CSCI 2300 — Introduction to Algorithms
Homework 3 (document version 1.1) — DUE February 21, 2020
More Graph Algorithms

- This homework is due by 11:59PM on the due date above and must be submitted in Submitty as a single PDF called `upload.pdf`.
- Just like the labs, please avoid using Google to find suggestions or solutions. You are allowed to use the textbooks and class notes as references. We strongly encourage you to use the office hours, labs/recitations, and Discussion Forum to successfully complete this assignment.
- Also, you are allowed to consult with your classmates, collaborating in line with the policies set forth in the syllabus. You must write up your own separate solution; do not copy-and-paste anything.
- Remember the goal is to use your own brain to work these problems out, which will help you develop the skills to do well on the exams and, more importantly, become a substantially better computer scientist.
- For your solutions and proofs, please write clearly and concisely, and use rigorous formal arguments and pseudocode that mimics the style used in the Dasgupta textbook. Be consistent.
- In general, if you are asked to provide an algorithm, the best thing to do is to provide a clear description in English (e.g., "Use BFS, but with the following change..." or "Run BFS two times as follows...").
- Be as concise as you can in your answers. Long answers are difficult to grade.

Warm-Up Problems (do not hand in for grading)

Work out the problems below as practice problems before going on to the next section. Note that warm-up problems might sometimes show up on an exam!

1. DPV Problem 4.1 (all parts)
2. DPV Problem 4.2 (all parts) – **(1.1)** Start from node S.
3. DPV Problem 4.3
4. DPV Problem 5.1 (all parts) – **SOLUTIONS:**
   (a) The cost of the MST is 19 (see parts (b) and (c) below).
   (b) There are two possible MSTs (see part (c) below for one MST; the other swaps edge (B, E) out for (B, F)).
   (c) Using Kruskal's algorithm on the given graph, edges are added to the MST in the following order, with cuts shown:
   - (A, E) with cut $\{A, B, C, D\}\&\{E, F, G, H\}$
   - (E, F) with cut $\{A, B, C, D, E\}\&\{F, G, H\}$
   - (B, E) with cut $\{A, E, F, G, H\}\&\{B, C, D\}$
   - (F, G) with cut $\{A, B, E\}\&\{C, D, F, G, H\}$
   - (G, H) with cut $\{A, B, E, F, G\}\&\{C, D, H\}$
   - (C, G) with cut $\{A, B, E, F, G, H\}\&\{C, D\}$
   - (D, G) with cut $\{A, B, C, E, F, G, H\}\&\{D\}$
5. DPV Problem 5.2 (all parts) – **SOLUTIONS:**
   (a) Using Prim's algorithm, vertices and edges are added in the following order, with costs shown:

Set S	Edge	A	B	C	D	E	F	G	H
$\{\}$		0	∞	∞	∞	∞	∞	∞	∞
$\{A\}$		0	1	∞	∞	4	8	∞	∞
$\{A, B\}$	(A, B)	0	1	3	∞	4	7	7	∞
$\{A, B, C\}$	(B, C)	0	1	3	6	4	7	5	∞
$\{A, B, C, G\}$	(C, G)	0	1	3	6	4	6	5	6
$\{A, B, C, G, D\}$	(G, D)	0	1	3	6	4	6	5	6
$\{A, B, C, G, D, F\}$	(G, F)	0	1	3	6	4	6	5	6
$\{A, B, C, G, D, F, H\}$	(G, H)	0	1	3	6	4	6	5	6
$\{A, B, C, G, D, F, H, E\}$	(A, E)	0	1	3	6	4	6	5	6

Required Problems (hand in for grading)

Work out the problems below and submit for grading before the deadline.

1. DPV Problem 4.11 – Be sure to explain the expected runtime of your algorithm using Big $O()$ notation.

   SOLUTIONS AND GRADING GUIDELINES:

   $\Longrightarrow$ 1 point for using an appropriate two-dimensional structure
   $\Longrightarrow$ 2 points for computing/storing the length of the shortest paths between pairs of nodes
   $\Longrightarrow$ 3 points for a correct and reasonable algorithm
   $\Longrightarrow$ 1 point for determining if the graph is acyclic
   $\Longrightarrow$ 3 points for describing the runtime using Big-$O()$ notation

   Define two-dimensional array or matrix D such that each element D_{ij} is the length of the shortest path from node i to node j in the given directed graph.

