



SOFTVERSKI ALATI U ELEKTROENERGETICI - DOMAĆI ZADATAK BR. 2

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

Zadatak 1: Napisati pseudokodove i 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 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 zamislijen kao binarno stablo. Uspostavljanjem regularnog hipa (deca svakog elementa su manja od njega) postizemo da se u korenu (na index-u 1) nalazi najveći 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 takođe, kao i Selection sort radi u mestu, što znači da ulazni niz ni u jednom trenutku ne deli na dva podniza. Heap-sort je stabilan sort algoritam, jer ukoliko su elementi koje treba sortirati isti, ne menjaju im mesta.

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

Algoritam	Najgori slučaj	Najbolji slučaj
Selection-sort	$O(n^2)$	$\Omega(n^2)$
Heap-sort	$O(n \cdot \log_2 n)$	$\Omega(n \cdot \log_2 n)$

Zadatak 3: Neka su izrazi za vremena izvršavanja algoritama:

$$A1: T_1(N) = b_2 N^2 + b_1 N + b_0$$

$$A2: T_2(N) = c_1 N + c_0$$

$$b_0 = 4, \quad b_1 = 87, \quad b_2 = 67, \quad c_0 = 5, \quad c_1 = 2100000.$$

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).

$$\begin{array}{cccccc} b_0 & b_1 & b_2 & c_0 & c_1 \\ 4 & 87 & 67 & 5 & 2100000 \end{array}$$

$$67n^2 + 87n + 4 > 2100000n + 5$$

$$67n^2 - 2099913n - 1 > 0$$

$$n_{1/2} = \frac{2099913 \pm \sqrt{4409634607569 + 268}}{134}$$

$$n_{1/2} = \frac{2099913}{134} = 31341,985$$

2 4 5 6 7

$$\underline{A_1 = 100000 A_2}$$

$$67n^2 + 87n + 4 = 100000(2100000n + 5)$$

$$67n^2 - 209999999913n - 499996$$

$$n_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$n_1 = 21000 \cdot e^{11}$$

