Softverski alati u elektroenergetici - domaći zadatak br. 2

|  |  |
| --- | --- |
| **Ime i prezime:** | **Vukasin Radic** |
| **Broj indeksa:** | **PR119-2016** |

**Zadatak 1:** Napisati pseudokodovei opisati rečima princip rada SELECTION-SORT i HEAP-SORT algoritama. Opis funkcionisanja algoritama treba da se oslanja na pseudokod. Obrazložiti da li su SELECTION-SORT i HEAP-SORT **stabilni** i da li rade **u mestu.**

**Selection-sort**

//Pseudo kod Selection Sort

selection-sort(a)

for i=1 to a.length-1

for j=i+1 to a.length

if a[i]>a[j]

a[i]↔a[j]

Ovaj algoritam selektuje svaki element niza I poredi ga sa svim ostalim elementima iz niza I ukoliko je uslov ispunjen, menja im mesta. Na taj nacin se postize sortiran niz (U rastucem ili opadajucem poretku).Selection sort radi u mestu jer ne deli niz u podnizove. Takodje je I stabilan jer ukoliko su elementi iste vrednosti, ne menja im mesta.

**Heap-sort**

//Pseudo kod Heap Sort

heapsort(a)

build-max-heap(a)

for i=a.length downto 2

a[1]↔a[i]

a.heap-size = a.heap-size-1

max-heapify(a,1)

build-max-heap(a)

a.heap-size = a.length

for i= ⌊a.length/2⌋ downto 1

max-heapify(a,i)

max-heapify(a,i)

l=left(i)

r=right(i)

if i <= a.heap-size and a[l] > a[j]

largest = l

else largest = i

if r<=a.heap-size and a[r]>a[largest]

largest = r

if largest ≠ i

a[i]↔a[largest]

max-heapify(a,largest)

Ovaj algoritam predstavlja niz zamisljen kao binarno stablo. Uspostavljanjanjem regularnog hipa (deca svakog elementa su manja od njega) postizemo da se u korenu (na index-u 1) nalazi najveci element. Ideja je da se maksimalan (koji je na prvom mestu) element zameni sa poslednjim elementom niza (poslednji list stabla). Tada zanemarujemo taj poslednji element (u funkciji build-max-heap, a.heap-size=a.heap-size-1) jer je on zapravo sortiran. Ovaj postupak se potom rekurzivno ponavlja sve dok niz ne bude sortiran! Heap sort takodje, kao I Selection sort radi u mestu, sto znaci da ulazni niz ni u jednom trenutku ne deli na dva podniza. Heap-sort je stabilan sort alogritam, jer ukoliko su elementi koje treba sortirati isti, ne menja im mesta.

**Zadatak 2:** Popuniti tabelu odgovarajućim vrednostima za asimptotske notacije vremena izvršavanja odgovarajućih algoritama.

