Please enter your name and uID below.

Name:

uID:

Collaborators, if any, and how you collaborated:

Submission notes

- Due at 11:59 pm (midnight) on Thursday, Nov 17th.
- Solutions must be typeset using one of the template files. For each problem, your answer must fit in the space provided (e.g. not spill onto the next page) *without* space-saving tricks like font/margin/line spacing changes.
- Upload a PDF version of your completed problem set to Gradescope.
- Teaching staff reserve the right to request original source/tex files during the grading process, so please retain these until an assignment has been returned.
- Please remember that for this problem set, you are allowed to collaborate in detail with your peers, as long as you cite them. However, you must write up your own solution, alone, from memory. If you do collaborate with other students in this way, you must identify the students and describe the nature of the collaboration. You are not allowed to create a group solution, and all work that you hand in must be written in your own words. Do not base your solution on any other written solution, regardless of the source.

1. (Borůvka's Edge Contraction, 30pts)

   Do problem 8a in chapter 7 of the textbook. This is copied, for your convenience, below.

   8. Minimum-spanning tree algorithms are often formulated using an operation called *edge contraction*. To contract the edge uv, we insert a new node, redirect any edge incident to u or v (except uv) to this new node, and then delete u and v. After contraction, there may be multiple parallel edges between the new node and other nodes in the graph; we remove all but the lightest edge between any two nodes.

      The three classical minimum-spanning tree algorithms described in this chapter can all be expressed cleanly in terms of contraction as follows. All three algorithms start by making a clean copy G' of the input graph G and then repeatedly contract safe edges in G'; the minimum spanning tree consists of the contracted edges.

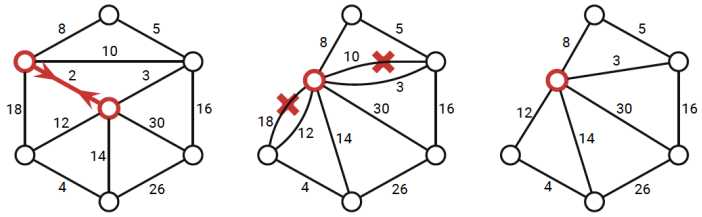

Figure 7.7. Contracting an edge and removing redundant parallel edges.

   - BORŮVKA: Mark the lightest edge leaving each vertex, contract all marked edges, and recurse.
   - JARNÍK: Repeatedly contract the lightest edge incident to some fixed root vertex.
   - KRUSKAL: Repeatedly contract the lightest edge in the graph.

Describe an algorithm to execute a single pass of Borůvka's contraction algorithm in $O(V+E)$ time. The input graph is represented in an adjacency list.

(For this problem, we are mostly interested in the mechanics of how you do contraction: how do you manage removing / adding vertices, redirecting edges, etc? You may include brief pseudocode if you'd like, but just pseudocode is not sufficient -- you must describe what your pseudocode is doing, how it works, etc.)

2. (Negative Bellman Ford, 35pts)

   Do problem 6 parts a and b in chapter 8 of the textbook. This is copied, for your convenience, below.

   6. (a) Describe and analyze a modification of Bellman-Ford that actually returns a negative cycle if any such cycle is reachable from s, or a shortest-path tree if there is no such cycle. The modified algorithm should still run in $O(VE)$ time.

      (b) Describe and analyze a modification of Bellman-Ford that computes the correct shortest path distances from s to every other vertex of the input graph, even if the graph contains negative cycles. Specifically, if any walk from s to v contains a negative cycle, your algorithm should end with $dist(v) = -\infty$; otherwise, $dist(v)$ should contain the length of the shortest path from s to v. The modified algorithm should still run in $O(VE)$ time.