$$n_2 = -1.6785 \cdot e^{-04}$$

* Напомена: највећа саски функцију
у математици која ми је срећетана
израз

$= A_1$

Zadatak 4: Napisati script fajl *zadatak4.m* u Octave-i koji će izvršiti sortiranje nizova slučajnih brojeva različitih dužina ($N \in [1, N_{max}]$) 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;
    endfor
endfor
```

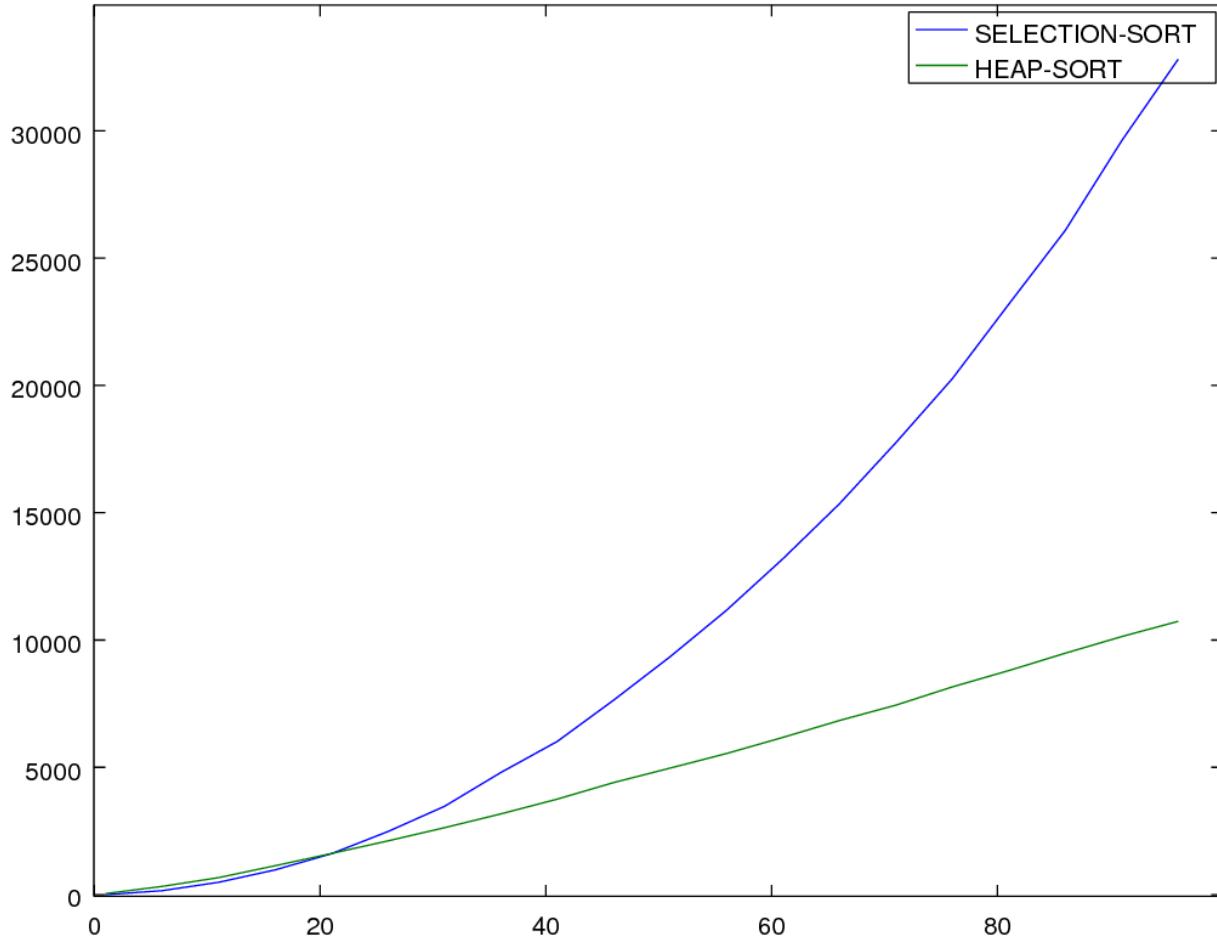


end

```
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 nakon fajla *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 $\theta(n^2)$. Prepostavićemo da je vreme izvršavanja za ovaj algoritam:

$$T_{xx}(N) = c_2 N^2 + c_1 N + c_0$$

Na osnovu 3 izabrane tačke odrediti koeficijente polinoma $T_{xx}(N)$. Određivanje koeficijenata uraditi i ručno (bez upotrebe računara).

- Prikazati na istom grafiku zavisnost izmerenog vremena trajanja sortiranja od dužine niza i procjenjenog trajanja sortiranja od dužine niza (na osnovu polinoma $T_{IS}(N)$)

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

Београдских електрана

$$\begin{aligned} T(1) : \quad C_2 + C_1 + C_0 &= 1 && \cdot (-50) \quad \cdot (-100) \\ T(50) : \quad 2500C_2 + 50C_1 + C_0 &= 6000 && \leftarrow \oplus \\ T(100) : \quad 10000C_2 + 100C_1 + C_0 &= 33000 && \leftarrow \oplus \end{aligned}$$

$$\begin{aligned} C_2 + C_1 + C_0 &= 1 \\ 2450C_2 - 49C_0 &= 5950 \\ 9900C_2 - 99C_0 &= 32900 \Rightarrow C_0 = \frac{9900C_2 - 32900}{99} = 100C_2 - \frac{32900}{99} \end{aligned}$$

$$\begin{aligned} C_2 + C_1 + C_0 &= 1 \\ 2450C_2 - 49 \cdot (100C_2 - \frac{32900}{99}) &= 5950 \\ C_2 (2450 - 4900) &= 5950 - 16284 \\ C_2 = \frac{10334}{3450} &= 4,2179 \approx 4 \end{aligned}$$

$$\begin{aligned} C_0 &= 68 \\ C_1 &= -71 \\ C_2 &= 4 \end{aligned}$$

```
%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');

```