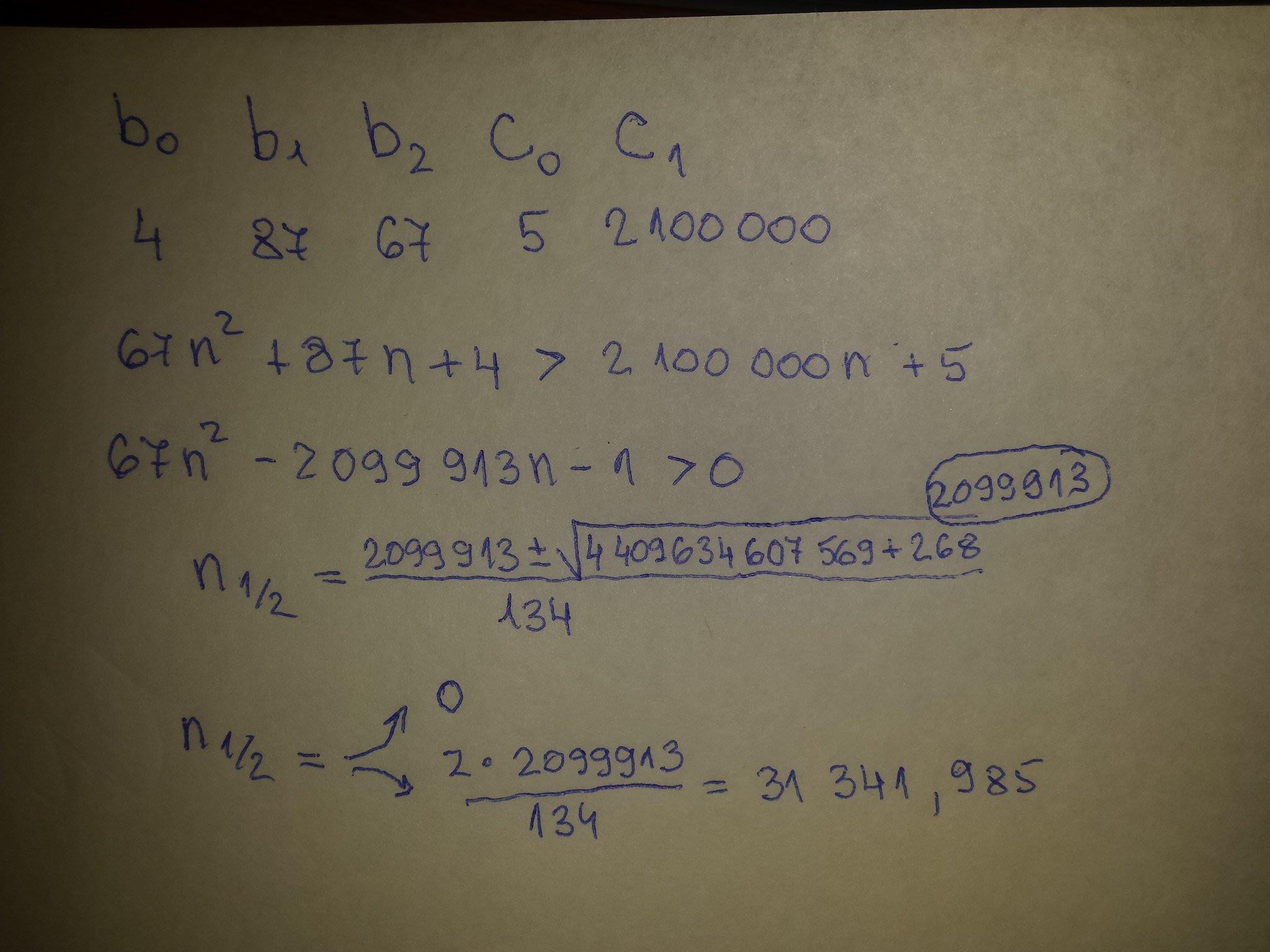
|  |  |  |
| --- | --- | --- |
| **Algoritam** | ***Najgori slučaj*** | ***Najbolji slučaj*** |
| **Selection-sort** | *O (n2)* | *Ω (n2)* |
| **Heap-sort** | *O(n•log2n)* | *Ω (*n•log2n) |

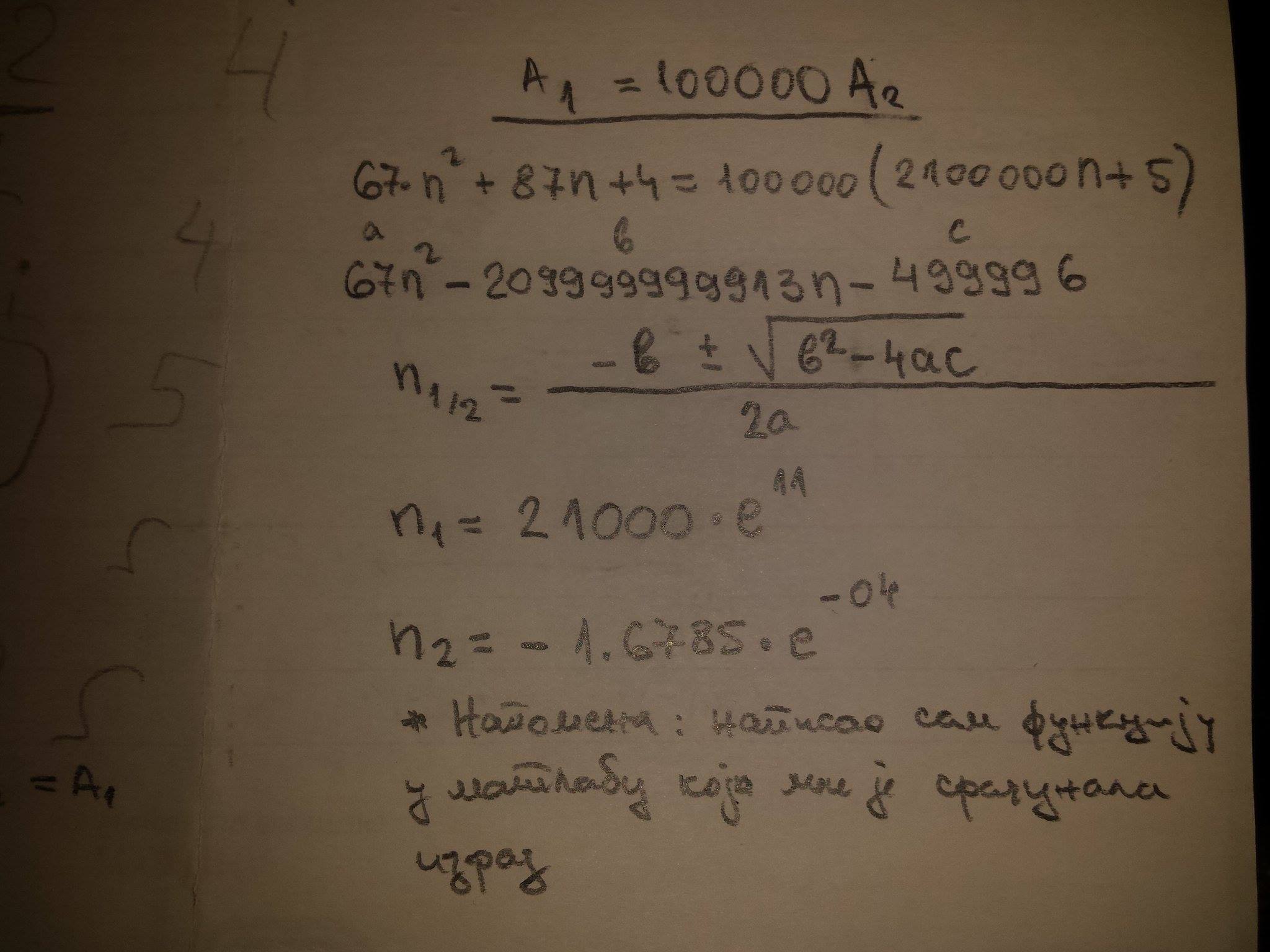
**Zadatak 3:** Neka su izrazi za vremena izvršavanja algoritama:

