Usporedba sortiranja: Selection sort, Bubble sort, Insertion sort

Iris Trgovec, 18.10.2023.

UVOD

Sortiranje je problem koji se često javlja u računarstvu, a uključuje permutiranje elemenata u određenom redoslijedu (uzlazno ili silazno) po nekom kriteriju uspoređivanja (npr. numerička vrijednost, leksikografski).

Ulaz za algoritam sortiranja je niz elemenata $(x_1,x_2,\ldots,x_n)$ duljine n, pri čemu se vrijednosti elemenata mogu uspoređivati, tj. elementi su u relaciji totalnog uređaja. Relacija totalnog uređaja je refleksivna, antisimetrična i tranzitivna relacija, a za svaka dva elementa vrijedi točno jedno od sljedećeg:

1. $x_i < x_j$
2. $x_i > x_j$
3. $x_i = x_j$

Izlaz algoritma je permutacija $(x_1',x_2',\ldots,x_n')$ ulaznog niza takva da vrijedi $x_1' \leq x_2' \leq \ldots \leq x_n'$, odnosno, niz je uzlazno sortiran. Iako će se u ovom projektu promatrati uzlazno sortiranje, bit će napomenuto kako se isti algoritmi mogu primijeniti i na problem silaznog sortiranja, to jest, da vrijedi $x_1' \geq x_2' \geq \ldots \geq x_n'$.

Dva osnovna resursa koji algoritmi koriste su vrijeme (vremenska složenost) i memorijski prostor (prostorna složenost). S obzirom na to da sortiranje ima veliku primjenu, npr. kod rada s velikom količinom podataka, traženja specifičnog elementa u bazi podataka, organizacije podataka… bitno je da je ono brzo i efikasno. U ovom radu usporedit ćemo Bubble, Insertion i Selection sort te promotriti njihovo djelovanje u praksi.

Selection sort

Selection sort je vrsta sortiranja izborom, točnije, to je algoritam u kojemu se prvo pronađe najmanji (najveći) element i na neki način izdvoji od ostatka, a potom se proces ponavlja za preostale elemente.

Opis algoritma:

1. Niz se sastoji od sortiranog i nesortiranog dijela. Na početku izvođenja, sortirani dio je prazan, a nesortirani je cijeli niz.
2. U nesortiranom dijelu traži se najmanji element.
3. Nakon što se pronađe najmanji element, on mijenja mjesto s prvim elementom u nesortiranom nizu. Sortirani dio niza povećava se za jedan element, a nesortirani smanjuje za jedan.
4. Sve dok nesortirani dio niza nije prazan, ponavljaju se koraci 2 i 3.

Za silazno sortiranje niza dovoljno je zamijeniti 'najveći' s 'najmanji'.

Složenost

Svaki element osim prvog u nesortiranom nizu uspoređujemo s trenutno najmanjim, zbog tog je ukupan broj usporedbi: $(n-1)+(n-2)+(n-3)+\cdots+2+1=\frac{n(n-1)}{2} \in O(n^2)$.
U svakom koraku je najviše jedna zamjena pa je ukupan broj zamjena u algoritmu najviše $n-1 \in O(n)$.

Stoga, za vremensku složenost Selection sorta vrijedi: $T(n) \in O(n^2)$.

Bubble sort

Da bi vrijedilo da je niz dan kao ulaz algoritma sortiran uzlazno, za zadani niz mora vrijediti da je monotono rastući, tj. da $\forall$ i, j $\in$ {1, . . . , n } vrijedi i < j $\Rightarrow$ $x_i \leq x_j$.

Dovoljno je provjeriti vrijednosti samo za sve susjedne parove indeksa, tj. mora vrijediti $x_i \leq x_{i+1}$. $\forall$ i $\in$ {1, . . . , n -1}. Ako niz nije sortiran, onda $\exists$ i $\in$ {0, . . . , n − 1}
takav da je $x_i > x_{i+1}$.

Opis algoritma:

- Unos niza koji je potrebno sortirati
- Prolazimo kroz cijeli niz, u svakoj iteraciji uspoređujemo trenutni element sa susjednim. Ako je trenutni element veći od susjednog, zamijenimo ih.
- Najveći element će doći na kraj niza. Zatim koristimo jednak postupak (na skraćenom nizu bez zadnjeg elementa) da nađemo drugi najveći element i dovedemo ga na predzadnje mjesto.
- Ponavljamo prethodno opisan postupak, a potrebno je najviše n-1 prolazaka kroz niz.

