EECS 114: Assignment 6

November 27, 2012

Due Friday 7 Dec 2012 at 5:00 pm

This assignment is designed to help prepare you for the final exam. *You may collaborate with other students.*

1. For a graph application which frequently needs to find all adjacent vertices of a given vertex, would you use an adjacency matrix representation or an adjacency list representation? Why?

[A]: An adjacency list is better in this case. This is because the adjacency list directly gives a list of all the adjacent vertices for each vertex.

2. We have a graph G:
   (a) Give the adjacency matrix representation of G.

[A]:

\[
\begin{array}{cccccc}
0 & 9 & \infty & 8 & 7 & 2 \\
9 & 0 & \infty & 2 & 3 & \infty \\
\infty & \infty & 0 & \infty & 6 & 1 \\
8 & 2 & \infty & 0 & \infty & 4 \\
7 & 3 & 6 & \infty & 0 & \infty \\
2 & \infty & 1 & 4 & \infty & 0
\end{array}
\]

(b) Write the result of breadth first search (BFS) on G. (Assume the BFS starts at A.)

[A]: A, B, D, E, F, C
(c) Draw the minimum spanning tree of G.

[A]:

3. (a) Why is not it possible that a minimum spanning tree contain a cycle?

[A]: A MST tree has |V|-1 edges. If we have a cycle in the MST tree, it means that we have at least |V| edges in the tree. Therefore, we can always eliminate the edge with the largest weight on the cycle and get a connected tree that has smaller weight.

(b) Can a minimum spanning tree contain the longest edge in a graph? Why or why not?

[A]: Yes. Here is an example:

4. Given the acyclic directed graph G:
(a) Provide the adjacency list representation of G.

[A]:

```
A: [C]
B: null
C: null
D: [B]
E: [A, B, C]
F: [A]
G: null
H: [G]
I: [G]
```

(b) Write the result of depth first search (DFS) on G.

[A]: A C B D E F G I H

1 2 5 7 9 11 12 14 15
4 3 6 8 10 18 13 17 16

(c) Give a topological sort of G.

[A]:

```
F -> I -> H -> G
E -> D -> B -> A -> C
```
5. How can the depth first search algorithm be used to check if a graph has cycles?

[A]: In the depth first search, if we find one of the node is gray, we find a cycle.

6. [Optional] Give the strongly connected components of the following graph. (5 extra points)
Section 22.5 of the textbook describes how to find the strongly connected components of a directed graph.

[A]: \{B, D, E\}
    \{A, C\}
    \{F, G, H, I\}

7. (a) What is the maximum number of edges in an acyclic undirected graph with \( N \) vertices?

[A]: \( N-1 \)

(b) What is the maximum number of distinct edges in an undirected graph with \( N \) vertices?

[A]: \( N(N-1)/2 \)
8. Run Dijkstra's algorithm on the directed graph below using vertex s as the source. Redraw the graph showing the distance to all nodes after each iteration of the while loop on page 658. (Section 24.3).
9. Given the following system of feasible solutions:
(a) Draw the constraint graph corresponding to the system.

\[
\begin{align*}
X_1 - X_2 &\leq -1 \\
X_1 - X_4 &\leq 3 \\
X_2 - X_4 &\leq 6 \\
X_3 - X_2 &\leq 5 \\
X_2 - X_5 &\leq 3 \\
X_3 - X_4 &\leq -7 \\
X_5 - X_1 &\leq -2 \\
X_4 - X_5 &\leq 4
\end{align*}
\]

(b) Use the Bellman-Ford algorithm to find a feasible solution or determine that no feasible solution exists. Show the result of each iteration of the outermost for loop on page 651. (Section 24.1).
10. Compare the running time of the Dijkstra's algorithm with that of the Bellman-ford algorithm.

[A]: The running time of the Bellman-ford algorithm is $O(VE)$. The running time of the Dijkstra's algorithm depends on the implementation of the minimum queue. If we implement the minimum priority queue by minimum heap, then the running time of the Dijkstra's algorithm is $O(V \log V + E \log V)$.

11. We have a graph $G$:

![Graph Image]

(a) Compute the maximum flow for the following graph. Redraw the graph labeling each edge with the flow along that edge that gives the maximum total flow.

[A]:

![Redrawn Graph Image]

The maximum flow is 7.

(b) Give a minimum cut of your flow network.

[A]: $\{s, B, C\}$, $\{A, D, E, F, t\}$
12. Can we say that either of Fulkerson and Edmonds-Karp algorithms always perform better than the other? Provide examples in supporting your answer.

[A]: The Edmonds-Karp algorithm is better. Here is an example:

13. Run Floyd-Warshal's algorithm on the graph below showing the $D^{(k)}$ matrix after each iteration of the outermost for loop.

[A]:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

$D^0=0 \infty \infty 4 9 \infty \quad D^1=0 \infty \infty 4 9 \infty \quad D^2=0 \infty \infty 4 9 \infty \quad D^3=0 \infty \infty 4 9 \infty \quad D^4=0 \infty \infty 4 9 \infty \quad D^5=0 10 11 6 4 9 11 \quad D^6=0 10 11 6 4 9 11 \quad D^7=0 10 11 6 4 9 11$
14. [Optional] Run Johnson's algorithm on the graph below. (15 extra points)
Redraw the graph after each step (i.e. re-weighting, Bellman-Ford's, and Dijkstra's).

D = 0, -7, -5, -1
9, 0, 4, 8
7, -2, 0, 6
3, -6, -4, 0

Bellman-ford:
Re-weight:
Dijkstra's:
15. If a NP-complete problem is solved by a polynomial time algorithm, can we conclude that every NP problem is solvable in polynomial time?

[A]: No. If a NP-complete problem is solved by a polynomial time algorithm, it does not mean every NP problem can be solved in polynomial time.

16. Is an NP-complete problem both NP and NP-hard?

[A]: Yes.