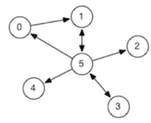
Week 5 Problem Set

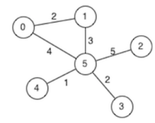
Minimum Spanning Trees, Shortest Paths, Maximum Flows

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

1. (Digraphs)

   a. Consider the following graphs, where bi-directional edges are depicted as two-way arrows rather than having two separate edges going in opposite directions:

   For each of the graphs show the concrete data structures if the graph was implemented via:

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

   b. Consider the following map of streets in the Sydney CBD:

   Represent this as a directed graph, where intersections are vertices and the connecting streets are edges. Ensure that the directions on the edges correctly reflect any one-way streets (this is a driving map, not a walking map). You only need to make a graph which includes the intersections marked with red letters. Some things that don't show on the map: Castlereagh St is one-way heading south, Curtin Pl is a little laneway that you can't drive down and, thanks to the shiny new light rail, George St is now closed for cars between "F" and "K".

   For each of the following pairs of intersections, indicate whether there is a path from the first to the second. Show a simple path if there is one. If there is more than one simple path, show two different paths.

   1. from intersection "D" on Margaret St to insersection "L" on Pitt St
   2. from intersection "J" to the corner of Margaret St and York St (intersection "A")
   3. from intersection "P" on Castlereagh St to the corner of Margaret St and Wynyard Ln ("C")
   4. from the intersection of Castlereagh St and Hunter St ("M") to intersection "H" at Wynyard Park

[show answer]

2. (Warshall's algorithm)

   Apply Warshall's algorithm to compute the transitive closure of your graph from exercise 1b. Choose vertices in alphabetical order. Show the reachability matrix:

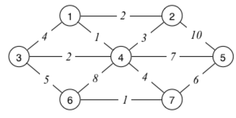
   - after the initialisation
   - after the 1st iteration (vertex 'A') of the outermost loop
   - after the 6th iteration (vertex 'F') of the outermost loop
   - at the end.

   Interpret the values in each of these matrices: Which connections between vertices do the different matrices encode?

[show answer]

3. (Minimum spanning trees)

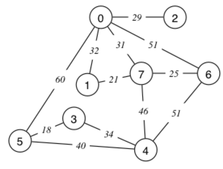
   a. Show how Kruskal's algorithm would construct the MST for the following graph:

   How many edges do you have to consider?

   b. 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.

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

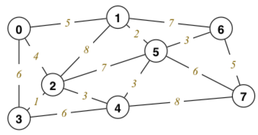
   

   Choose a random vertex to start with. Draw the resulting minimum spanning tree.

[show answer]

4. (Shortest paths)

   Consider the following graph:

   a. What is the shortest path from vertex 0 to vertex 3? From vertex 0 to vertex 7? How did you determine these?

   b. Trace the execution of Dijkstra's algorithm on the 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.

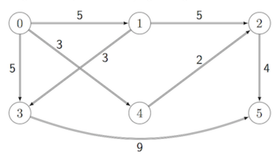
   c. In the given graph, what is the shortest path from vertex 3 to vertex 6? From vertex 6 to vertex 3? How did you determine these?

   d. Trace the execution of Floyd's algorithm on the graph to compute the shortests paths between *all* pairs of vertices.

[show answer]

5. (Maximum flow)

   a. Identify a maximum flow from source=0 to sink=5 in the following network (without applying the algorithm from the lecture):

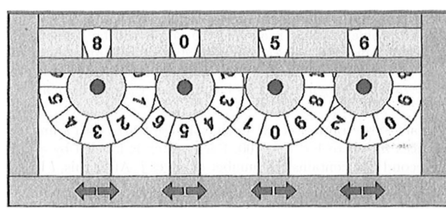
   What approach did you use in finding the maxflow?

   b. Show how Edmonds and Karp's algorithm would construct a maximum flow from 0 to 5 for the network above. How many augmenting paths do you have to consider?

[show answer]

6. **Challenge Exercise**

   Consider a machine with four wheels. Each wheel has the digits 0…9 printed clockwise on it. The current *state* of the machine is given by the four topmost digits `abcd`, e.g. 8056 in the picture below.

   Each wheel can be controlled by two buttons: Pressing the button labelled with "←" turns the corresponding wheel clockwise one digit ahead, whereas presssing "→" turns it anticlockwise one digit back.

   Write a C-program to determine the *minimum* number of button presses required to transform

   - a given initial state, `abcd`
   - to a given goal state, `efgh`
   - without passing through any of *n*≥0 given "forbidden" states, `forbidden[]`$=\{w_1x_1y_1z_1,\ldots,w_nx_ny_nz_n\}$.

   For example, the state 8056 as depicted can be transformed into 0056 in 2 steps if `forbidden[]`$=\{\}$, whereas a minimum of 4 steps is needed for the same task if `forbidden[]`$=\{$`9056`$\}$. (Why?)

   Use your program to compute the least number of button presses required to transform 8056 to 7012 if

   a. there are no forbidden states;
   b. you are not permitted to pass through any state 7055–8055 (i.e., 7055, 7056, …, 8055 all are forbidden);
   c. you are not permitted to pass through any state 0000–0999 or 7055–8055

[show answer]