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)
1  return ⌊i/2⌋

LEFT(i)
1  return 2i

RIGHT(i)
1  return 2i + 1
```

```
MAX-HEAPIFY(A, i)
1   l = LEFT(i)
2   r = RIGHT(i)
3   if l ≤ A.heap-size and A[l] > A[i]
4       largest = l
5   else largest = i
6   if r ≤ A.heap-size and A[r] > A[largest]
7       largest = r
8   if largest ≠ i
9       exchange A[i] with A[largest]
10      MAX-HEAPIFY(A, largest)
```

```
BUILD-MAX-HEAP(A)
1  A.heap-size = A.length
2  for i = ⌊A.length/2⌋ downto 1
3      MAX-HEAPIFY(A, i)
```

```
HEAPSORT(A)
1  BUILD-MAX-HEAP(A)
2  for i = A.length downto 2
3      exchange A[1] with A[i]
4      A.heap-size = A.heap-size − 1
5      MAX-HEAPIFY(A, 1)
```

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

2. Implementirati *Priority-queue (max)* strukturu koristeći pomoćne funkcije koje su u pseudokodu prikazane na slici 2. Koristeći implementirane funkcije simulirati rad raspoređivača zasnovanog na prioritetu zadataka. Svaki zadatak je određen dužinom trajanja obrade koja ujedno određuje i njegov prioritet – duže vreme obrade definiše veći prioritet zadatka. Na isti način, ali koristeći minimalnu *Priority-queue (min)* strukturu, simulirati rad raspoređivača koji zadataku sa najmanjim vremenom izvršenja daje najveći prioritet izvršenja.

```
HEAP-MAXIMUM(A)
1  return A[1]
```

```
MAX-HEAP-INSERT(A, key)
1  A.heap-size = A.heap-size + 1
2  A[A.heap-size] = −∞
3  HEAP-INCREASE-KEY(A, A.heap-size, key)
```

```
HEAP-EXTRACT-MAX(A)
1  if A.heap-size < 1
2      error "heap underflow"
3  max = A[1]
4  A[1] = A[A.heap-size]
5  A.heap-size = A.heap-size − 1
6  MAX-HEAPIFY(A, 1)
7  return max
```

```
HEAP-INCREASE-KEY(A, i, key)
1  if key < A[i]
2      error "new key is smaller than current key"
3  A[i] = key
4  while i > 1 and A[PARENT(i)] < A[i]
5      exchange A[i] with A[PARENT(i)]
6      i = PARENT(i)
```

Slika 2 – Pseudokodovi pomoćnih funkcija *Priority-queue (max)* strukture

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.