PDAJ zadatak za ocenjivanje

Problem koji se rešava:
- Generisati koordinate dvodimenzione table na osnovu prosleđenih vrednosti za veličinu table (n i m).
- Navesti niz koordinata polja koje se nalaze na tabli i koja predstavljaju specijalna polja. Broj polja u ovom nizu treba da bude znatno manji od ukupnog broja polja.
- Za svako od polja na tabli pronaći udaljenost do svakog od specijalnih polja.
- Pronaći najbliže specijalno polje svakom polju na tabli.
- Kao rezultat, za svako od polja vratiti indeks njegovog najbližeg specijalnog polja iz niza specijanih polja.
- Ako je neko polje jednako udaljeno od dva specijalna polja, vratiti indeks bilo kojeg od njih.

Zadatak rešiti na sledeće načine:
- sekvencijalno
- upotrebom list comprehension-a
- upotrebom generatora
- upotrebom multiprocessing biblioteke

Rešenje prvo kreirati zasebno za svaki od načina rešavanja u programskom jeziku Python, a zatim u obliku Django rest API-ja.

Za svaki od načina rešavanja kreirati odvojenu putanju (npr: calculation/sequential, /calculation/multiprocessing, ...). Svaka putanja prima ulazne podatke u istom obliku i na osnovu njih vraća izlazne podatke u istom obliku kao sve ostale putanje.

Ako API zahtev u JSON obliku i izgleda ovako:

```
{
        'n':'10',
        'm':'10',
        'points':['1,3', '3,2', '6,8', '9,6', '5,5']
}
```

Očekivani rezultat bi bio:

```
{
        'result': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1, 0, 0, 0, 0,
4, 2, 2, 1, 1, 1, 1, 1, 4, 4, 4, 2, 2, 1, 1, 1, 1, 4, 4, 4, 2, 2, 2, 1, 1, 1, 4, 4, 4, 4, 2, 2, 2, 1,
1, 1, 4, 4, 4, 4, 2, 2, 2, 1, 1, 4, 4, 4, 4, 3, 2, 2, 2, 1, 4, 3, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3, 3, 3, 3,
3,
3, 3, 3],
        'time_in_s': 0.0213773250579834,
        'max_memory_in_MB': 0.035714
}
```

Izabrati veličinu table tako da sekvincalno vreme izvršavanja bude barem 5 sekundi. Svakoj putanji proslediti iste koordinate specijalnih polja i veličinu table. Na osnovu dobijenih rezultata popuniti sledeće tabele:

	n	m	points
vrednost	1000	1000	["11,300","9,420","1,3","15,15","76,54","89,430","336,423","211,211","350,350","426,18","0,500","7,420","766,894","999,3","876,542","667,718","319,54","1,800","803,20"]

Način rešavanja	Vreme u s	Memorija u MB	Broj jezgara	Zaključak
sekvencijalno	42.63	102.13	1	Brzo i efikasno se izvršava nad tabelom manjih dimenzije i jednako je dobro rešenje kao i svako drugo u tim slučajevima. Vremenska razlika među rešenjima skoro da i nije vidljiva do dimenzija 500x500, dok je zauzeće memorije u toku sekvancijalnog načina izvršavanja uvek znatno veće.
comperhension	40.49	102.12	1	Zauzeće memorije u toku izvršavanja comperhension-a uvek je skoro isto kao i kod sekvencijalnog izvršavanja, dok zaostaje u tom segmentu za generatorima i multiprocessing-om. Što se tiče brzine, uvek je brže od sekvencijalnog načina, što je izraženije sa povećanjem dimenzija tabele, isto kao i vremensko zaostajanje comperhension-a od preostala 2 načina izvršavanja.
generator	35.14	8.29	1	Generator je što se tiče zauzeća memorije u procesu testiranja rešenja uvek bio bolji od sekvencijalnog i comperhension rešenja (i sa tabelom malih i sa tabelom velikih dimenzija) dok je uvek sličan zauzeću koje daje multiprocessing. Čak i nad tabelama većih dimenzija generator se izvršava relativno brzo. Ali se njegovo zaostajanje u brzini u odnosu na izvršavanje multiprocessing-om nad većim tabelama kao što je 2000x2000 znatno vidi (1.5 min). Ta razlika nastavlja se povećavati sa povećanjem dimenzija tabele i povećanjem broja specijalnih polja.
multiprocessing	19.39	8.67	4	Multiprocessing metoda se najbolje pokazuje nad tabelama velikih dimenzija. Zauzeće memorije je uvek tolerantno i u većini slučajeva jednako zauzeću koje daje rešenje sa generatorima. Ovo rešenje u brzini i te kako prednjači pogotovo sa povećanjem dimenzija tabele. Map_unordered nisam koristila u svom rešenju jer mi je bio bitan redosled povratnih vrednosti indexa najbližih specijalnih polja (najbliže specijalno polje koordinate (0,0) mora da se ispiše pre svih ostalih i tako nadalje). Pretpostavljajuće da će se sa povratim indexima najbližih polja nešto raditi izabrala sam imap jer map funkcija čeka da se dobiju svi rezultati da bi se nešto moglo uraditi sa njima dok imap čim dobije svoj prvi rezultat, on se dalje može obrađivati.

- Za računanje vremena izvršavanja proračuna koristiti Python biblioteku time.
- Za računanje maksimalnog zauzeća memorije pri izvršavanju proračuna koristiti Python biblioteku tracemalloc.
- Broj procesorskih jezgara zaključiti na osnovu načina rešavanja zadatka.

Izvršavanje optimizovati za izvršavanje na svom računaru.

- Na osnovu dobijenih rezultata, doneti zaključak o prednostima i manama svakog

načina rešavanja.

- Za multiprocessing obrazložiti i izbor funkcije koja je korišćena (map, imap, imap_unordered).

Za testiranje primera manjih dimenzija table koristiti display_results.py. Prikaz primera biće kreiran u 2 oblika:

- konzolni oblik:

```
Time is     0.0026130676269531 25s
Max memory is    0.036935MB
0   0   0   0   0   0   0   0   0   0
1   0   0   *   0   0   0   0   0   2
1   1   1   0   0   0   0   4   2   2
1   1   *   1   1   4   4   4   2   2
1   1   1   1   4   4   4   2   2   2
1   1   1   4   4   *   4   2   2   2
1   1   1   4   4   4   4   2   *   2
1   1   4   4   4   4   3   2   2   2
1   4   3   3   3   3   3   3   2   2
3   3   3   3   3   3   *   3   3   3
```

- grafički prikaz: