Assignment #6, CSE250

Due Date: Thur. May 5, 2011, 9:00 - 10:00am, 232 Bell Hall

THERE WILL BE NO EXTENSION

UNSUPPORTED SOLUTIONS RECEIVE NO CREDIT.

Total points: 23

1. (2+2= 4 pts) Consider the graph shown in Figure 1.

(a) Represent the graph by using the adjacency matrix representation.

(b) Represent the graph by using the adjacency list representation. (The vertices in each adjacency list should appear in numerical order).

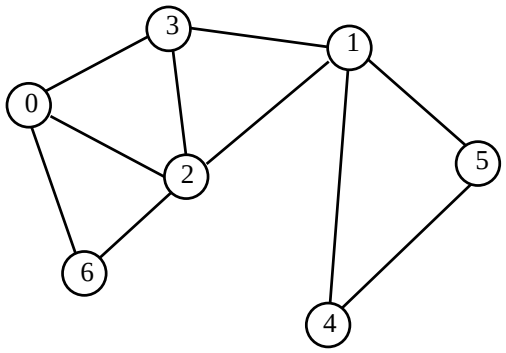


Figure 1:

2. (4 pts) Run BFS (bread-first search) algorithm on the graph shown in Fig 1, using vertex 0 as the starting vertex. (Assuming the adjacency lists are arranged in numerical order).

You should use a table similar to the Table 12.4 (page 717) to show the progress of the algorithm. Also show the BFS tree constructed by the algorithm.

3. (5 pts) Run Dijkstra's algorithm on the directed graph in Figure 2, using the vertex a as the starting vertex. Show the $d[*]$ and the $p[*]$ values after each iteration of the **while** loop. Also show the shortest path tree constructed by the algorithm.

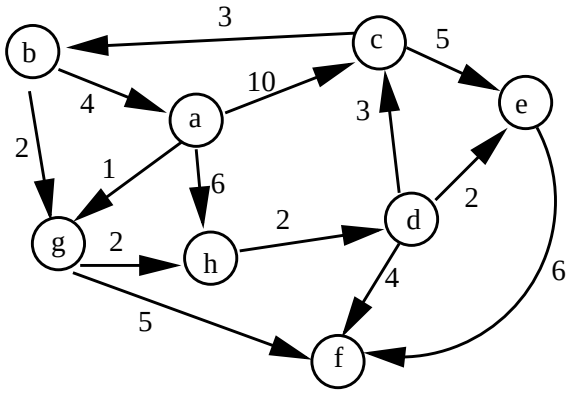
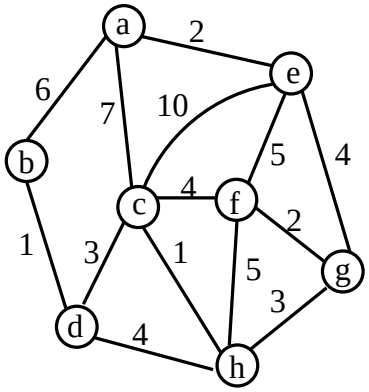


Figure 2: Graph for Dijkstra's Algorithm

4. (5 pts) Run Prim's algorithm on the graph $G = (V, E)$ shown in Figure 3. (The integer near an edge is its weight) to compute a minimum spanning tree T of G starting from the vertex a.

Use the style shown in Figure 12.28 (page 740) to show the progress of the algorithm after each step.



Prim's algorithm

Figure 3:

5. (2 pts) In both Prim's and Dijkstra algorithms, a *priority queue* data structure is used. Briefly explain why.

6. (3 pts) We have two graphs $G = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Suppose that $n = |V_1| = |V_2|$, $|E_1| = 4n$ and $|E_2| = n^2/4$. We want to ran certain algorithms on G_1 and G_2. The algorithm will call the function `get_edge(i,j)` (if $\{i, j\}$ is an edge of the graph, the function retrieves the information associated with the edge, if not, the function returns nil), and the function *neighbor-listing(i)* (which returns a list of the neighbors of the vertex i). We need to decide which graph representation (adjacency list or adjacency matrix) should be used.

1. If the space requirement is our main concern, which representation should be used for G_1? and G_2? Why?

2. Suppose that the algorithm will call the function *find-edge(i,j)* $O(n)$ times, and the function *neighbor-listing(i)* $O(\log n)$ times. If the run time is our main concern, which representation should be used for G_1? and G_2? Why?

3. Suppose that the algorithm will call the function *find-edge(i,j)* $O(1)$ times, and the function *neighbor-listing(i)* $O(n)$ times. If the run time is our main concern, which representation should be used for G_1? and G_2? Why?