Matematički fakultet

Projekat iz predmeta Konstrukcija i analiza algoritama 2

Školska 2018/2019

Radix sort

Student:
Jelena Ćosić 1081/2018

Mentori:
dr Vesna Marinković
Mirko Spasić

February 3, 2019

Sadržaj

1 Algoritam

PROBLEM: Sortiranje brojeva zapisanih u dekadnom sistemu direktnim višestrukim razvrstavanjem - radix sort.

U zavisnosti od redosleda razmatranja cifara u nizu brojeva za sortiranje, postoje dve vrste algoritma za sortiranje višestrukim razvrstavanjem. To su **sortiranje direktnim višestrukim razvrstavanjem**(cifre se uzimaju zdesna ulevo, od cifre najmanje težine) i **sortiranje obratnim višestrukim razvrstavanjem**(cifre se uzimaju sleva udesno, od cifre najveće težine).

Sortiranje direktnim višestrukim razvrstavanjem (engl. radix sort) koristi pozicionu reprezentaciju brojeva i odvojeno analizira cifre na različitim pozicijama, polazeći od one najmanje težine. Takodje, ovaj aloritam se može primeniti i ako su elementi stringovi koje treba sortirati leksikografski. U tom slučaju upoređuje se znak po znak od kraja stringa ka njegovom početku, pa se dobija leksikografsko sortiranje, ali to ovde neće biti obrađeno.

Za algoritam koji ćemo ovde detaljno analizirati pretpostavka je da su elementi veliki brojevi, predstavljeni sa k cifara u sistemu sa osnovom 10 (cifre su iz opsega od 0 do 9). Algoritam se zasniva na primeni indukcije obrnutim redosledom (cifre se uzimaju zdesna ulevo) gde induktivna hipoteza glasi:

Induktivna hipoteza: Umemo da sortiramo brojeve sa manje od k cifara.

Ovaj algoritam i algoritam sortiranja obratnim višestrukim razvrstavanjem razlikuju se jedino u načinu na koji se definiše induktivna hipoteza (ideja primene induktivne hipoteze obrnutim redosledom slična je kao kod Hornerove šeme). U datom nizu brojeva sa k cifara mi najpre ignorišemo najvišu (prvu) cifru i brojeve sortiramo prema ostalim ciframa primenom indukcije. Na taj način se dobija lista brojeva sortiranih prema najnižih $k-1$ cifara, dakle sortirani brojevi bez posmatranja prve cifre. Nakon toga prolazi se još jednom kroz sve elemente i oni se razvrstavaju prema vodećoj (prvoj) cifri u 10 pregrada (od 0 do 9). Na kraju se objedinjuju sadržaji svih pregrada redom, polazeći od pregrade sa oznakom '0' pa sve do pregrade sa oznakom '9'. Potrebno je još i dokazati da su elementi na kraju sortirani po svih k cifara.

Dakle, elementi koji su svrstani u različite pregrade su u ispravnom poretku jer je najviša cifra prvog od njih veća od najviše cifre drugog od njih. Slično, ako dva elementa imaju jednake najviše cifre, onda su oni po induktivnoj hipotezi već sortirani. Najvažnije je da elementi koji su smešteni u istoj pregradi ostaju u istom redosledu kao i pre samog razvrstavanja. Ovo se postiže tako što se za smeštanje elemenata svake pregrade koriste liste, a posle toga prilikom objedinjavanja listi u svakom koraku algoritma nastaje jedna globalna lista svih elemenata i oni su sortirani posmatrajući najnižih i cifara.

Drugim rečima, algoritam direktnim višestrukim razvrstavanjem radi tako što posmatrajući najmanje značajnu cifru u svakom broju, brojeve razvrstava u različite skupove označene ciframa od 0 do 9. Nakon toga, dolazi do ponovnog objedinjavanja nastalih skupova brojeva spajajući brojeve iz pregrada od 0 do 9 u jednu listu i algoritam se nastavlja posmatrajući narednu cifru po značajanosti. Postojaće onoliko različitih koraka koliko cifara ima najveći broj u nizu koji se sortira (najviše k cifara). Algoritam se završava kada su obrađene sve cifre u svim brojevima i tada se, na kraju, dobija sortiran niz brojeva.

