COMP9024 19T0

# Week 05 Problem Set

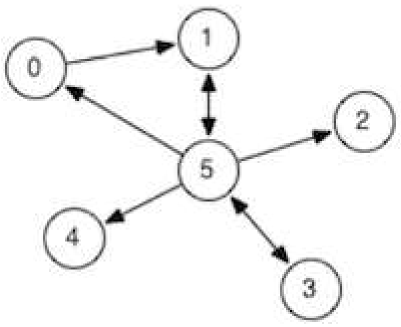
Minimum Spanning Trees, Shortest Paths

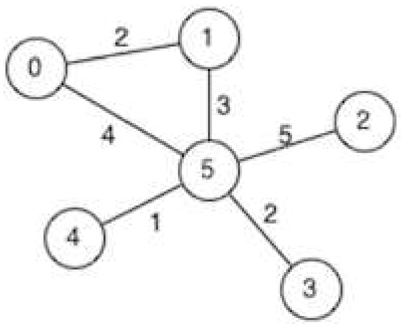
Data Structures and Algorithms

[Show with no answers] [Show with all answers]

1. (Graph representations)

   For each of the following graphs:





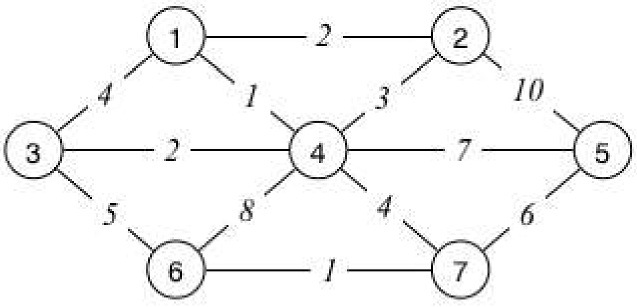
   Show the concrete data structures if the graph was implemented via:

   a. adjacency matrix representation (assume full V×V matrix)
   b. adjacency list representation (if non-directional, include both (v,w) and (w,v))

   [show answer]

2. (MST – Kruskal's algorithm)

   a. Identify a minimum spanning tree in the following graph (without applying any of the algorithms from the lecture):



   What approach did you use in determining the MST?

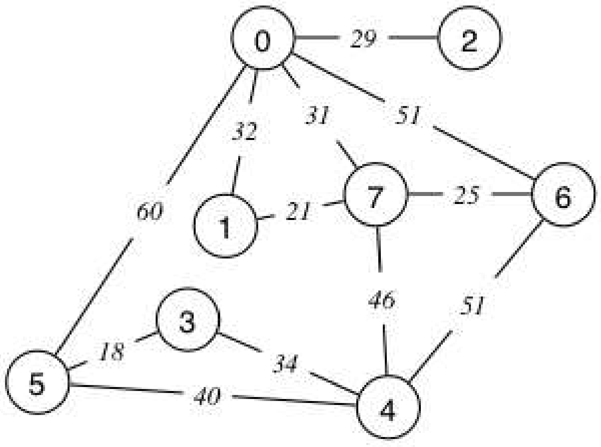
   b. Show how Kruskal's algorithm would construct the MST for the above graph. How many edges do you have to consider?

   c. For a graph *G=(V,E)*, what is the least number of edges that might need to be considered by Kruskal's algorithm, and what is the most number of edges? Add one vertex and edge to the above graph to force Kruskal's algorithm to the worst case.

   [show answer]

3. (MST – Prim's algorithm)

   Trace the execution of Prim's algorithm to compute a minimum spanning tree on the following graph:



Choose a random vertex to start with. Draw the resulting minimum spanning tree.

[show answer]

4. (Priority queue for Dijkstra's algorithm)

   Assume that a priority queue for Dijkstra's algorithm is represented by a global variable `PQueue` of the following structure:

```
#define MAX_NODES 1000
typedef struct {
   Vertex item[MAX_NODES];  // array of vertices currently in queue
   int    length;           // #values currently stored in item[] array
} PQueueT;

PQueueT PQueue;
```

   Further assume that vertices are stored in the `item[]` array in the priority queue in *no* particular order. Rather, the `item[]` array acts like a set, and the priority is determined by reference to a `priority[0..nV-1]` array.

   If the priority queue `PQueue` is initialised as follows:

```
void PQueueInit() {
   PQueue.length = 0;
}
```

   then give the implementation of the following functions in C:

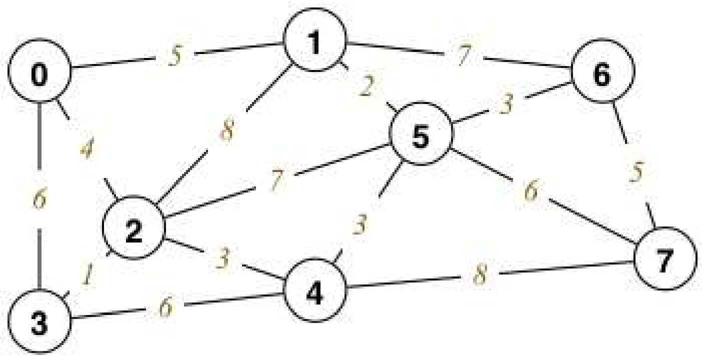
```
// insert vertex v into priority queue PQueue
// no effect if v is already in the queue
void joinPQueue(Vertex v) { ... }

// remove the highest priority vertex from PQueue
// remember: highest priority = lowest value priority[v]
// returns the removed vertex
Vertex leavePQueue(int priority[]) { ... }
```

   [show answer]

5. (Dijkstra's algorithm)

   a. Trace the execution of Dijkstra's algorithm on the following graph to compute the minimum distances from source node 0 to all other vertices:



Show the values of `vSet`, `dist[]` and `pred[]` after each iteration.

b. Implement Dijkstra's algorithm in C using your priority queue functions for Exercise 4 and the Weighted Graph ADT (`WGraph.h`, `WGraph.c`) from the lecture. Your program should

- prompt the user for the number of nodes in a graph,
- prompt the user for a source node,
- build a weighted undirected graph from user input,
- compute and output the distance and a shortest path from the source to every vertex of the graph.

An example of the program executing is shown below. The input graph consists of a simple triangle along with a fourth, disconnected node.

```
prompt$ ./dijkstra
Enter the number of vertices: 4
Enter the source node: 0
Enter an edge (from): 0
Enter an edge (to): 1
Enter the weight: 42
Enter an edge (from): 0
Enter an edge (to): 2
Enter the weight: 25
Enter an edge (from): 1
Enter an edge (to): 2
Enter the weight: 14
Enter an edge (from): done
Finished.
0: distance = 0, shortest path: 0
1: distance = 39, shortest path: 0-2-1
2: distance = 25, shortest path: 0-2
3: no path
```

Note that any non-numeric data can be used to 'finish' the interaction. Test your algorithm with the graph from Exercise 5a.

*We have created a script that can automatically test your program. To run this test you can execute the* `dryrun` *program that corresponds to the problem set and week. It expects to find a program named* `dijkstra.c` *in the current directory. You can use dryrun as follows:*

```
prompt$ ~cs9024/bin/dryrun prob05
```

[show answer]