Za silazno sortiranje potrebno je zamijeniti znak usporedbe.

Iz opisa algoritma jasno je odakle potječe njegov naziv, naime, najveći element (najmanji kod silaznog sortiranja) „ispliva“ na vrh poput mjehurića.

Složenost

U prvom prolazu uspoređujemo n − 1 parova susjeda, u drugom
n − 2…., pa je ukupan broj usporedbi $(n-1)+(n-2)+(n-3)+\cdots+2+1=\frac{n(n-1)}{2} \in$ O (n^2).

Broj zamjena može biti od 0 (već sortiran niz) pa sve do $\frac{(n-1)n}{2}$ (niz sortiran u suprotnom redoslijedu. Vrijedi da je $\frac{n(n-1)}{2} \in$ O (n^2).

Dakle, za vremensku složenost Bubble sorta vrijedi: T (n) $\in$ O(n^2).

Algoritam možemo poboljšati, naime, algoritam se smije zaustaviti čim on u nekom prolazu ustanovi da nema parova elemenata koje bi trebalo zamijeniti (nadalje ćemo u projektu tu verziju nazivati Bubble sort 1). Također, možemo pamtiti indeks zadnje zamjene u trenutnom prolazu, a u sljedećem prolasku gledamo niz samo do te pozicije jer je nakon nje niz već sortiran (nadalje u projektu ćemo tu verziju nazivati Bubble sort 2).
U obje varijante poboljšanja za ispravno sortirani niz bit će potreban samo jedan prolaz da algoritam završi. No, u slučaju da je za ulaz dan silazno sortiran niz koji je potrebno sortirati uzlazno, bit će i dalje potrebno svih n-1 prolaza.

Insertion sort

Insertion sort je algoritam koji rješava problem sortiranog ubacivanja elemenata u već sortirani niz. Naime, u zadan uzlazno sortirani niz elemenata potrebno je ubaciti novi element tako da novodobiveni niz i dalje bude uzlazno sortiran.

Opis algoritma:

- Za vrijeme rada algoritma, početni komad polja već je sortiran, a ostatak polja nije.
- U jednom prolasku algoritam uzima prvi element iz nesortiranog dijela, te ga stavlja na "pravo mjesto" u sortirani dio.
- Pritom dolazi do pomicanja nekih elemenata za jedno mjesto dalje.
- Jednim prolaskom duljina sortiranog dijela se poveća za 1, a duljina nesortiranog dijela se smanji za 1.
- Potrebno je n-1 prolazaka.

Složenost

Broj usporedbi u i-tom koraku algoritma varira između 1 (u slučaju da je element na pravoj poziciji) do i-1 (kad ga treba usporediti sa svim elementima s početka niza).
U najgorem slučaju (obratno sortirani niz), ukupan broj usporedbi je:
$(n-1) + (n-2) + \cdots + 2 + 1 = \frac{(n-1)\cdot n}{2} \in O(n^2)$.

Dakle, za vremensku složenost Insertion sorta vrijedi: $T(n) \in O(n^2)$.

Insertion sort algoritam može se poboljšati ako se za pronalaženje odgovarajuće pozicije elementa koristi binarno traženje (umjesto sekvencijalnog). Pomicanje elemenata za jedno mjesto je neefikasno pa možemo problem svesti na to da sortiramo više potpolja te na taj način povećamo razmak za pomake. Algoritam koji koristi taj postupak naziva se Shell sort, a za njega se pokazuje da ima manju složenost.

Iz opisa algoritama te promatranja njihove složenosti, sistematizirali smo zaključeno u tablicu.

Tablica 1 usporedba algoritama

Algoritam	Složenost u najgorem slučaju	Složenost u najboljem slučaju
Selection sort	$O(n^2)$	$O(n^2)$
Bubble sort (sve tri verzije)	$O(n^2)$, niz sortiran obrnuto	O(n), već sortiran niz, n-1 usporedba i 0 zamjena
Insertion sort	$O(n^2)$	O(n), već sortiran niz