Na narednoj slici (Slika 1) nalazi se pseudokod algoritma za sortiranje direktnim višestrukim razvrstavanjem.

Алгоритам DVR_sort(X, n, k);
Улаз: X (низ од n целих ненегативних бројева са по k цифара).
Излаз: X (сортирани низ).
begin
    Претпостављамо да су на почетку сви елементи у глобалној листи GL
    {GL се користи због једноставности; листа се може реализовати у X}
    for $i := 0$ **to** $d - 1$ **do**
    {d је број могућих цифара; $d = 10$ у декадном случају}
        иницијализовати листу $Q[i]$ као празну листу;
    for $i := k$ **downto** 1 **do**
        while GL није празна **do** {разврставање по i-тој цифри}
            скини x из GL; {x је први елемент са листе}
            $c := i$–та цифра x; {гледано слева удесно}
            убаци x у $Q[c]$;
        for $t := 0$ **to** $d - 1$ **do** {обједињавање локалних листа}
            укључи $Q[t]$ у GL; {додавање на крај листе}
    for $i := 1$ **to** n **do** {преписивање елемената из GL у низ x}
        скини $X[i]$ из GL
end

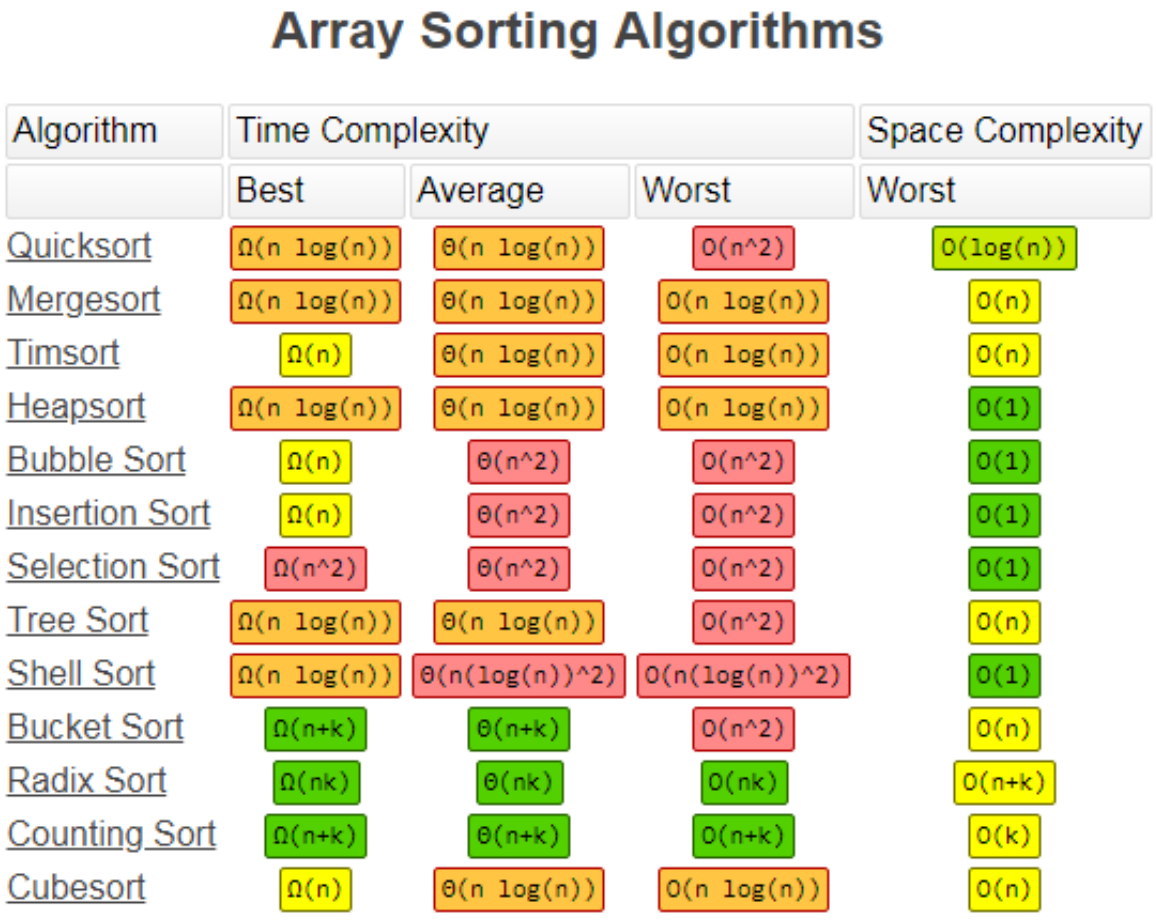
Slika 1: Sortiranje direktnim višestrukim razvrstavanjem

