CS201c Tutorial: Graph Algorithms

A. Dijsktra's algorithm and Breadth-first search.

1. Suppose you implement Dijkstra's algorithm without a binary heap H. Instead of H, you maintain an array expected_wave_arrival[1...n] of n (future) times.

At any point of time, expected_wave_arrival[i] denotes the earliest time at which a wave will arrive at node i. If expected_wave_arrival[i]=-1, it means no wave arrival event is scheduled for node i.

Show how to implement Dijkstra's algorithm on a graph G(V,E) using this array in O(|V|^2+|E|) time.

Are there any input graphs for which this implementation is better than the binary heap based implementation discussed in class?

Note. In addition to expected_arrival[1...n], maintain a second array sender_node[1...n]. If expected_arrival[i] is not equal to -1, sender_node[i] contains the node from which the earliest wave arrival event is scheduled for node i.

2. Suppose you run Dijkstra's algorithm on a graph G(V,E) in which length of every edge is one of the W integers in the set {1, 2, ..., W}.

(a) Show that, if we use binary heap, at any point of time in the execution of Dijkstra's algorithm, there are at most W distinct wave arrival times in heap H.
(b) Use (a) to give a O(|V||W|+|E|) time implementation of Dijkstra's algorithm.
(c) Use (a) to give a O((|V|+|E|) * log(W)) time implementation of Dijkstra's algorithm.

Note. BFS is a special case of the above problem with W=1. You can use data structures other than heap, if required, for parts (b) and (c) above.

B. Depth-first search and topological sort.

3. *Finding all bridges using DFS.* Let G(V,E) be a connected, undirected graph. Recall that a **bridge** (or, cut-edge) is an edge e of G whose removal increases the number of connected components of G to 2.

We will give an O(|V|+|E|) algorithm to find all bridges in graph G using DFS.

(i) Let T be the rooted (DFS-)tree returned by executing DFS on graph G. Show that every bridge of G must occur as an edge of tree T.

(ii) Assign levels to each vertex of T as follows: the root is at level 0, root's children are at level 1, the children of root's children are at level 2, and so on.

For every vertex v in T, define:

$$A(v) = \min_{\{ u \mid (u,v) \text{ is a non-tree edge}\}} [\text{level}(u)]$$

Show how we can compute A(v), for all vertices v in E, in O(|V|+|E|) time while executing DFS on G.

(iii) For every vertex v in T, define:

$$B(v) = \min_{\{ u \mid u \text{ is a node in the subtree of } v \}} [A(v)]$$

Note. We assume that v is a node in its own subtree.

Show that given tree T and the array of A(v) values, B(v) can be computed in O(|V|) time using post order traversal of T.

(iv) Prove that an edge e=(u,v) is a bridge of G if and only if all of the following three conditions are satisfied:

(i) (u,v) is an edge of tree T,
(ii) level(u) < level(v), and
(ii) level(u) > B(v)

(v) Give an O(|V|+|E|) time algorithm which finds all bridges in graph G.

(vi) Extend the above algorithm to the case when G has more than one connected components.

C. Minimum spanning trees.

4. Let G(V,E) be a connected, undirected graph with distinct edge costs c(e), e in E.

(a) Show that G has a unique minimum spanning tree T.
(b) Let e be an edge of G not in tree T. Let C be the unique cycle in T + { e }. Show that e is the heaviest edge on this cycle.
(c) Let e be an edge of tree T. Let C1 and C2 be the two connected components of T - { e }. Show that e is the lightest edge across the cut (C1, C2).
(d) Suppose you are already given the MST T of G. Show how to recompute the minimum spanning tree of G in O(|V|) time if:
    (i) the cost of a single edge e not in tree T is increased.
    (ii) the cost of a single edge e in tree T is increased.

(iii) the cost of a single edge e not in tree T is decreased.
(iv) the cost of a single edge e in tree T is decreased.

(You can assume that edge costs remain unique after the above changes.)

(e) Show that the minimum cost edge belongs to every MST of G.
(f) Does the second minimum cost edge also belong to every MST of G? Justify your answer.
(g) Suppose we increase the cost of every edge of G by the same amount M (i.e., new cost of e is c(e) + M). Does the MST of G change? Justify your answer.

5. Obtain implementations of MST algorithms with the following running times (assume |E| >= |V|):

Reverse delete O(|E|^2)
Prim O(|E| log |V|)
Kruskal O(|E| log |V|)
Boruvka O(|E| log |V|)