Empirijska analiza

Usporedit ćemo kako se Selection, Insertion i Bubble sort ponašaju u praksi pomoću programa u programskom jeziku C. Generirat ćemo na slučaj način (pomoću funkcija srand i rand) nizove veličine 10^k, pri čemu je $k \in \{1,2,3,4,5\}$ te usporediti vrijeme potrebno za njihovo sortiranje uzlazno navedenim algoritmima. Za svaki k generirat ćemo 100 nizova te prikazati prosječno i ukupno vrijeme (u milisekundama) u tablici. Također, vidjet ćemo i prosječno i ukupno vrijeme potrebno za izvršavanje algoritma kod već uzlazno i silazno sortiranih nizova (posljednja 4 stupca u tablicama). Kod za izvršavanje je zbog preglednosti podijeljen u dvije datoteke. U datoteci empirijska_analiza_uzlazno_i_random.c promatraju se

nizovi generirani na slučajan način i oni koji su već sortirani uzlazno, dok se u datoteci empirijska_analiza_silazno.c promatraju nizovi koji su na početku sortirani silazno. Programe ćemo pokrenuti na računalu MacBook Air s M1 čipom i 8 GB RAM memorije.

Tablica 2 dobiveni rezultati za Selection sort

Algoritam	**k**	**Ukupno vrijeme (ms)**	**Prosječno vrijeme(ms)**	**Ukupno vrijeme (silazno, ms)**	**Prosječno vrijeme (silazno, ms)**	**Ukupno vrijeme (uzlazno, ms)**	**Prosječno vrijeme (uzlazno, ms)**
Selection sort	1	0.181	0.00181	0.122	0.00122	0.034	0.00034
	2	2.689	0.02689	2.627	0.02627	0.772	0.00772
	3	71.063	0.71063	76.401	0.76401	63.671	0.63671
	4	6359.672	63.59672	7059.386	70.59386	6296.941	62.96941
	5	637462.55	6374.6255	718571.86	7185.7186	636259.26	6362.5926

Tablica 3 dobiveni rezultati za Insertion sort

Algoritam	**k**	**Ukupno vrijeme (ms)**	**Prosječno vrijeme(ms)**	**Ukupno vrijeme (silazno, ms)**	**Prosječno vrijeme (silazno, ms)**	**Ukupno vrijeme (uzlazno, ms)**	**Prosječno vrijeme (uzlazno, ms)**
Insertion sort	1	0.132	0.00132	0.127	0.00127	0.043	0.00043
	2	1.258	0.01258	2.503	0.02503	0.054	0.00054
	3	36.427	0.36427	72.659	0.72659	0.249	0.00249
	4	3410.564	34.10564	6764.175	67.64175	2.692	0.02592
	5	350968.69	3509.6869	705697.37	7056.9737	23.12	0.2312

Tablica 4 dobiveni rezultati za Bubble sort

Algoritam	**k**	**Ukupno vrijeme (ms)**	**Prosječno vrijeme (ms)**	**Ukupno vrijeme (silazno, ms)**	**Prosječno vrijeme (silazno, ms)**	**Ukupno vrijeme (uzlazno, ms)**	**Prosječno vrijeme (uzlazno, ms)**
Bubble sort	1	0.171	0.00171	0.133	0.00133	0.046	0.00046
	2	4.166	0.04166	3.779	0.03779	0.777	0.00777
	3	113.682	1.13682	119.222	1.19222	63.733	0.63733
	4	15563.702	155.63702	11173.712	111.73712	6302.699	63.02699
	5	1971101.78	19711.0178	1158689.82	11586.8982	637994.61	6379.9461

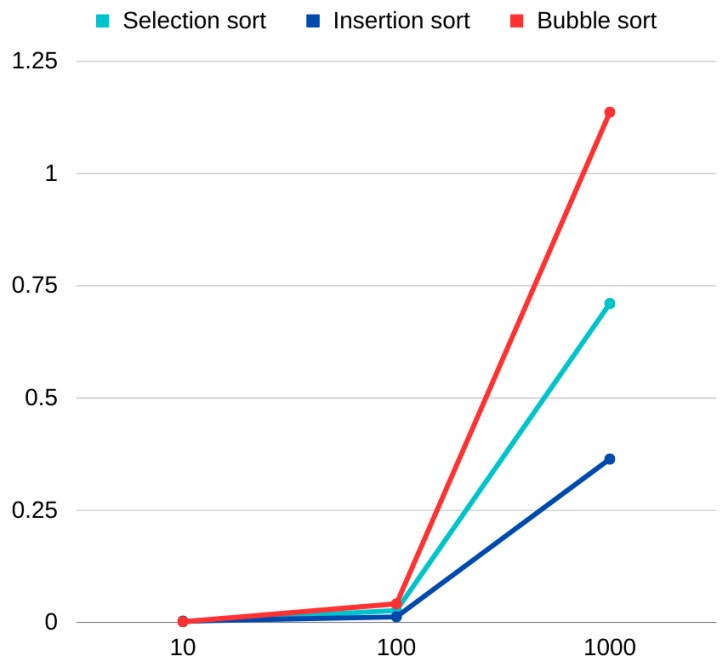
Tablica 5 dobiveni rezultati za Bubble sort 1