2 Složenost

Potrebno je n koraka za kopiranje elemenata u globalnu listu GL i d koraka za inicijalizaciju lista $Q[i]$. Glavnoj petlja algoritma, koja se izvršava k puta, svaki elemenat se vadi iz globalne i stavlja u neku od lista $Q[i]$. Na kraju se sve liste $Q[i]$ ponovo objedinjavaju u GL. Ukupna vremenska složenost algoritma je $O(k \cdot (n + d))$.

Ipak, broj cifara i opseg su cesto konstantni pa je složenost u prosečnom slučaju $O(n)$.

Sa naredne slike (Slika 2) može se zaključiti da **radix sort** je jedan od najefikasnijih algoritama sortiranja što se tiče vremenske složenosti. Ako posmatramo prostornu složenost, tu je situacija malo lošija jer je potrebno prostora za smeštanje $n + k$ elemenata neophodnih u toku sortiranja.



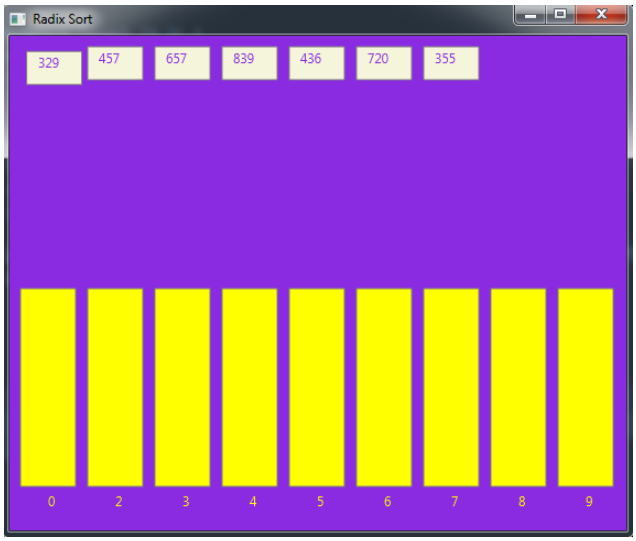
Slika 2: Složenost različitih algoritama sortiranja

Posmatrajući algoritam za sortiranje brojeva u dekadnom sistemu direktnim višestrukim razvrstavanjem dolazimo do zaključka da je broj pregrada za raspoređivanje elemenata uvek isti = 10 pregrada (oznake od 0 do 9 jer je dekadni sistem). Složenost i ovom slučaju se računa $O(k \cdot (n + d)) = O(k \cdot (n + 10)) = O(kn)$.

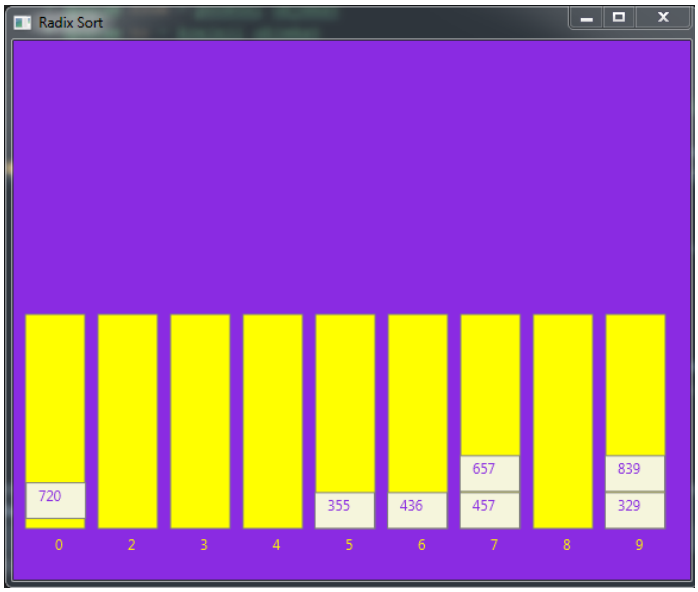
3 Primeri

Primer 3.1 *Postepeni prikaz sortiranja brojeva : 329, 457, 657, 839, 436, 720, 355 algoritmom radix sort.*

