čo mám robiť.

Po chvíľke rozmýšľania som vymyslel jednoduchý algoritmus, ktorý mi pomohol sa dostať k celkom zaujímavému výsledku, ale má pár problémov, ktoré chcem vyriešiť cez hill climb algoritmus.

Tento algoritmus spočíva v tom, že:

- Navštívim prvý bod, ktorý skúmam
- Nastavím mu farbu X
- Všetkým jeho susedom nastavím novú farbu Y
- Pokračujem do ďalšieho skúmaného bodu
- Nastavím mu farbu Z
- Všetkým jeho susedom...

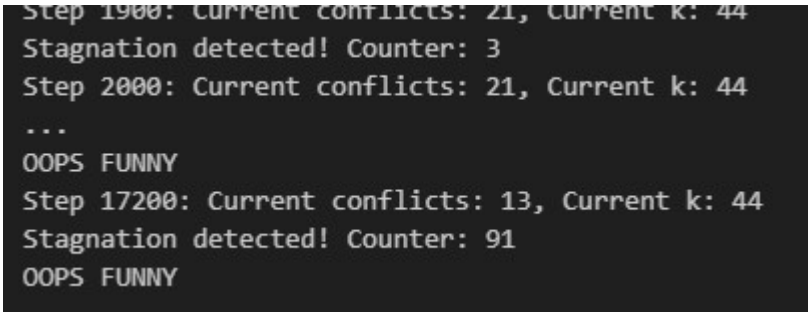
Nafarbilo mi to graf dsjc125.9 na 55 farieb, čo je celkom zaujímavé a je extrémne rýchly.

Hlavný problém je taký, že graf nie je správne ofarbený, skúšal som na to urobiť funkciu ale po 15 minútach behu som to vzdal. (Problém bol v implementácií checkovania správnosti ofarbenia grafu, kvôli posunutým farbám od 1 nie od 0)

Vytváram hill climb algoritmus, ktorý:

- Zavolá funkciu simpleAdjacencyColoring
- Priradí z nej col, k
- Načitá target_k
- Vyberie random číslo / farbu
- Bude na týchto miestach skúšať farby z existujúcich farieb bez predošlej vybranej farby, dokým nebude graf správne zafarbený

Vzdal som tento postup, nefungovalo to: Hill climb sa snaží zmenšiť počet konfliktov, ale môj algoritmus na simple adjacency coloring nemá žiadne konflikty.
Namiesto toho som zlepšil svoj hill climb a zaimplementoval po 10x rovnakom výsledku stagnation counter, ktorý triggeruje beh random walk, ktorý pomáha sa dostať von z lokálneho „loopu“. Myslím že problém je v tom, že cykluje medzi rovnakými bodmi, ktorých zmena farby nepomáha sa dostať k skutočnému výsledku. Mám tam 20% šancu na ten random walk ale podľa toho čo vidím tak budem to musieť dať trochu vyššie.
Prosím, tolerujte môj profesionálny debug message na beh random walk ...




1 stagnácia = 10x rovnaký výsledok za sebou, čiže som mal ~31x 10x rovnaký výsledok za sebou. Neresetujem tento inkrement, preto je veľký.

Taktiež zaujímavá vec, keďže to je na pseudo-náhode, tak niekedy resp. niektorý beh algoritmu je lepší ako ten predošlý. Na screenshote to bol celkom dobrý beh, bežalo to 15min a potom som to vypol.

Najprv som skúšal každých 10 stagnacii šancu na random walk, zmenil som to na ak sa 3x opakuje rovnaké číslo 100% že urobí random walk a vyzerá to, že to funguje.

Po 30min behu som sa nedokázal dostať pod 7 konfliktov, preto sa pokúsim s niekým porozprávať a skúsiť niečo ďalšie.

Všimol som si tabu search, pokúsim sa to zaimplementovať, znie to sľubne.

Do existujúceho algoritmu som nedokázal „vopchať“ tabu search, tak som sa rozhodol program znova prepísať.

