CSC C78F 1999
Assignment 5

due: Monday, December 6, 7:10 PM in class.

1. Two-sided BFS.

   We can find the distance between two vertices s and t in a graph G in $O(|E| + |V|)$-time by building a BFS starting at s. In some cases we can do better by starting a BFS from both s and t simultaneously. This is what the following algorithm does. The field "color" is replaced by "status", which can have integer values from -2 to 2. 0 replaces color white (not yet visited), -1 and 1 replace gray (discovered from s, respectively t) and -2 and 2 replace black (examined from s, respectively t). The algorithm terminates when a vertex u wants to examine a vertex v of opposite sign (the two BFS meet) by returning the distance from s to t.

```
TWO-SIDED-BFS(G, s, t)
for each vertex v ∈ V[G] − {s, t}
    do status[v] ← 0
       d[v] ← ∞
       π[v] ← NIL
status[s] ← −1
d[s] ← 0
π[s] ← NIL
status[t] ← 1
d[t] ← 0
π[t] ← NIL
Q ← {s, t}
while Q ≠ ∅
    do u ← head[Q]
       for each v ∈ Adj[u]
           do if status[v] = 0
                then status[v] ← status[u]
                     d[v] ← d[u] + 1
                     π[v] ← u
                     ENQUEUE(Q, v)
              else if status[v] · status[u] < 0
                     then return d[u] + d[v] + 1
       DEQUEUE(Q)
       status[u] ← 2 status[u]
return ∞
```

   (a) Prove that the algorithm returns the correct distance and that it does so in time $O(|E|+|V|)$.

   (b) Modify this algorithm so that it returns a shortest path from s to t.

   For the rest of the question let G be a complete binary tree on $n = 2^k - 1$ nodes for $k \geq 1$.

(c) Explain why $\Theta(n)$ is the worst-case running time for determining the distance between two nodes in G using BFS. Explain why $\Theta(n)$ is the worst-case running time even if we if we only consider the loop in lines 10–18 of the algorithm on page 470 of CLR.

(d) Show that the time to execute the loop (lines 12–23) is $\Theta(\sqrt{n})$ when TWO-SIDED-BFS is used to find the distance between the root s and a leaf t.

(e) BONUS: Show that the worst-case time to execute the loop (lines 12–23) is $\Theta(n^{2/3})$ when TWO-SIDED-BFS(G, s, t) is called for arbitrary s and t.

2. Recognizing bipartite graphs.

There are many fast graph algorithms for bipartite graphs, so it is important to be able to recognize whether a given graph is bipartite.

(a) Show that a graph is bipartite if and only if it contains no odd cycle.

(b) Describe a $O(|E|)$-time algorithm BIPARTITE(G) that, given a connected graph G in terms of its adjacency-list, returns TRUE if G is bipartite and FALSE otherwise. Prove that your algorithm works and has the required running time.

(c) Building on the previous algorithm, describe an $O((|V| + |E|)\log^* |V|)$-time algorithm

BIPARTITION(G) that, given a not necessarily connected graph G returns TRUE if G is bipartite and FALSE otherwise. Furthermore, if the graph is bipartite it returns a valid bipartition by storing the vertices in one partite set in a set V_1 and the vertices in the other set in a set V_2. If the graph is not bipartite the algorithm should also return an odd cycle $C = v_1 v_2 \ldots v_{2k+1}$ as a circular linked list.

3. Spanning tree algorithms

(a) Execute Dijkstra's, Prim's and Kruskal's algorithms on the graph with vertex set $\{0, \ldots 5\}$ and adjacency matrix given below. As a source use vertex 0. If there is a tie between two edges to be chosen in one of the algorithms pick the one with the smaller sum of the keys of the nodes

$$Adj = \begin{pmatrix} 0 & 10 & 15 & 20 & 9 & 7 \\ 10 & 0 & 3 & 3 & 6 & 2 \\ 15 & 3 & 0 & 3 & 1 & 8 \\ 20 & 3 & 3 & 0 & 1 & 9 \\ 9 & 6 & 1 & 1 & 0 & 9 \\ 7 & 2 & 8 & 9 & 9 & 0 \end{pmatrix}$$

(b) Depending on how the edges are sorted, Kruskal's algorithm can return different MST's. Prove that for every MST in a given graph G there is an ordering so that Kruskal's algorithm will return this tree as its output.

(c) Dijkstra's algorithm can also be used for undirected graphs without modifications. Prove that if Dijkstra's algorithm is called on an undirected graph with all edges of weight 1, then the resulting tree is a BFS-tree.

(d) Find a connected weighted undirected graph on the minimum number of vertices, so that independent of the choice of the source and tie-breaker rules, Dijkstra's algorithm will NOT return a MST.

(e) Show that if all edge-weights are distinct, then the MST is unique.

(f) Show that a MST omits some heaviest edge from every cycle in G.