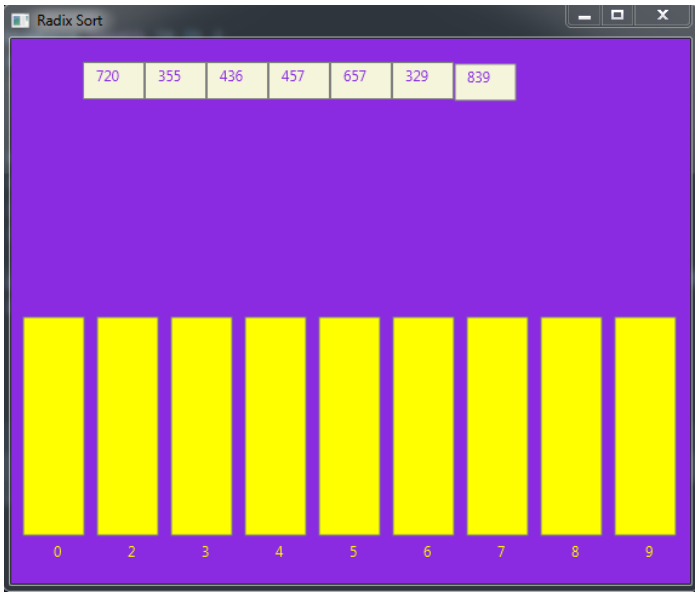
Rešenje: Na sledećim slikama su predstavljeni svi koraci prilikom sortiranja.