## Výsledná verzia algoritmu

**Load web graph** je funkcia, ktorá načíta getovacím requestom algoritmus.
**Count conflicts** je funkcia, ktorá spočíta konflikty v grafe. Konflikt = 2 nodes s rovnakou farbou vedľa seba.
**Get_conflicting nodes** rovnaká ako count_conflicts ale pridáva do setu nodes, ktoré majú rovnakú farbu a vráti ich.
**Random walk** vykoná random walk s prioritou na conflicting nodes, ak nie sú konflikty ide na náhodný node. Vyberá možné farby, ktoré nepoužívajú susedia a mení node z dostupných farieb.
**Is stagnating** je jednoduchá funkcia ktorá vráti **True** alebo **False** na základe posledných 20 výsledkov a threshold: ak je viac rovnakých výsledkov ako threshold, vráti **True.**
**Hill climbing with walks** je hill climb rozšírený o **stagnation detection** a prioritou konfliktujúcich nodes, taktiež periodicky (80% šanca) mení medzi hill climb a random walk.
**Tabu search** je funkcia, ktorá zaznamenáva predošlé výsledky v tzv. „taboo“ tabuľke a skúša nové možnosti. Riešenie je taboo práve vtedy, ak sa už vyskytuje v tabuľke, vtedy sa preskakuje. Riešenie sa prijíma práve vtedy, ak nie je taboo alebo je lepšie ako riešenie doteraz.
**Simulated annealing** je funkcia, ktorá obsahuje teplotu a cooling rate. Skúša náhodné možnosti, v tomto prípade s 90% šancou na vybratie farby, ktoré nekonfliktujú so susedom, ináč vyberá zo všetkých farieb. Rozhodnutie je prijaté na základe „teploty“.

Beh na algo dsjc125.9 s 44 farbami:

```
===== Attempting coloring with k=44 colors =====
Attempting to color graph with 44 colors (nodes: 125, edges: 6961)
Initial greedy coloring has 155 conflicts
Phase 1: Hill Climbing with Random Walks (12000 steps)
Step 0: Best conflicts = 155, Current conflicts = 155
Step 500: Best conflicts = 17, Current conflicts = 34
Step 1000: Best conflicts = 13, Current conflicts = 88
Step 1500: Best conflicts = 13, Current conflicts = 21
Step 2000: Best conflicts = 13, Current conflicts = 77
Step 2500: Best conflicts = 13, Current conflicts = 37
Step 3000: Best conflicts = 13, Current conflicts = 86
Step 3500: Best conflicts = 13, Current conflicts = 33
Step 4000: Best conflicts = 12, Current conflicts = 76
Step 4500: Best conflicts = 12, Current conflicts = 38
Step 5000: Best conflicts = 12, Current conflicts = 85
Step 5500: Best conflicts = 12, Current conflicts = 40
Step 6000: Best conflicts = 12, Current conflicts = 92
Step 6500: Best conflicts = 12, Current conflicts = 35
Step 7000: Best conflicts = 12, Current conflicts = 82
Step 7500: Best conflicts = 12, Current conflicts = 34
Step 8000: Best conflicts = 11, Current conflicts = 76
Step 8500: Best conflicts = 10, Current conflicts = 22
Step 9000: Best conflicts = 10, Current conflicts = 77
Step 9500: Best conflicts = 10, Current conflicts = 24
Step 10000: Best conflicts = 10, Current conflicts = 83
Step 10500: Best conflicts = 7, Current conflicts = 20
Step 11000: Best conflicts = 7, Current conflicts = 84
Step 11500: Best conflicts = 7, Current conflicts = 26
Phase 2: Tabu Search (12000 steps)
Tabu Search Step 0: Best conflicts = 5
Tabu Search Step 500: Best conflicts = 2
Tabu Search Step 1000: Best conflicts = 2
Tabu Search Step 1500: Best conflicts = 1
Successfully found coloring with 44 colors using tabu search!
[SUCCESS]: Found valid coloring with 44 colors in 1040.64 seconds!
[COLORING]:  [41, 40, 30, 3, 26, 3, 25, 29, 4, 20, 10, 9, 27, 28, 42, 5, 20, 16, 23, 38, 24, 14, 32, 13, 27, 33, 40
Color correctness: True
```

Failed to find valid coloring. Best solution has 2 conflicts.
[FAILURE]: Could not find valid coloring with 43 colors after 4804.22 seconds.
Stopping search. Minimum colors found:  44