   Row i of matrix D can be computed by a run of Dijkstra's algorithm in time $O(|V|^2)$. We can therefore calculate all rows of D in $O(|V|^3)$.

   For any pair of nodes u, v, if there is a cycle, we will have two finite lengths in D_{uv} and D_{vu}. If one or both are ∞ we know there is no cycle. Further, $D_{uv} + D_{vu}$ is the shortest cycle containing both u and v.

   Given this, we can compute the minimum $D_{uv} + D_{vu}$ over all pairs of vertices u, v to find the length of the shortest cycle. If the minimum is ∞, then the graph is acyclic. This computation takes $O(|V|^2)$.

   The overall runtime is therefore $O(|V|^3)$.

2. DPV Problem 4.13 (all parts)

SOLUTIONS AND GRADING GUIDELINES:

$\Longrightarrow$ 3 points for part (a)
$\Longrightarrow$ 3 points for modifying Dijkstra's for part (b)
$\Longrightarrow$ 3 points for working algorithm for part (b)
$\Longrightarrow$ 3 points for describing the runtime for part (b)

(a) To determine if there is a feasible route from node s to node t, use either DFS or BFS and ignore edges with weights greater than L.

(b) We can modify Dijkstra's algorithm to solve this problem. More specifically, redefine the distance from s to t to be the minimum over all paths P from s to node u of the maximum length edge over all edges of P.

Compare this with the original definition of distance, i.e., the minimum over all P of the sum of edge lengths in P; this comparison suggests that by modifying the way distances are updated in Dijkstra's algorithm, we can produce a new version that solves the given problem:

```
while H is not empty:
  u = deletemin(H)
  for all edges (u,v) in E:
    if dist(v) > max( dist(u), l(u,v) ) then:
      dist(v) = max( dist(u), l(u,v) )
      prev(v) = u
      decreasekey(H, v)
```

When implemented using a binary heap, the required runtime of $O((|V| + |E|) \log |V|)$ is achieved.

3. DPV Problem 4.20 – Use an undirected graph to model this problem. For full credit, your algorithm should be more efficient than a brute force approach.

SOLUTIONS AND GRADING GUIDELINES:

$\Longrightarrow$ 6 points for describing a working non-brute force algorithm
$\Longrightarrow$ 3 points for describing the runtime

Given graph G is an undirected graph with edge weights l_e. First we make an observation. If the distance between node s and node t decreases with the addition of edge $e' = (u, v)$, the new shortest path from s to t will be the concatenation of the shortest path from s to u, edge (u, v), and the shortest path from v to t.

We can compute the length of the above path by running Dijkstra's algorithm once from s and once from t. With all of these shortest path distances, we can then compute–in constant time–the length of the shortest path from s to t that traverses e' for any $e' \in E'$.

The shortest of these paths will indicate the best edge to add.

The overall runtime of the above algorithm is $O(|V|^2 + |E'|)$.

4. DPV Problem 5.5 (all parts)

SOLUTIONS AND GRADING GUIDELINES:

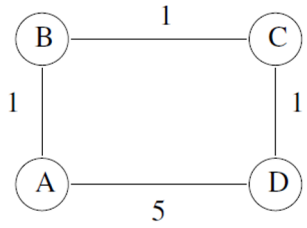
$\Longrightarrow$ 3 points for part (a)
$\Longrightarrow$ 3 points for part (b)

(a) The minimum spanning tree does not change.

Consider all spanning trees, not just the minimum. Since each spanning tree contains exactly $n-1$ edges, the cost of each tree is increased by $n-1$. Therefore, the minimum is guaranteed to again be the minimum.

(b) The shortest paths may change.

For example, consider the graph below with two paths from node A to node D. The shortest path from A to D changes if each edge weight is increased by 1.



5. DPV Problem 5.6

SOLUTIONS AND GRADING GUIDELINES:

$\Longrightarrow$ 3 points for setting up the proof correctly
$\Longrightarrow$ 3 points for a convincing and correct proof

We can prove this by contradiction.

Suppose graph G has two different minimum spanning trees T_1 and T_2. Let e be the lightest (lowest-weighted) edge present in exactly one of the trees. We know e exists in only one tree since the trees must differ by at least one edge.

Without loss of generality, say $e \in T_1$. Then adding e to T_2 produces a cycle. Further, this cycle must contain an edge e' that has a higher weight than that of e; we know this since all lighter edges are also present in T_1 where e does not produce a cycle.

Then adding e to T_2 and removing e' would yield a better spanning tree than T_2, which is a contradiction.