Department of Computer Technology and Information Systems

CTIS264 – Computer Algorithms

Spring 2019 - 2020

Lab Guide 9 - Week 12

Instructor : Füsun YÜRÜTEN

Assistant : Leyla SEZER

OBJECTIVE:

- **Heap Tree Construction**
- **Heap Sort**
- **Morse Code Example (Tree-based algorithms)**

Q1. Apply bottom-up heap tree construction for the given list of numbers.

```
Algorithm HeapBottomUp(H[1..n])
//Constructs a heap from the elements of a given array
// by the bottom-up algorithm
//Input: An array H[1..n] of orderable items
//Output: A heap H[1..n]
for i ← ⌊n/2⌋ downto 1 do
    k ← i;   v ← H[k]
    heap ← false
    while not heap and 2 * k ≤ n do
          j ← 2 * k
          if j < n   //there are two children
             if H[j] < H[j + 1]    j ← j + 1
          if v ≥ H[j]
                heap ← true
          else H[k] ← H[j];   k ← j
     H[k] ← v
```

Output:

NOTE: Ignore the first item in all the arrays (index 0)

```
Beginning Array: [0, 2, 9, 7, 6, 5, 8, 10]
Array after heapify:  [0, 10, 9, 8, 6, 5, 2, 7]
root node: 10
2. level nodes:
[9, 8]
3. level nodes:
[6, 5, 2, 7]

Beginning Array: [0, 12, 11, 13, 5, 6, 7]
Array after heapify:  [0, 13, 11, 12, 5, 6, 7]
root node: 13
2. level nodes:
[11, 12]
3. level nodes:
[5, 6, 7]
>>>
```

Q2. Apply heapsort for the given list of numbers. To apply heapsort, you have to apply heap tree construction for the given list in the first step, then you may apply heapsort. You can use the bottom-up heap tree construction that you wrote in the first question. The program generates a list with the size of 1000 with random numbers between (1-10000) and makes the empirical analysis of the heapsort algorithm.
(The below output does not contain all the numbers in the list, just shows the part of it!!)

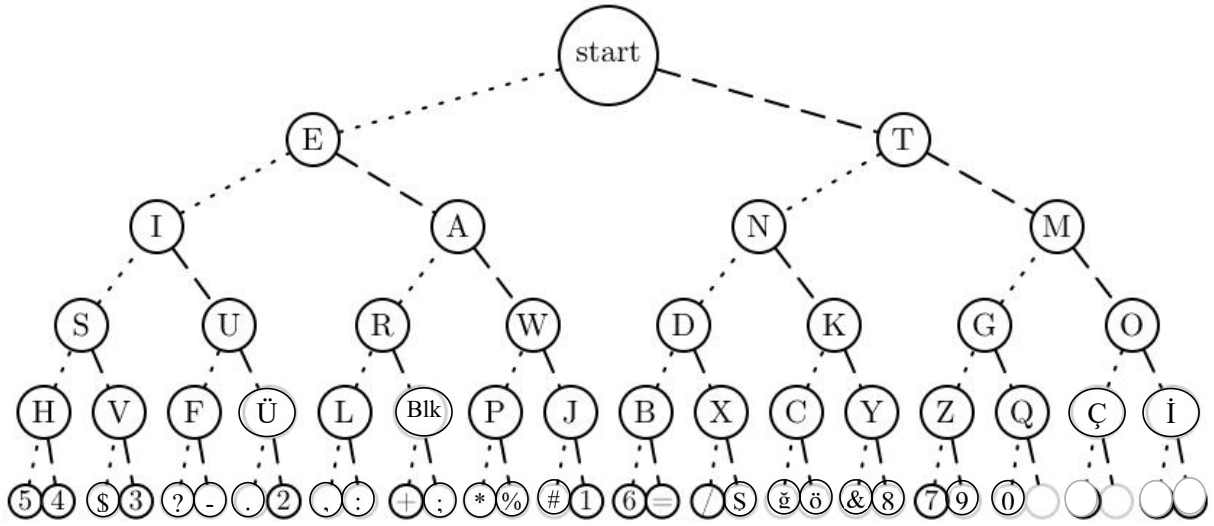
Output:

```
Initial Array: [269824, 993107, 998453, 587419, 20401, 498249, 851476, 607521,
600606, 122634, 79545]

Heapified Array:
Sorted Array: [3191, 3568, 5224, 6674, 6780, 7908, 9360, 11845, 12249, 12860,
12957]

Seconds 0.015625953674316406
```

Q3. There is a way to translate the messages with the use of the **decision tree**. A decision tree models a sequence of decisions or choices in which selections are made in stages from among multiple alternatives at each stage. The stages in the decision are represented as nodes while the branches indicate the decisions that can be made at each stage.

This idea can be used to decode the Morse code sequence. Each code sequence is unique and they are separated by blanks.

The below decision tree is in the form of a **complete and balanced binary tree** like a **binary-heap**. Starting root node does not contain any value, all the **left nodes** have a **dot (.)** and **right nodes** have a **line (-)** as a decision expansion. Some characters may have 1, some may have 2, 3, 4 or 5 notations.
For example:
A : .-
X : -..-



Note: the above Morse Codes are not completely correct Morse Codes. To make it applicable with a complete balanced binary tree, some of the missing nodes added with some special symbols. “Blk” is the blank, while you are preparing your binary tree array, please use “ ” instead of “Blk”.

Write a python code that will use this binary tree while decoding the Morse alphabet. You are asked to implement this tree like a complete **binary-heap tree array**. Find left and right child of each node indexes in this tree with the formulas that we used in a binary heap (i.e. left= 2j, right= 2j+1)

For a given Morse code like(given as a string):

.-- .. - .... .-.- -- -.-- .-.- -... . ... - .-.- .-- .. ..

This program will display the decoded morse-code string as:

WITH MY BEST WISHES

If the given Morse Code is wrong, it will display an error message like:

```
Invalid Morse Code at  the 23'th position of the code array
```

Give some more example Morse Code and display the answers within your program.
Also, your program should ask an input Morse string and should give the decoded results of them as an output.