Algoritam	**k**	**Ukupno vrijeme (ms)**	**Prosječno vrijeme (ms)**	**Ukupno vrijeme (silazno, ms)**	**Prosječno vrijeme (silazno, ms)**	**Ukupno vrijeme (uzlazno, ms)**	**Prosječno vrijeme (uzlazno, ms)**
Bubble sort 1	1	0.179	0.00179	0.144	0.00144	0.037	0.00037
	2	3.199	0.03199	3.803	0.03803	0.122	0.00122
	3	119.993	1.19993	120.058	1.20058	0.155	0.00155
	4	16449.164	164.49164	11254.001	112.54001	2.169	0.02169
	5	1980088.004	19800.88004	1165591.21	11655.9121	13.041	0.13041

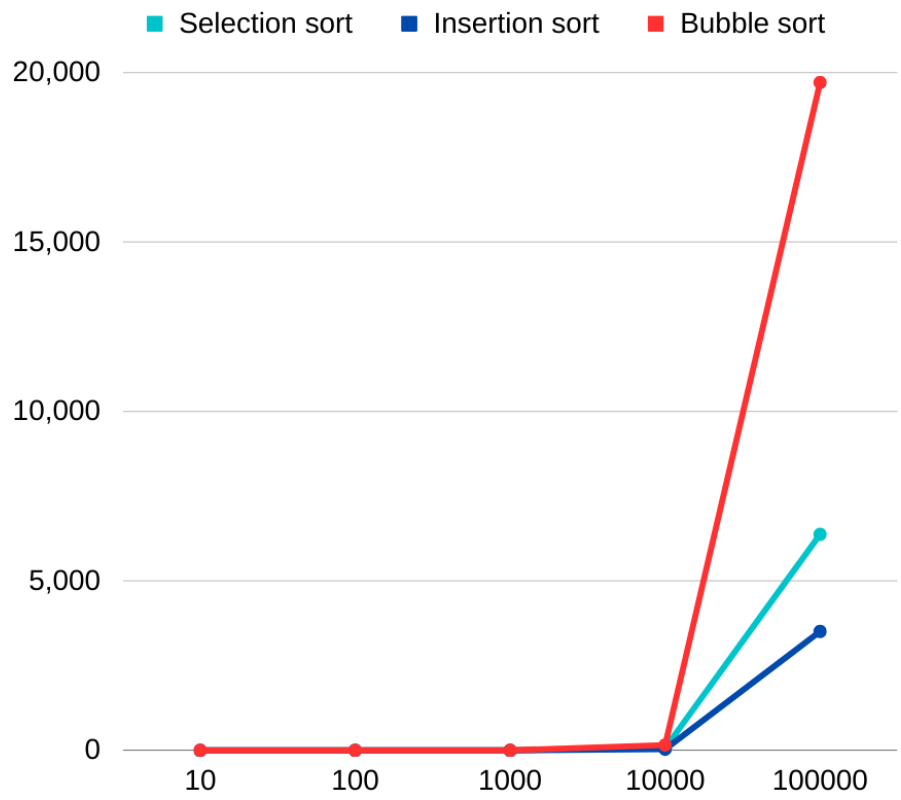
Tablica 6 dobiveni rezultati za Bubble sort 2

Algoritam	k	Ukupno vrijeme (ms)	Prosječno vrijeme (ms)	Ukupno vrijeme (silazno, ms)	Prosječno vrijeme (silazno, ms)	Ukupno vrijeme (uzlazno, ms)	Prosječno vrijeme (uzlazno, ms)
Bubble sort 2	1	0.135	0.00135	0.134	0.00134	0.031	0.00031
	2	3.667	0.03667	3.799	0.03799	0.048	0.00048
	3	119.317	1.19317	118.871	1.18871	0.154	0.00154
	4	16464.939	164.64939	11253.652	112.53652	1.526	0.01526
	5	1977758.17	19777.5817	1165347.23	11653.4723	12.98	0.1298

