COMS21103: Problem set 3

2015/2016

Remark: Most problem sets for the next few weeks will contain at least one starred problem, which is more challenging. If any of the problems seem unclear, please post a question on the Blackboard discussion board.

1. You are given a weighted graph G with integer edge weights between 1 and 5. Argue that you can solve the (single source) shortest paths problem on G using Breadth First Search in $O(|V| + |E|)$ time.

2. Consider the following algorithm takes as input an unweighed, undirected, *possibly unconnected* graph G. What does it do? What is its time complexity?

```
MYSTERY(G) :
All vertices are initially unmarked
x = 0
For each vertex v,
    If v is unmarked
        TRAVERSE(v) (any vertices marked by TRAVERSE remain marked)
        x = x + 1
Return x
```

3. Prove the claim made in lecture that if a Binary heap is stored in an array, the operations Parent, Left and Right work correctly as defined.

4. Give an example of a graph G with at least one negative-weight edge such that Dijkstra's algorithm does not correctly output a shortest path in G. Explicitly identify where the property of having only non-negative weight edges was used in the proof of correctness of Dijkstra's algorithm.

5. Imagine we want to "fix" a graph with negative-weight edges to make Dijkstra's algorithm work, and do this by adding some large constant to the weight of each edge to make all the weights positive. Give an example which shows that this approach does not work.

6. Imagine that we want to modify our Binary Heap to support a new IncreaseKey operation (as well as Insert, DecreaseKey and ExtractMin). The IncreaseKey(x, k) operation is defined in the natural way to increase the key of element x so that $x.key = k$ (assuming $k > x.key$). How would you modify your Binary Heap to support IncreaseKey in $O(\log n)$ time?

7. ($\star$) Imagine we would like to implement Dijkstra's algorithm using a restricted priority queue which does not support the DecreaseKey operation (so it only supports Insert and ExtractMin). Modify Dijkstra's algorithm to work using such a queue. What is the time complexity of your modified algorithm?

8. ($\star$) Design a data structure which supports the operations Insert(x) and ExtractRandom(). The Insert(x) operation inserts an element x. The ExtractRandom operation should extract a (uniformly chosen) random item from the data structure. This can be seen as a random implementation of the 'bag' abstract data structure seen in lectures. Both operations should run in worst-case $O(1)$ time. You are allowed to assume that you have of use of an O(1) time random number generator.