|  |  |
| --- | --- |
| A1: |  |
| A2: |  |

b0 = 4, b1 = 87 , b2 = 67 , c0 = 5 , c1 = 2100000.

Analitički odrediti za koje N (dužina niza) algoritam A2 postaje efikasniji od A1 algoritma. Za koje N algoritam A2 ima 100000 puta manje vreme izvršavanja od algoritma A1. (Zadatak uraditi ručno).

****

****

**Zadatak 4:** Napisati *script* fajl *zadatak4.m* u *Octave*-i koji će izvršiti sortiranje nizova slučajnih brojeva različitih dužina (*Nϵ[1,Nmax]*) pomoću SELECTION-SORT i HEAP-SORT algoritAma i zabeležiti vremena izvršavanja svakog od njih. Rezultate prikazati na jednom crtežu tako da jedna funkcija predstavlja zavisnost trajanja sortiranja od dužine niza pomoću jednog algoritma, a druga funkcija predstavlja zavisnost trajanja sortiranja od dužine niza pomoću drugog algoritma. Vizuelno utvrditi za koju dužinu niza N algoritam koji ima manju složenost postaje efikasniji.

%zadatak4.m

arrayLengths = 1:5:100;

numberOfTestsPerArrayLength = 10;

numberOfArrayLengths = length(arrayLengths);

selection\_sortRTMeanValue = zeros(1, numberOfArrayLengths);

heap\_sortRTMeanValue = zeros(1, numberOfArrayLengths);

for k =1:numberOfArrayLengths

arrayLength = arrayLengths(k);

runningTimesselection\_sort = zeros(1, numberOfTestsPerArrayLength);

runningTimesheap\_sort = zeros(1, numberOfTestsPerArrayLength);

for i = 1:numberOfTestsPerArrayLength

A = generateRandomIntegerArray(arrayLength, [-arrayLength/2 arrayLength/2]);

clearAlgorithmStats

selection\_sort(A);

runningTimesselection\_sort(i) = getTotalRunningTime;

clearAlgorithmStats

heap\_sort(A);

runningTimesheap\_sort(i) = getTotalRunningTime;