**Beh trval 144m 48.2s**

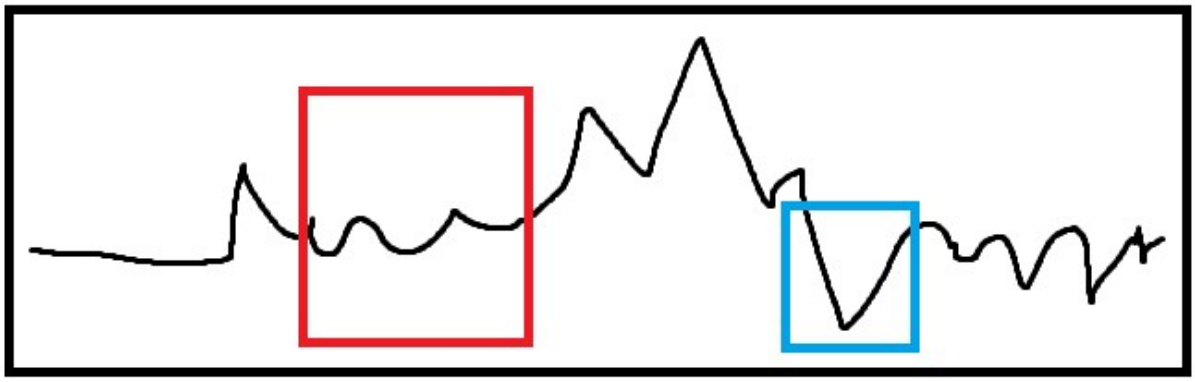
## Hill climb

Hill climb je algoritmus, ktorý sa snaží optimalizovať riešenie tým, že vyberá náhodné body a ich susedov, mení ich a ak sa riešenie (v tomto prípade zníženie počtu konfliktov alebo kolízií) stane lepším, tak ho nechá, ak nie tak skúša ďalej.

Výhody: Celkom ok rýchlosť, pri jednoduchých riešeniach funguje super

Nevýhody: Závislosť na náhode, všimol som si že ak spúšťam algoritmus na domácej úlohe, tak sa vie „zaseknúť“ a nepokračuje už ďalej. Pri zdĺhavých riešeniach trvá exponenciálne dlhšie.

Do svojej úlohy som zaimplementoval detekciu stagnácie, a následne vykonanie random walk, ktoré pomáha dostať sa z tzv. lokálneho minima. Znamená to, že ak mám napr. hľadanie najnižšej hodnoty, môže sa stať že hill climb sa bude pohybovať len v červenom štvorci, ale „nedostane sa odtamadiaľ“ lebo kvôli implementácií si myslí že ďalej sú už len vyššie hodnoty.



## Tabu Search

Tabu search je taký vylepšený hill climb - robí to, že si pamätá posledné riešenia a pridáva ich do „taboo“ tabuľky. V praxi to znamená to, že ak nájde riešenie, ktoré je už v taboo tabuľke, tak ho preskočí. Zároveň porovnáva medzi riešeniami v tejto tabuľke a vyberie najlepšie z nich (aj keď je celkovo horšie), čo pomáha dostať sa von z lokálneho minima.

## Simulated Annealing

Zdroje:
https://en.wikipedia.org/wiki/Simulated_annealing
https://www.machinelearningplus.com/machine-learning/simulated-annealing-algorithm-explained-from-scratch-python/
Tzv. simulované kalenie:

- Optimalizačný algoritmus
- Vyberie náhodný stav
- Rozhodne či vybrať výsledok na základe „teploty“ a hodnoty funkcie

Ako mám chápať túto teplotu, a funkčnú hodnotu? Môžem to jednoducho vysvetliť na svojom algoritme (aj keď ho neuvidíte).

Problém je nasledovný: Počas hľadania najlepšieho riešenia mám určitý počet konfliktov, a snažím sa ho znížiť, ale neviem ísť pod určité číslo (keďže výber je pseudonáhodný). Annealing pomáha sa dostať k výsledku tým, že pomocou teploty vyberá niekedy aj horšie riešenie tzn. escapuje local minimum.

Ak je funkčná hodnota vyššia a teplota nižšia, je väčšia šanca vybrať toto riešenie.

```
# delta
delta = new_conflicts - old_conflicts
# vypocet sance prijatie riesenia
 delta <= 0 or random.random() < math.exp(-delta / max(0.01,
temperature)):
 # prijat a prepocitat konflikty
current_conflicts = count_conflicts(G, current_col)
```