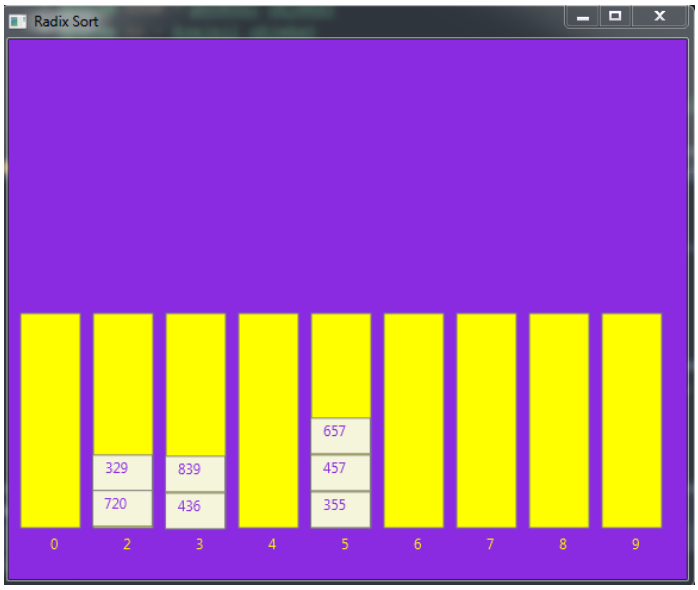
Slika 3: Početak



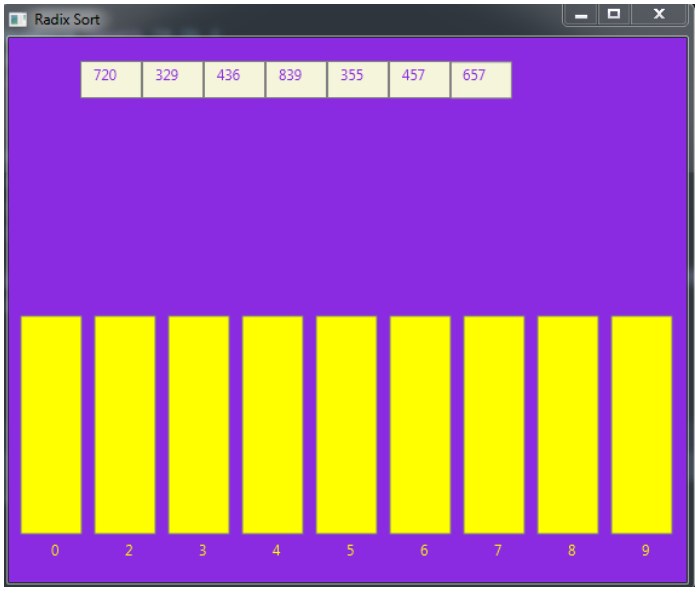
Slika 4: Korak 1



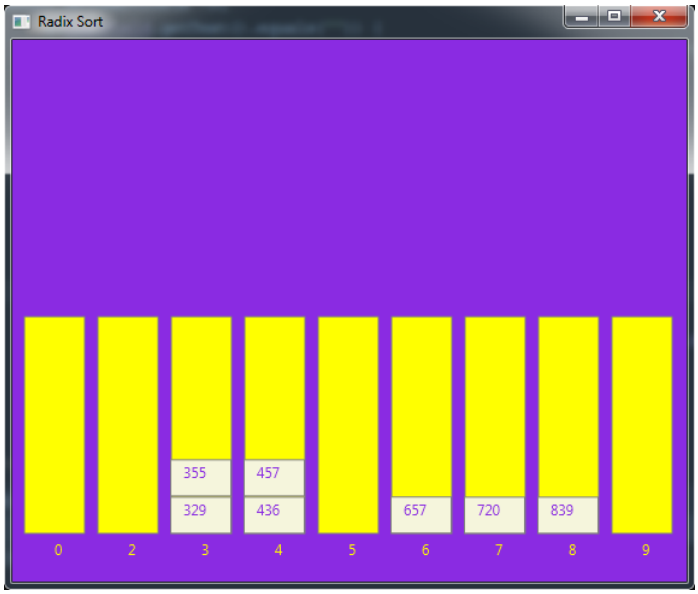
Slika 5: Korak 1



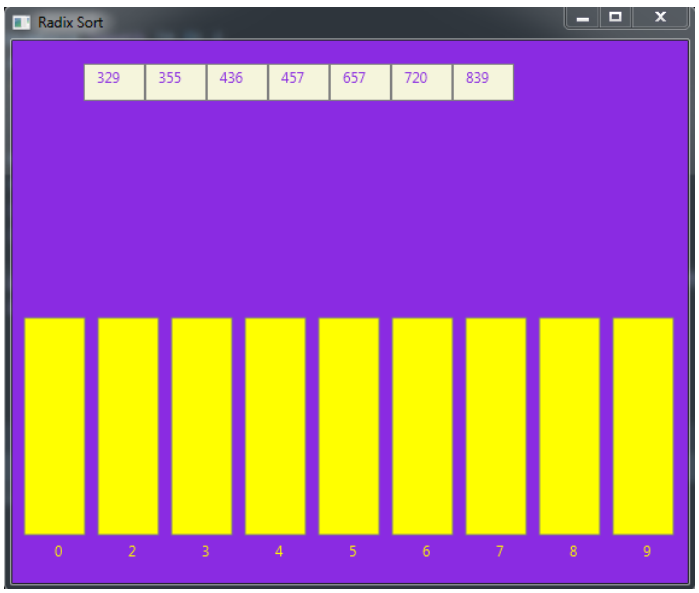
Slika 6: Korak 2



Slika 7: Korak 2



Slika 8: Korak 3



Slika 9: Korak 3

Primer 3.2 *Pokazati kako je moguće sortirati n celih brojeva iz opsega $[0, n^2 - 1]$ u vremenu $O(n)$.*

Rešenje 1: Ako bismo primenili algoritam radix sort, pri čemu su brojevi zapisani u osnovi n, onda je $d = \log_n n^2 = 2$ i $k = n$, pa je vremenska složenost $O(2 \cdot (n+n)) = O(n)$. U implementaciji ovog algoritma nije neophodno pretvarati brojeve iz dekadne osnove u osnovu broja n, već je moguće izdvojiti potrebne cifre korišćenjem operatora mod n i div n.

Rešenje 2: Svaka vrednost bi bila razmatrana kao dvocifreni broj, gde je svaka cifra iz opsega $[0, n-1]$. Ovaj algoritam bio bi složenosti $O(2(n+n)) = O(n)$.

