IITM-CS2800-2025 : Design and Analysis of Algorithms

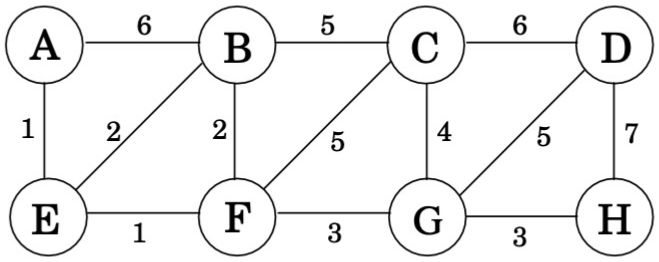
Practice Problem Sheet #8 Released On : Apr 5, 2 PM

1. Give the state of the disjoint-sets data structure after the following sequence of operations, starting from singleton sets $1, \ldots, 8$. Use path compression. In case of ties, always make the lower numbered root point to the higher numbered one.

   `union(1,2),union(3,4),union(5,6),union(7,8),union(1,4),union(6,7),union(4,5),find(1)`

2. Suppose you implement the disjoint-sets data structure using union-by-rank but not path compression[1]. Give a sequence of m union and find operations on n elements that take $\Omega(m \log n)$ time. Note that, the inverse Ackermann function is an even slower growing function that $\log^* n$.

3. Consider the following graph.



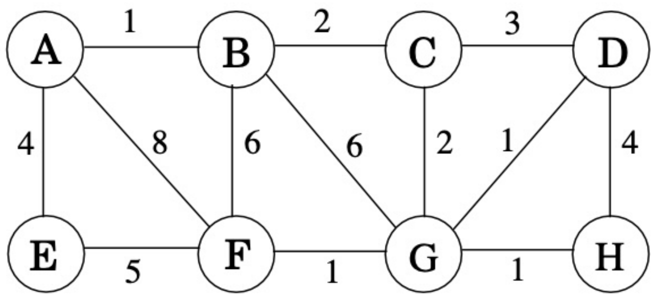
   (a) What is the cost of the minimum spanning tree in the above graph?

   (b) How many minimum spanning trees does it contain?

   (c) Suppose Kruskal's Algorithm is run on this graph. In what order will the edges by added to the MST? For each edge that gets added, specify the cut that justifies its addition as per the correctness proof that we discussed in the class.

4. Consider the following graph.

[1]The analysis of the path compression that we did in class, which gave $O((m+n)\log^* n)$ for m find operations - is not tight. In fact, Tarjan, in 1975, argued that the amortized cost per operation of union-find with path compression is $\Theta(\alpha(m,n))$, where $\alpha(m,n)$ is the inverse Ackermann function. That is, the (worst-case) cost of m operations is $\Theta(m\alpha(m,n))$

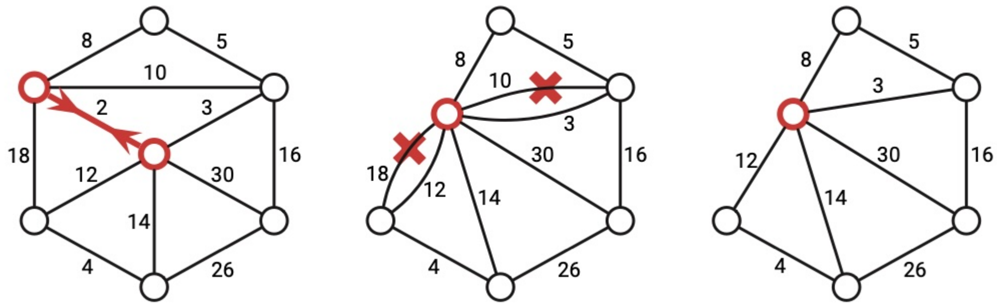


(a) Run Prim's algorithm; whenever there is a choice of nodes, always use alphabetic ordering (e.g., start from node A). Draw a table showing the intermediate values of the attachment cost array on each step.

(b) Run Kruskal's algorithm on the same graph. Show how the disjoint-sets data structure (union-find data structure that we discussed in class) looks at every intermediate stage (including the structure of the directed trees), assuming path compression is used.

5. Recall an operation called *edge contraction* of a graph G, that you may have studied in the graph theory course. To contract the edge (u, v), we insert a new node, redirect any edge incident to u or v (except (u, v)) to this new node, and then delete u and v.

