Modern Algorithm Design (Monsoon 2024)
Homework 1

Deadline: 2nd September, 2024, 11:59 pm (IST)

Release Date: 19th August, 2024

1. (a) For a graph G, consider two edge-weight functions w_1 and w_2 such that

$$w_1(e) \le w_1(e') \iff w_2(e) \le w_2(e')$$

for all edges $e, e' \in E$. Show that T is an MST wrt w_1 iff it is an MST wrt w_2. (In other words, only the sorted order of the edges matters for the MST.)

   (b) Suppose graph G has integer weights in the range $\{1, \cdots W\}$, where $W \ge 2$. Let G_i be the edges of weight at most i, and κ_i be the number of components in G_i . Then show that the MST in G has weight exactly $n - W + \sum_{i=1}^{W-1} \kappa_i$.

   (c) Show that if the edge weights are all distinct, there is a unique MST.

2. (a) Show how to implement the "contract" subroutine in $O(m)$ time. This algorithm takes as input a graph $G = (V, E)$ with some edges colored blue, and outputs a new graph $G' = (V', E')$ in which vertices $v_C \in V'$ correspond to blue connected components C in G, there are no self-loops, and there is a (single) edge $e_C = (v_C, v_{C'})$ if there exists some edge between the corresponding components C, C' in G, and the weight of edge $(v_C, v_{C'})$ is given by $\min\limits_{x,y \in E: x \in C, y \in C'} w_{xy}$.

   (b) We proved in class that Boruvka reduces the number of nodes by a constant factor in each round, but what about the number of edges? Show an example of a graph with n nodes and m edges where the number of edges in G_i remains $\Omega(m)$ for $\Omega(\log n)$ rounds, even after cleaning up.

   (c) Design an $O(m \log\log n)$-time algorithm to find MST using algorithms/data-structures you have seen in the lectures.

3. We will design yet another $O(m \log\log n)$-time MST algorithm, but without using any fancy data-structures: *in Boruvka's algorithm we scan all the edges in the graph in each pass, and we should avoid this repetition.* Assume that G is a connected simple graph, and edge weights are distinct.

   (a) Suppose for each vertex, the edges adjacent to that vertex are stored in increasing order of weights. Show a slight variant of Boruvka's algorithm with runtime $O(m + n \log n)$.

   (b) (*k-partial sorting*) Given a parameter k and a list of N numbers, give an $O(N \log k)$-time algorithm that partitions this list into k groups $g_1, g_2, \cdots, g_k$ each of size at most $\lceil N/k \rceil$, so that all elements in g_i are smaller than those in g_{i+1}, for each i.

   (c) Adapt your algorithm from part (a) to handle the case where the edges adjacent to each vertex are not completely sorted but only k-partially-sorted. Ideally, your run-time should be $O(m + \frac{m}{k} \log n + n \log n)$.

   (d) Use the two parts above (setting $k = \log n$), preceded by some additional rounds of Boruvka, to give an $O(m \log\log n)$-time MST algorithm.

4. Recall that Dijkstra's algorithm computes the single-source shortest-path (SSSP) correctly for directed graphs with non-negative edge-lengths. For graphs with negative-length edges, we use typically the Bellman-Ford or Floyd-Warshall algorithms. Let us explore what happens if we use Dijkstra's algorithm instead. Assume that the graph does not have negative-length cycles.

   (a) Show an example of a graph with negative edge-lengths where Dijkstra's algorithm returns the wrong shortest-path distance from the source s. For your reference, we give Dijkstra's algorithm in algorithm 1.

Algorithm 1: Dijkstra's Algorithm

$D(s) = 0$, $D(v) = \infty$ for all $v \neq s$
run **Dijkstra-Iteration**

Algorithm 2: Dijkstra-Iteration

unmark all nodes
while *not all vertices marked* **do**
    $u \leftarrow$ unmarked vertex with least label $D(u)$
    mark u
    for *all out-edges (u,v) of* u **do**
        $D(u) \leftarrow \min\{D(v), D(u) + l(u,v)\}$
    end
end

(b) Now suppose we iterate through Dijkstra's algorithm K times (shown formally as under). Consider any node v such that the shortest-path from s to node v contains at most $K - 1$ negative-length edges. Show that the final value of the label $D(v)$ equals the length of this shortest s-v path.

Algorithm 3: K-Fold Dijkstra's Algorithm

$D(s) = 0$, $D(v) = \infty$ for all $v \neq s$
for $i = 1, 2, \cdots, K$ **do**
    run **Dijkstra-Iteration**
end