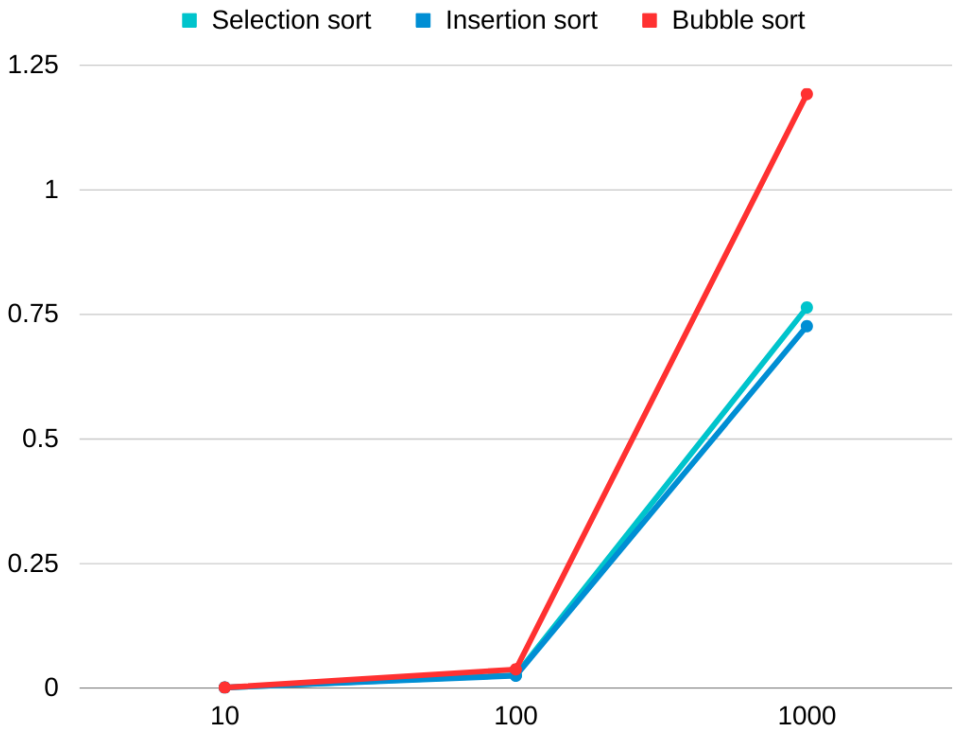
Dobivene rezultate prikazat ćemo grafički. S obzirom na to da vidimo da se sve tri verzije Bubble sort algoritma ponašaju približno jednako, zbog preglednosti ćemo promatrati samo „nepoboljšan“ Bubble sort. Na x-osi bit će veličina niza, a na y-osi prosječno vrijeme izvršavanja u milisekundama. Također, posebno ćemo gledati nizove generirane na slučajan način, nizove koji su na početku izvršavanja algoritma sortirani silazno i nizove koji su na početku sortirani uzlazno.



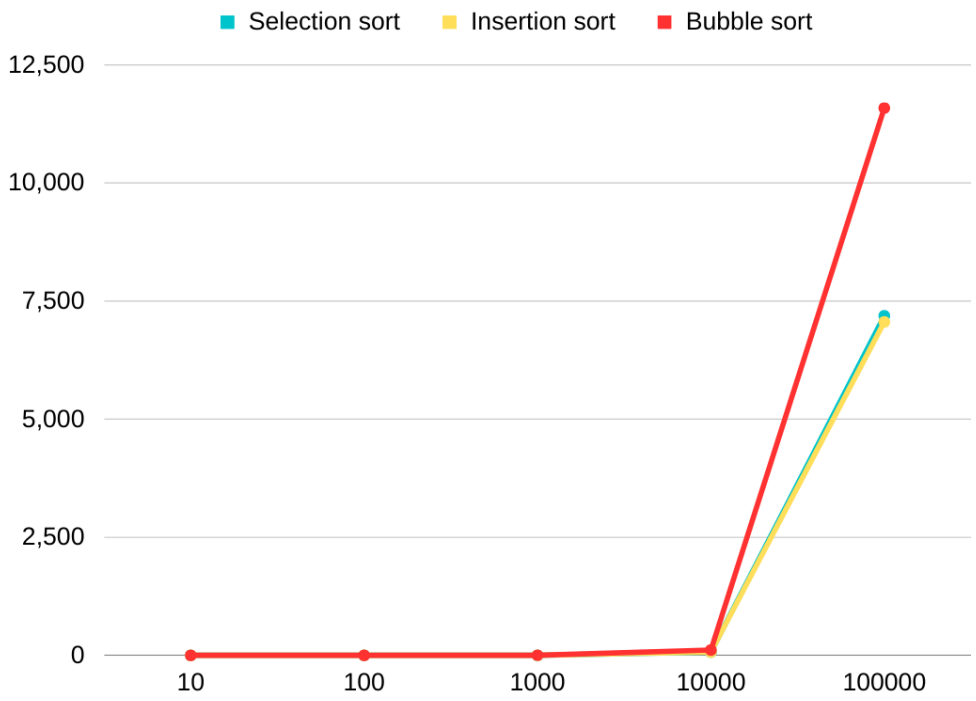
Slika 1 usporedba algoritama na nizovima generiranim na slučajan način, veličine 10-1000 elemenata