Na narednoj slici (Slika 10) prikazan je pseudokod rešenja.

```
Algoritam: Sortiranje_u_linearnom_vremenu(niz)
Ulaz: niz (n celih brojeva iz opsega [0, n^2 − 1])
Izlaz: sortirani niz sort_niz
begin
    niz_1 := niz mod n; // operator mod se primenjuje na svaki element niza niz
    niz_2 := niz div n; // operator div se primenjuje na svaki element niza niz
    // drugi argument algoritma bucket_sort je broj pregrada
    sort_niz_1 := bucket_sort(niz_1, n);
    sort_niz_2 := bucket_sort(niz_2, n);
    objedini nizove sort_niz_1 i sort_niz_2 u niz sort_niz kao u radix_sort algoritmu;
    return sort_niz;
end
```

Slika 10: Sortiranje u linearnom vremenu

Primer 3.3 *Dato je k listi i svaka od njih sadrži n elemenata. Ključevi elemenata su celi brojevi iz opsega $[1, m]$. Pokazati kako se mogu sortirati sve liste tako da vremenska složenost u najgorem slučaju bude $O(kn + m)$.*

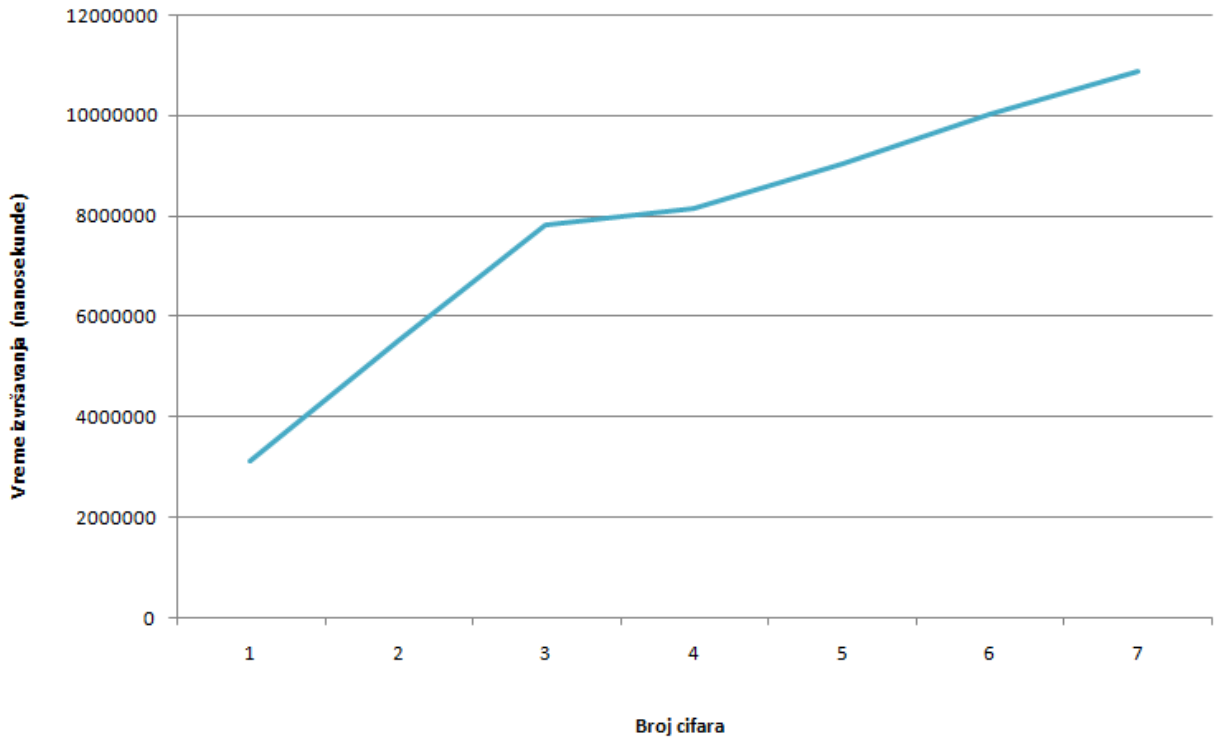
Rešenje: Prvo algoritam grube sile: Kada bismo svaku listu sortirali razvrstavanjem, onda bi vremenska složenost bila $O(k \cdot O(n + m)) = O(kn + km)$. Ova složenost je lošija od zahtevane složenosti, pa je treba popraviti.