end

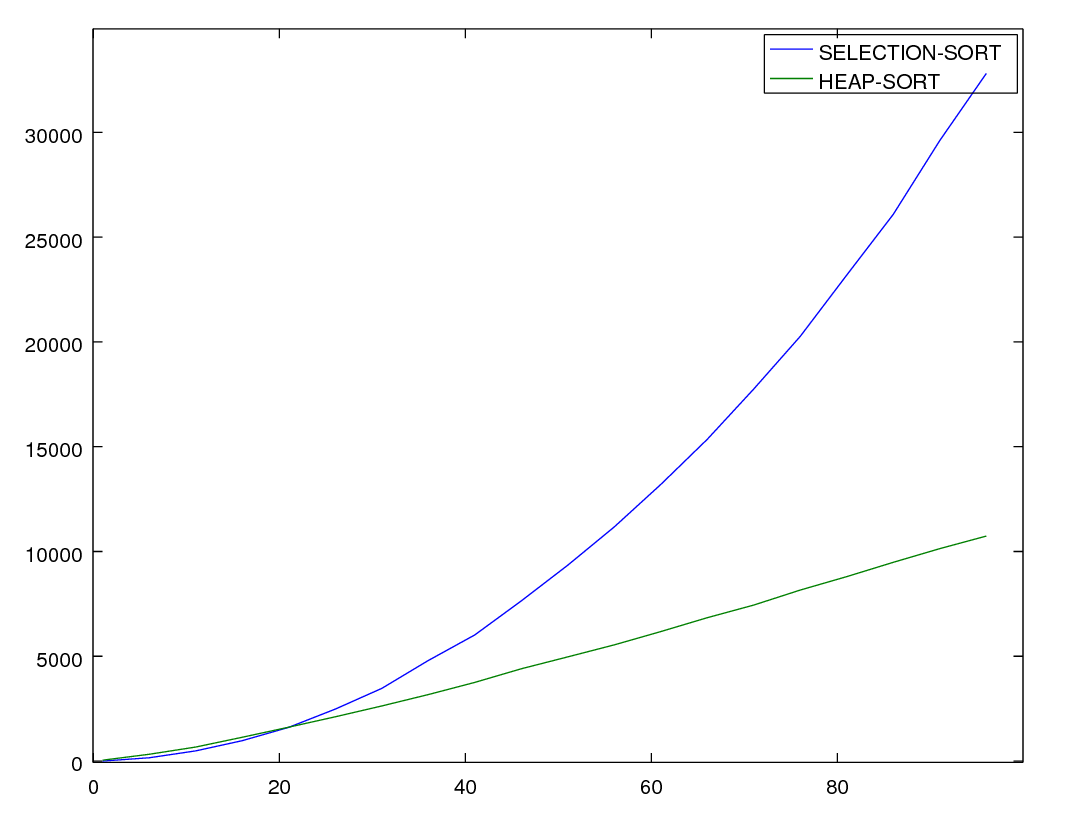
selection\_sortRTMeanValue(k) = mean(runningTimesselection\_sort);

heap\_sortRTMeanValue(k) = mean(runningTimesheap\_sort);

end

plot(arrayLengths, [selection\_sortRTMeanValue' heap\_sortRTMeanValue'])

legend('SELECTION-SORT', 'HEAP-SORT')



Slika 1 – Upoređivanje performansi SELECTION-SORT i HEAP-SORT

HEAP-SORT postaje efikasniji pri sortiranju niza dužine od oko 20 elementa (ima manje vreme izvršavanja).