   In the context of minimum weight spanning trees, 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. This is demonstrated below.



(a) Describe the Kruskal's and Prim's algorithm in terms of repeated contractions of edges.

(b) Consider the following algorithm : *Mark the lightest edge leaving each vertex, contract all marked edges, and recurse. All the contracted edges are in the MST.*

6. Let $G(V, E)$ with $|V| = m$ and $|E| = n$ be an undirected, connected graph whose weight function is $w : E \to \mathbb{R}$ and suppose that $m \geq n$ and all edge weights are distinct.

   We define the *second-best minimum spanning tree* as follows: Let $\mathcal{A}$ be the set of all minimum spanning trees, and let T be a minimum spanning tree of G. The second-best minimum spanning tree of G is a spanning tree of weight $w(T)$ where:

$$w(T) = \min_{T' \in \mathcal{A} \setminus T} w(T')$$

(a) Show that the minimum spanning tree is unique but the second best minimum spanning tree need not be unique.

(b) Let T be minimum spanning tree of G. Prove that G contains edges $(u, v) \in T$ and $(x, y) \notin T$ such that $T \setminus \{(u, v)\} \cup \{(x, y)\}$ (*Hint: the second best MST must have edges that are not present in the MST in the given graph. Can there be more than one such edges?*).

(c) Let T be a spanning tree of G and, for any two vertices $u, v \in V$, let $\max[u, v]$ denote an edge of maximum weight on the unique simple path between u and v in T. Describe an $O(n^2)$-time algorithm that, given T, computes $\max[u, v]$ for all $u, v \in V$. (Hint: Think in terms of BFS on the tree T)

(d) Use the above to give an efficient algorithm to compute the second-best minimum spanning tree of G.

7. In this question we argue several properties of minimal spanning trees: Let T be a minimum spanning tree of a graph $G(V, E)$.

(a) Let L be the sorted list of edge-weights of edges in T. Show that L must be the sorted list of edge-weights for any minimum spanning tree in T.

(b) Let V' be any subset of vertices in V. Let T' be the subgraph of T induced by V'. Similarly G' be the subgraph of G induced by V'. Question is, is T' necessarily an MST of G'? Show that it is indeed the case, if T' is connected.

(c) Show that a graph has a unique MST if, for every cut of the graph, there is a unique the minimum weight edge among the edges crossing the cut. Is the converse of this statement true?

8. In this question, we analyse the implementation details of Prim's algorithm. We recall the algorithm first: We start with a root node s and try to greedily grow a tree from s outward. At each step, we simply add the node that can be attached as cheaply as possibly to the partial tree we already have. More concretely, we maintain a set $S \subseteq V$ on which a spanning tree has been constructed so far. Initially, $S = s$. In each iteration, we grow S by one node, adding the node v that minimizes the "attachment cost" defined as : $\min_{e=(u,v):u \in S} c(e)$ and including the edge $e = (u, v)$ that achieves this minimum in the spanning tree to grow the set S.

(a) By running over all edges. By a naive implementation using arrays, show that the algorithm runs in time $O(mn)$.

(b) Use priority queues to maintain the attachment costs using `Insert`, `ExtractMin` and `ChangeKey` operations. Describe (write the precise pseudocode) for implemeting the Prim's Algorithm on graph with n nodes and m edges to run in $O(m)$ time, plus the time for n `ExtractMin`, and m `ChangeKey` operations.

(c) Using the implementation of priority queues using binary heaps (that you have seen in PDS class and implementation of Dijsktra's algorithm) show that Prim's algorithm

9. The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph $G = (V, E)$ is undirected. Do not assume that edge weights are distinct unless this is specifically stated.

   (a) If graph G has more than $n - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree.

   (b) If G has a cycle with a unique heaviest edge e, then e cannot be part of any MST.

   (c) Let e be any edge of minimum weight in G. Then e must be part of some MST.

   (d) If the lightest edge in a graph is unique, then it must be part of every MST.

   (e) If e is part of some MST of G, then it must be a lightest edge across some cut of G.

   (f) If G has a cycle with a unique lightest edge e, then e must be part of every MST.

   (g) The shortest-path tree computed by Dijkstra's algorithm is necessarily an MST.

   (h) The shortest path between two nodes is necessarily part of some MST.

   (i) Prim's algorithm works correctly when there are negative edges. node t.