Umesto da sortiramo k listi pojedinačno, bilo bi poželjno sortirati ih sve odjednom. Svakom elementu j iz liste i možemo pridružiti uređen par (l_i, e_j). Nad ovim parovima primenimo radix sort, koji će prvo sortirati razvrstavanjem po brojevima e_j , a zatim po rednim brojevima listi l_i. Vremenska složenost opisanog algoritma se jednostavno izračunava iz najgore složenosti algoritma radix sort $O(d' \cdot (n' + k'))$ za $d' = 2$, $n' = nk$ i $k' = m + k$, odnosno, $O(2 \cdot (nk + (m + k))) = O(k \cdot (n + 1) + m) = O(kn + m)$.

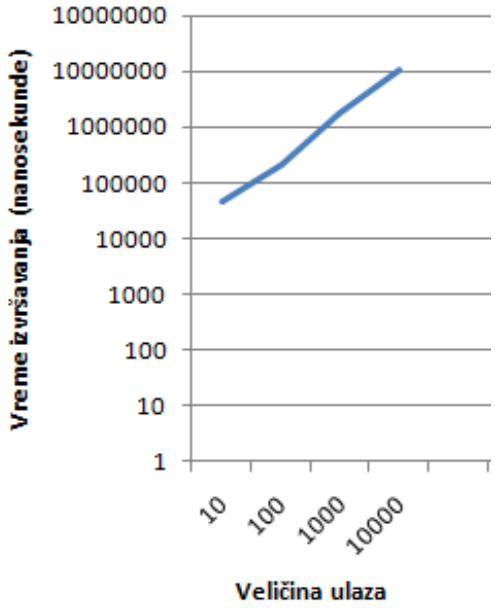
4 Rezultati

Broj cifara/ velicina ulaza	10	100	1 000	10 000
1	18 820	46 622	316 520	3 136 978
2	24 381	73 142	572 731	5 533 553
3	26 947	104 366	810 977	7 816 780
4	31 225	129 174	1 115 522	8 141 856
5	38 068	157 833	1 248 546	9 023 837
6	41 917	189 910	1 521 438	10 030 288
7	45 767	209 122	1 856 328	10 870 583

Slika 11: Rezultati testiranja (nanosekunde)



Slika 12: Zavisnost vremena izvršavanja i broja cifara



Slika 13: Zavisnost vremena izvršavanja i veličine ulaza

Na priloženoj slici (Slika 11) prikazani su rezultati testiranja rada algoritma. Sve unete vrednosti u ćelijama tabele su predstavljene u nanosekundama. Svaki red predstavlja broj cifara ulaznih brojeva, a kolone predstavljaju veličinu ulaza.

Na grafiku (Slika 12) je predstavljena zavisnost vremena izvršavanja od broja cifara. Uzet je uzorak iz tabele (Slika 11) gde je veličina ulaza 10 000 brojeva, a broj cifara se povećava od 1 do 7. U ovom slučaju $n = 10000$ pa je složenost $O(nk) = O(k \cdot 10000) = O(k)$. Pošto je veličina ulaza konstatna, sa grafika se može videti da vreme izvršavanja linearno zavisi od broja cifara.

Na drugom grafiku (Slika 13) je predstavjena zavisnost vremena izvršavanja algoritma i veličine ulaza. Ovde je uzet uzorak iz tabele (Slika 11) gde je broj cifara jednak 7, a veličina ulaza raste od 10 do 10000. Dakle, $k = 7$, pa je složenost $O(nk) = O(n \cdot 7) = O(n)$, a to se i može videti sa grafika.

Na osnovu svih navedenih test primera i grafika, moše se zaključiti da je složenost algoritma radix sort približno $O(nk)$, gde je n veličina ulaza, a k broj cifara najvećeg elementa.

Literatura

[1] Miodrag Živković, *Algoritmi*, Matematički fakultet, Beograd

[2] Miodrag Živković, Vesna Marinković, *Konstrukcija i analiza algoritama 2, Materijal sa predavanja*, Matematički fakultet, Beograd

[3] Nikola Ajzenhamer, *Konstrukcija i analiza algoritama 2, materijal sa vežbi*, Matematički fakultet, Beograd