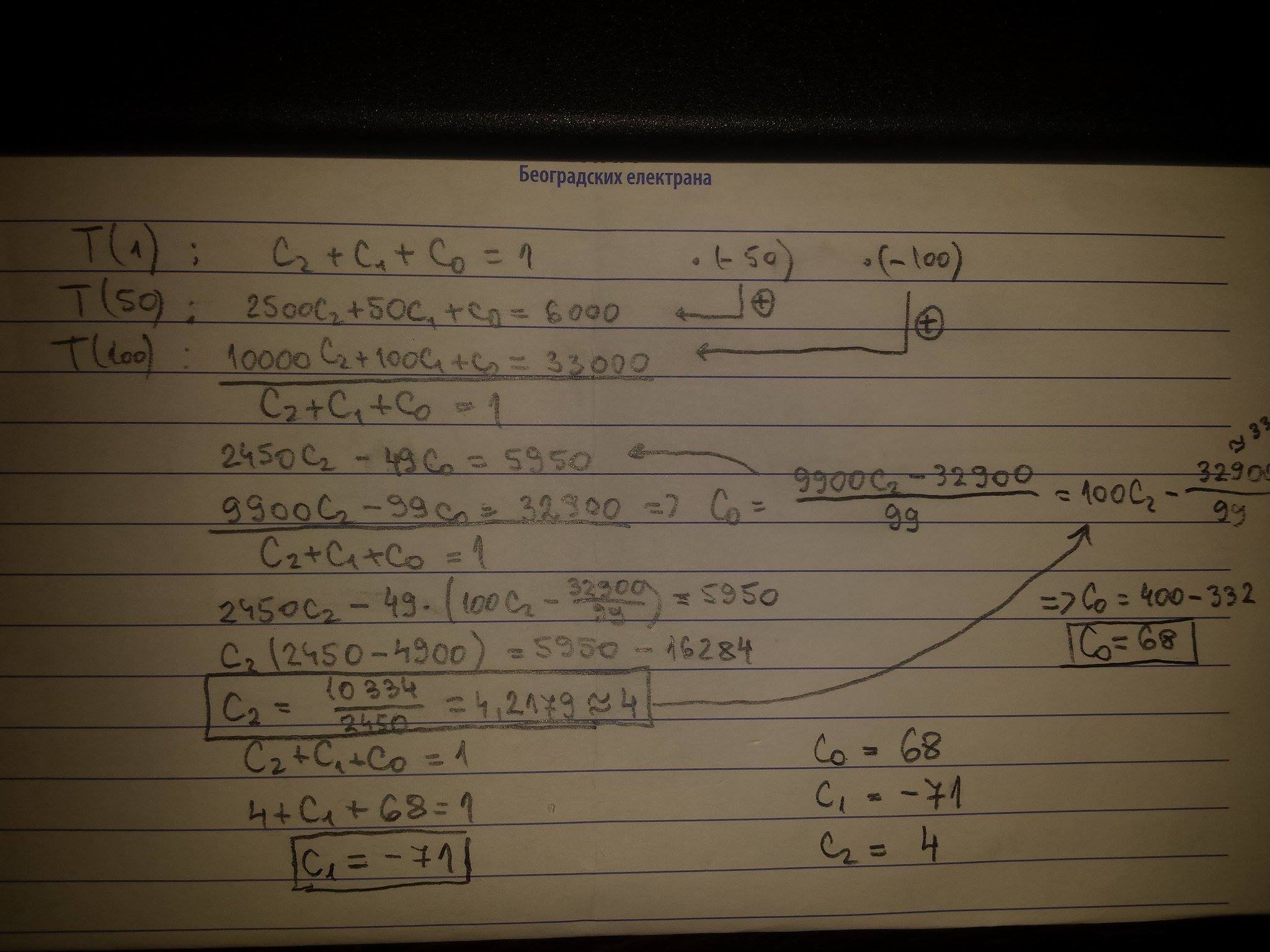
**Zadatak 5:** Napisati *script* fajl *zadatak5.m*koji treba pozvati nakonfajla *zadatak4.m* kako bi uradio sledeće:

* Za izabrane 3 dužine niza za koje je bilo izvršeno merenje trajanja sortiranja SELECTION-SORT algoritmom, odrediti trajanje sortiranja (3 dužine treba da se razlikuju što više, radi boljih rezultata iz sledećih stavki ).
* Ovaj algoritam ima složenost *θ(n2)*. Pretpostavićemo da je vreme izvršavanja za ovaj algoritam:

Na osnovu 3 izabrane tačke odrediti koeficijente polinoma . Određivanje koeficijenata uraditi i ručno (bez upotrebe računara).

* Prikazati na istom grafiku zavisnost izmerenog vremena trajanja sortiranja od dužine niza i procenjenog trajanja sortiranja od dužine niza (na osnovu polinoma )

Ovde rešiti sistem 3 linearne jednačine sa 3 nepoznate. Nepoznate su koeficijenti.



%skripta5.m

n = length(arrayLengths);

N1 = arrayLengths(1);

N2 = arrayLengths(n/2);

N3 = arrayLengths(n);

TxxN1 = selection\_sortRTMeanValue(1);

TxxN2 = selection\_sortRTMeanValue(n/2);

TxxN3 = selection\_sortRTMeanValue(n);

%Ax = b -> x=A\b

A = [N1^2 N1 1;

N2^2 N2 1;

N3^2 N3 1];

b = [TxxN1; TxxN2; TxxN3];

c = A\b;

c2 = c(1);

c1 = c(2);

c0 = c(3);

selection\_sortprocenjeno = c2\*arrayLengths.^2 + c1\*arrayLengths + c0;

plot(arrayLengths, selection\_sortRTMeanValue,'b', arrayLengths, selection\_sortprocenjeno, 'r',...

[N1 N2 N3], [TxxN1 TxxN2 TxxN3], 'mo')

legend('selection\_sort','selection\_sort - procena')

ylabel('Vreme izvrsavanja');

xlabel('N - duzina niza');

