Homework 6 Programming

(Maximum 25 points)

Due 11:59 pm Wednesday March 24, 2021

Submit your Java codes via Blackboard. Refer to the grading guidelines posted on Blackboard to understand how the submitted exercises will be graded.

1. (25) [Prim's algorithm programming] Implement the Prim's minimum spanning tree algorithm (pseudocode shown below) and run it against the graph shown below. Your program code must follow the specifications below.

- Use a minimum-oriented priority queue of nodes as studied in class. Feel free to use Java PriorityQueue class, or implement one of your own.
- Use any data structure to represent the graph. At the beginning of the program run, it should display the data structure used, e.g., "The graph is represented as an adjacency list."
- In addition to the main function, the program code must include the three key priority queue functions: the constructor BuildPQ that initializes a priority queue of all nodes in the graph, the method ExtractMin that extracts the node with the minimum priority key, and ChangeKey that changes the priority key of a node.
- At the beginning of the program run, your program must prompt for the start node number and accept it when typed by the user. Note a node number is an alphabet character.
- Before the first iteration of the While loop and at the end of each iteration of the While loop, show the content of the cut S, the content of the MST T, and the content of the priority queue Q. Specifically, show the content of S as a set of nodes in the order of insertion, the content of T as a set of edges, and the content of Q as a set of ordered pairs of a node number *v* and a priority key *π(v)*. (For ease of grading, list the priority queue elements in an increasing order of the node number.)
- An example screen output (in the first a few steps) is shown below for a start node *e*. Here, the symbol '-' denotes '∞', the priority key value '?' should be replaced by the correct value in your program output.

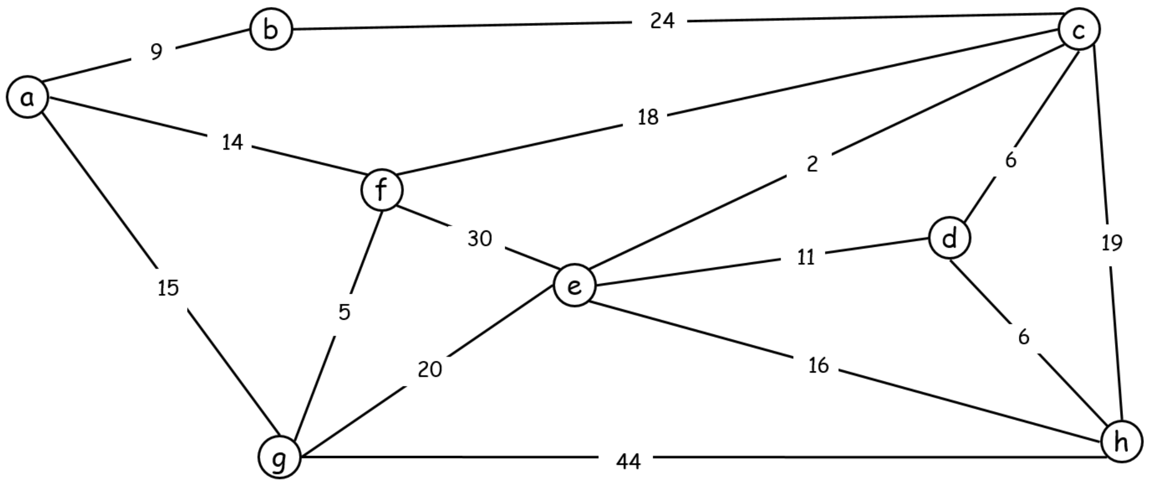
```
Before the 1st iteration:
    S = { }
    T = { }
    Q = {(a, -), (b, -), (c, -), (d, -), (e, 0), (f, -), (g, -), (h, -)}
At the end of the 1st iteration:
    S= {e}
    T = { }
    Q = {(a, ?), (b, ?), (c, ?), (d, ?), (f, ?), (g, ?), (h, ?)}
At the end of the 2nd iteration:
…
```

- Name the file of your program code as "PrimMST.java".
- Submit the source codes in one file via Blackboard. Program codes should be working and neatly organized and well commented.

Algorithm pseudocode: (from the lecture slide, with an explicit call to BuildPQ added.)

```
Prim(G, s) {
    BuildPQ to initialize Q of V with $\pi(v) \leftarrow \infty$ for each node v in V.
    Pick an arbitrary node as the start node s.
    $S \leftarrow \{\ \}$, $T \leftarrow \{\ \}$, $\pi(s) \leftarrow 0$.
    while (Q is not empty)
        v = ExtractMin(Q).
        Add v to S.
        if $v \neq s$ then add the edge (pred(v),v) to T.   // pred ≡ “predecessor”
        for each edge e=(v, w) such that $w \notin S$
            if ($c_e < \pi(w)$) {
                $\pi(w) = c_e$,  pred(w) = v.
                ChangeKey(Q, w, $\pi(w)$).
            }
        }
    endwhile
}
```

Graph:



Last modified: March 14, 2021