
Please download the code for this exercise from http://lec.inf.ethz.ch/baug/informatik2/2015/ex/Assignment10.zip. We are not using the judge for this exercise. Please mail your TA for feedback to your solution.

1 Online Median implemented with Heaps

In the lecture you learned about Heaps and how they could be used to implement an Online Median Heap. On slide 12 of the lecture you can find a verbal description of the algorithm. Your task for the first part of this exercise is to implement exactly this algorithm. First study the code of $ArrayHeap.java$ which you already seen during the lecture. Then complete two functions in $OnlineMedian.java$. Start of with

```
public void Insert(double value) { .. }
```

Remember that slide 12 of the lecture describes how to code this up. Next you should complete the method:

```
public double Get() { ... }
```

Do not be surprised if your solution for this method is super short - its supposed to be like this. You can test your Solution by running $TestMedian.java$

2 Dijkstra's algorithm

Within the second part of this exercise we asked you to implement Dijkstra's shortest path algorithm that was also introduced in the lecture. While all parts that are needed for the implementation have been demonstrated in the lecture already, we recommend that you again study

- $Edge.java$ - implementation of an edge of the graph
- $Node.java$ - implementation of a node within the graph
- $Heap.java$ - a heap implementation as seen in previous task. Required for the algorithm

Once you familiarized yourself with the code, you should try to complete the method $ShortestPath$ inside $Graph.java$.

Since the code is a bit more involved then most code you have seen so far we basically just made "holes" in the algorithm and your task is to fill them again. Use the visual illustration of the algorithm from the lecture for help. Every ??? inside should only require a single line of code to fill it. See next page for the code with the holes. Once you completed the algorithm you can test it by running $TestShortestPath.java$.

```
Vector<Node> ShortestPath(Node S, Node E)
{
    for (int i = 0; i< nodes.size(); ++i) //we start of by traversing through all nodes
    {
        Node node = ???
        ??? //for each node update its pathlen value to Integer.MAX_VALUE
        ??? //and set its parent node to null
    }

    ??? //create a vector to store the path
    ??? //create a Heap to work with during the algorithm
    ??? //set the path length at the startnode to 0
    Node newNode = S;

    while (newNode != null && newNode != E)
    {
        Vector<Edge> edges = ??? //get all the edges from the newNode

        for (int i = 0; i<edges.size(); ++i) //for each of the edges
        {
            Edge edge = ??? //take one edge
            ??? //calculate the length the newNode plus this edge length
            ??? //get the Node that we would reach with this edge
            ??? //get the PathLength that node had so far (the one we just reached)
            if (newLength < prevLength) //if previous is longer
                            //(which is always true if we did not visit that node yet)
            {
                ??? //set its length to the newly calculated length
                ???//set its parent to the node we just came from
                if (prevLength == Integer.MAX_VALUE) // we did not see this node yet
                    ??? //therefore we would like to insert it to the heap
                else
                    ??? //otherwise update its value in the heap
            }
        }
        newNode = ??? //we then continue by taking node with
                      //shortest path out of the heap (the root)
                      //and the simply continue the loop from there
    }


    //now we backtrack from the final node
    while (newNode != null)
    {
        path.add(newNode);
        newNode = newNode.GetPathParent();
    }

    return path;
}
```