

Zadaci

1. Implementirati *Heap-sort* algoritam, proveriti njegovu funkcionalnost i analizirati vreme izvršenja. Pseudokodovi algoritma i pomoćnih funkcija su prikazani na slici 1.

PARENT(<i>i</i>) 1 return $\lfloor i/2 \rfloor$ LEFT(<i>i</i>) 1 return $2i$ RIGHT(<i>i</i>) 1 return $2i + 1$	MAX-HEAPIFY(<i>A, i</i>) 1 $l = \text{LEFT}(i)$ 2 $r = \text{RIGHT}(i)$ 3 if $l \leq A.\text{heap-size}$ and $A[l] > A[i]$ 4 $\text{largest} = l$ 5 else $\text{largest} = i$ 6 if $r \leq A.\text{heap-size}$ and $A[r] > A[\text{largest}]$ 7 $\text{largest} = r$ 8 if $\text{largest} \neq i$ 9 exchange $A[i]$ with $A[\text{largest}]$ 10 MAX-HEAPIFY ($A, \text{largest}$)	BUILD-MAX-HEAP(<i>A</i>) 1 $A.\text{heap-size} = A.\text{length}$ 2 for $i = \lfloor A.\text{length}/2 \rfloor$ downto 1 3 MAX-HEAPIFY (A, i) HEAPSORT(<i>A</i>) 1 BUILD-MAX-HEAP (A) 2 for $i = A.\text{length}$ downto 2 3 exchange $A[1]$ with $A[i]$ 4 $A.\text{heap-size} = A.\text{heap-size} - 1$ 5 MAX-HEAPIFY ($A, 1$)
---	---	---

Slika 1 – Pseudokodovi *Heap-sort* algoritma i pomoćnih funkcija

2. Implementirati *Bucket-sort* algoritam, proveriti njegovu funkcionalnost i analizirati vreme izvršenja. Pseudokodovi algoritma i pomoćnih funkcija su prikazani na slici 2.

BUCKET-SORT(<i>A</i>) 1 let $B[0..n-1]$ be a new array 2 $n = A.\text{length}$ 3 for $i = 0$ to $n - 1$ 4 make $B[i]$ an empty list 5 for $i = 1$ to n 6 insert $A[i]$ into list $B[\lfloor nA[i] \rfloor]$ 7 for $i = 0$ to $n - 1$ 8 sort list $B[i]$ with insertion sort 9 concatenate the lists $B[0], B[1], \dots, B[n-1]$ together in order

Slika 2 – Pseudokod *Bucket-sort* algoritma

3. Implementirati *Counting-sort* algoritam, proveriti njegovu funkcionalnost i analizirati vreme izvršenja. Pseudokodovi algoritma i pomoćnih funkcija su prikazani na slici 3.

```
COUNTING-SORT(A, B, k)
1  let C[0..k] be a new array
2  for i = 0 to k
3      C[i] = 0
4  for j = 1 to A.length
5      C[A[j]] = C[A[j]] + 1
6  // C[i] now contains the number of elements equal to i.
7  for i = 1 to k
8      C[i] = C[i] + C[i - 1]
9  // C[i] now contains the number of elements less than or equal to i.
10 for j = A.length downto 1
11     B[C[A[j]]] = A[j]
12     C[A[j]] = C[A[j]] - 1
```

Slika 3 – Pseudokod *Counting-sort* algoritma

Napomene:

- Ulazni podaci su celobrojne vrednosti organizovane u listu.
- Funkcionalnost algoritma proveriti na malom broju ulaznih podatka.
- Tokom analize vremena izvršenja algoritma koristiti različite veličine ulaznih podataka.