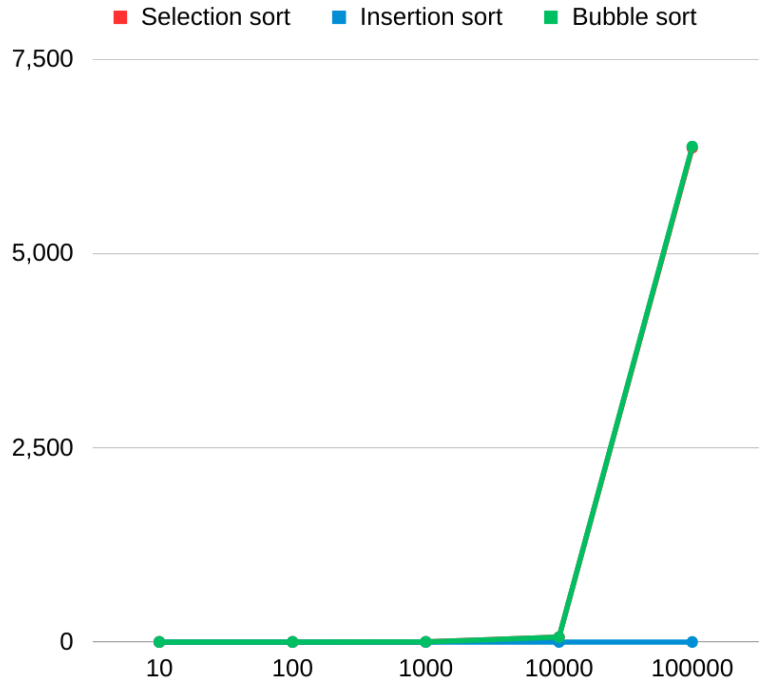
Slika 3 usporedba algoritama na nizovima generiranim na slučajan način, veličine 10-100000 elemenata



Slika 4 usporedba algoritama na silazno sortiranim nizovima, veličine 10-1000 elemenata



Slika 5 usporedba algoritama na silazno sortiranim nizovima, veličine 10-100000 elemenata



Slika 6 usporedba algoritama na uzlazno sortiranim nizovima, veličine 10-100000 elemenata

Iz rezultata iz tablice i grafova, možemo zaključiti da je Bubble sort u praksi najsporiji od promatranih algoritama, dok se Insertion sort ponaša najbolje, ali je i dalje potrebno puno

vremena za sortiranje velikih nizova. Kod uzlazno sortiranih nizova, Bubble sort 2 i Insertion sort odmah prepoznaju da je niz sortiran te algoritam završava. Iako se u nekim posebnim slučajevima te na malom broju elemenata algoritmi donekle brzo izvrše, sva tri algoritma su neefikasna na općenitim (nesortiranim) nizovima velikog broja elemenata pa je u tom slučaju bolje koristiti neke druge algoritme koji imaju manju složenost, poput Merge sorta i Quick sorta.

Literatura:

https://web.math.pmf.unizg.hr/nastava/prog1/materijali/predavanja-skracenoMM/Predavanje13.pdf [pristupljeno: 16.10.2023.]

http://web.studenti.math.pmf.unizg.hr/~manger/spa/SPA-5.pdf [pristupljeno: 17.10.2023.]

https://web.math.pmf.unizg.hr/nastava/em/em1/materijali/EM1-skripta.pdf [pristupljeno: 16.10.2023.]

https://web.math.pmf.unizg.hr/nastava/prog2/materijali/predavanja-skraceno-mm/P12PSM.pdf [pristupljeno: 18.10.2023.]

https://www.geeksforgeeks.org/bubble-sort/ [pristupljeno: 16.10.2023.]

https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/ [pristupljeno: 18.10.2023.]

https://www.canva.com/graphs/ [izrada grafova]