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. Od Django rest API-ja napraviti docker kontejner.

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			

Način rešavanja	Vreme u s	Memorija u MB	Broj jezgara	Zaključak
sekvencijalno				
comperhension				
generator				
multiprocessing				

- 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).

Napomene:
- **Rok za izradu projekta je 23.01.2022. u 23:59.**
- Na git repozitorijumu napraviti granu projekat. Ako već niste, na repozitorijum dodati MilenaVujicic kao contributor-a. Nakon završetka, napraviti pull request i dodati MilenaVujicic kao reviewer-a.
- **Repozitorijum mora sadržati ime, prezime i broj indeksa u README.md fajlu.**
- Po želji, moguće je zameniti jedan od načina rešavanja zadatka (sequential, comprehension, generator ili multiprocessing) za rešenje upotrebom mpi4py biblioteke. U tom slučaju obrazložiti izbor mpi funkcija za rešavanje zadatka. Mpi rešenje takođe optimizovati na svom računaru.
- Za testiranje vremena i memorije moguće je koristiti `display_results.py`.

- Za testiranje primera manjih dimenzija table koristiti funkcije `display_results(results, n, m, points)` i `show_table(results, n, m, points)` u `display_results.py`.

Prikaz primera biće kreiran u 2 oblika:

- konzolni oblik:

```
Time is  0.002613067626953125
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: