National University of Singapore
School of Computing

IT5003: Data Structure and Algorithm

Semester I, 2019/2020

Tutorial x Lab 7 Suggested Solution
Heap and Graph

As this is the last tutorial, there is **no** lab questions. Solution will be uploaded after Saturday's PE.

1. [Heap Insertion vs Construction] Suppose we have this sequence of integers: { 8, 4, 9, 2, 1, 6, 11, 12, 3, 10, 7, 5 }, let us see the difference in building a heap via insertion vs heapify algorithm.
   a. [Using Insertion] Insert the given sequence into an empty max heap.
   b. [Using Heapify] Perform heapify on the given sequence into a single max heap.

ANS:

a.

```
          8          8       bubble up 9        9
         /          / \                        / \
        4          4   9                      4   8
```

```
      9                                 11
     / \                               /  \
    4   8       bubble up 11,         4    9
   / \ / \       insert 12           / \  / \
  2  1 6  11                        2   1 6  8
                                   /
                                  12
```

```
bubble up 12    12          bubble up 10      12
               /  \                          /   \
             11    9                       11     9
            /  \  / \                     /  \   / \
           4    1 6  8                   4   10  6  8
          / \  /                        / \  / \ /
         2  3 10                       2  3 1  7 5
```

b.

```
          8
       /     \
     4         9
   /   \     /   \
  2     1    6    11
 / \   / \  /
12  3 10  7 5
```

bubble 1 and 2 down

```
          8
       /     \
     4         9
   /   \     /   \
  12   10    6    11
 / \   / \  /
2   3 1   7 5
```

Bubble 9 down

```
          8
       /     \
     4         11
   /   \     /   \
  12   10    6     9
 / \   / \  /
2   3 1   7 5
```

Bubble 4 down

```
          8
       /     \
     12        11
   /   \     /   \
  4    10    6     9
 / \   / \  /
2   3 1   7 5
```

Bubble 8 down to get final heap

```
          12
       /     \
     10        11
   /   \     /   \
  4     8    6     9
 / \   / \  /
2   3 1   7 5
```

2. [Additional Heap Operations] Given a max-heap stored in ***items*[0..size -1]**, write two functions as follows. You can utilize the ***bubbleUp*()** and ***bubbleDown*()** functions given in the lecture notes. Also, state the time complexity for each function.

   a. ***updateKey*(p, v)**, which changes the value of the key at position **p** to **v**.

   b. ***delete*(p)**, which deletes the key at position **p** of the heap

ANS:

a.

```
def updateKey( p, newVal ):
    if p >= size or p < 0:
         return #error

    #newVal can violate heap property in either directions

    if newVal  > itemArr[p]:
        items[p] = newVal
        bubbleUp(p)
    elif newVal < itemArr[p]:
        itemArr[p] = newVal
        bubbleDown(p)
```

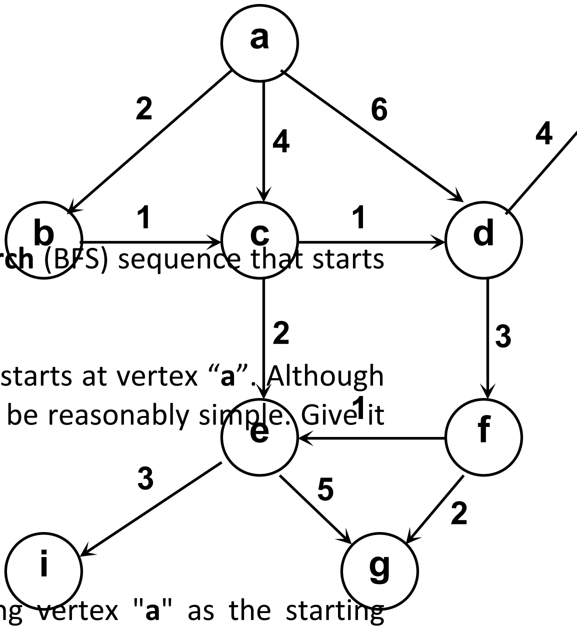
Worst Case Complexity: O(lg N)

b.

```
def delete( p ):
    if p >= size or p < 0:
         return #error

     #use the last item to replace p
     # then use part (a) to help

     lastItem = itemArr[ size – 1 ]
     size -= 1
     if size == 0:
           return     #last item deleted, nothing to do

     updateKey( p, lastItem )
```

Worst Case Complexity: O(lg N)

3. [Graph Traversals & Algorithms]



a. Given the graph above, give one possible **Breath-First Search** (BFS) sequence that starts at vertex “**a**”.

b. Give one possible **Depth-First Search** (DFS) sequence that starts at vertex “**a**”. Although DFS is not formally covered in the lecture, the idea should be reasonably simple. Give it a try?

c. Give one possible topological sort sequence.

d. Use Dijkstra’s shortest path algorithm on the graph using vertex "**a**" as the starting point. You can use a table (example given below) for your working instead of the more graphical tracing used in the lecture.

			Shortest distance from source								
Step	**v**	**S**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Init	-	-	0	∞	∞	∞	∞	∞	∞	∞	∞
1	-	[a]	0	2	4	6	∞	∞	∞	∞	∞
2	b	[a,b]	0	2	3	6	∞	∞	∞	∞	∞

Note: The “S” column is the set of fixed nodes with confirmed shortest distance (i.e. the "red" vertices"). “v” is the vertex picked in each round for updating its neighbors.

ANS:

a. **a, b, c, d, e, f, h, i, g** OR **a, d, c, b, h, f, e, g, i** OR other possible sequences.

b. **a, b, c, d, h, f, g, e, i** OR **a, b, c, e, i, g, d, h, f** OR other possible sequences.

c. **a, b, c, d, h, f, e, g, i** OR **a, b, c, d, f, h, e, g, i** OR other possible sequences.

d.

			Shortest distance from source								
Step	v	S	a	b	c	d	e	f	g	h	i
Init	-	-	0	∞	∞	∞	∞	∞	∞	∞	∞
1	-	[a]	0	2	4	6	∞	∞	∞	∞	∞
2	b	[a,b]	0	2	3	6	∞	∞	∞	∞	∞
3	c	[a,b,c]	0	2	3	4	5	∞	∞	∞	∞
4	d	[a,b,c,d]	0	2	3	4	5	7	∞	8	∞
5	e	[a,b,c,d,e]	0	2	3	4	5	7	10	8	8
6	f	[a,b,c,d,e,f]	0	2	3	4	5	7	9	8	8
7	h	[a,b,c,d,e,f,h]	0	2	3	4	5	7	9	8	8
8	i	[a,b,c,d,e,f,h,i]	0	2	3	4	5	7	9	8	8
9	g	[a,b,c,d,e,f,h,i,g]	0	2	3	4	5	7	9	8	8