	n	m	points
vrednost	800	800	["1,3", "3,2", "6,8", "9,6", "5,5", "3,3", "523,678", "2,5", "253,142", "11,787", "154,705"]

Način rešavanja	Vreme u s	Memorija u MB	Broj jezgara	Zaključak
sekvencijalno	16.9068	58.8095	1 jezgro u trenutku	Mane: - Pri većini testiranja najsporije rešenje - Najveći utrošak memorije - Ne koristi nikakvu vrstu paralelizacije, pa stoga koristi jedno jezgro u trenutku Prednosti: - Razumljiv kod - Okupira malo procesorskih resursa
comprehension	16.3731	58.7772	1 jezgro u trenutku	Mane: - Ne donosi preveliko ubrzanje u odnosu na sekvencijalno rešenje (povremeno i sporije) - U ovom zadatku, sve petlje iz sekvencijalne verzije prevedene su u comprehension formu. Ovo može zahtevati određen trud za pisanje ispravnog koda koji donosi zanemarljivo ubrzanje. - Visok utrošak memorije - Ne koristi nikakvu vrstu paralelizacije, pa stoga koristi jedno jezgro u trenutku Prednosti: - U odnosu na sekvencijalnu verziju, povremeno pokazuje nešto bolje rezultate po pitanju vremena izvršavanja i zauzeća memorije - Na pojedinim mestima čini kod kompaktnijim i smanjuje broj linija - Okupira malo procesorskih resursa
generator	15.2816	5.4644	1 jezgro u trenutku	Mane: - Iako primetno, nije preveliko ubrzanje u odnosu na sekvencijalnu i comprehension verziju - Ne koristi nikakvu vrstu paralelizacije, pa stoga koristi jedno jezgro u trenutku Prednosti: - Brže od sekvencijalne i comprehension verzije - Okupira malo procesorskih resursa - Znatno niži utrošak memorije, s obzirom na to da se rezultat ne čuva kao celina (cela lista) u memoriji
multiprocessing	4.0682	5.7883	8 (logičkih) jezgara u trenutku	Mane: - Troši više procesorkse moći računara - Zahteva organizovanje koda tako da bude pogodno za korišćenje multiprocessing biblioteke - Zahteva više rada od strana OS scheduler-a Prednosti:

				- Znatno manje vreme izvršenja u odnosu na prethodna rešenja, kao rezultat korišćenja paralelizacije - Nizak utrošak memorije, kao rezultat korišćenja *imap* funkcije iz multiprocessing biblioteke. Utrošak memorije zavisi od veličine chunk-a, i raste zajedno sa njim. - Najefikasnije iskorišćenje CPU resursa Obrazloženja bitnih pojedinosti: - I**zbor broja radnika u Pool-u** - Nakon testiranja, ustanovljeno je da je najbolja varijanta sa 8 radnika. 8 je takođe i podrazumevana vrednost, s obzirom na to da je to broj logičkih jezgara korišćenog računara. Postavljanje više radnika nike dovelo ni do kakvog ubrzanja, s obzirom na to da više od 8 procesa ne može na korišćenoj mašini da se izvršava u paraleli, a postavljanje većeg broja radnika takođe dovodi do potrebe većeg uključivanja OS scheduler-a za ravnomerno raspoređivanje svih tih radnik procesa po jezgrima procesora. Manje od 8 jezgara dovodi do usporenja zbog nižeg stepena paralelizacije. - **Izbor veličine chunk-a** - Nakon testiranja, ustanovljeno je da su najbolje performanse kada se svaki radnik bavi približno jednom vrstom matrice table. Manji chunk-ovi dovode do manjeg zauzeća memorije, ali i do potrebe za češćim dodelama chunk-ova procesima, što usporava rad. Veći chunk-ovi povećavaju utrošak memorije i smanjuju mogućnost boljeg planiranja dodele podataka, jer se jedan proces “zarobi” jednim velikim delom ulaznog niza. - **Izbor *imap* funkcije** - Ova funkcija je najbolja opcija zbog toga što koristi generatore, pa stoga smanjuje utrošak memorije, i zbog toga što su rezultati dostupni odmah po sračunavanju, bez potrebe za čekanjem da se obradi ceo niz, kao što je slučaj sa *map*. Funkcija *map* bi takođe utrošila i više memorije jer bi morala ceo rezultat da smesti u nju. Funkcija *imap_unordered* nije pogodna, iako bi možda izvršila posao za kraće vreme, jer je potrebno da konačni rezultati zadrže redosled iz početnog niza.