CS302 Final Exam, December 10, 2013 - James S. Plank - Page 2

Question 3: Network Flow

To the right are specifications of **Node** and **Edge** classes. You are implementing network flow in a **Graph** class, which contains a method called **Process_Augmenting_Path()**. This processes an augmenting path, which is represented as a vector of pointers to edges. Your job in this question is to write **Process_Augmenting_Path()**, which has the following prototype:

```
void Graph::Process_Augmenting_Path(vector <Edge *> path);
```

```
class Node {
  public:
    vector <class Edge *> adj;
};

class Edge {
  public:
    Node *n1;
    Node *n2;
    Edge *reverse;
    int residual;
    int flow;
};
```

You don't need to worry about the order of the edges in **path**. You should assume that all edges have reverse edges, which may have zero flow and/or residual value. Therefore, you should not delete edges. You may also assume that **path's** size and residual flow are both non-zero.

To be specific, **residual** flow in the residual flow graph. **Flow** is the flow that will be in the final flow graph.

Question 4: Running Times

Let *G* be a connected, directed weighted graph with *n* nodes, where each node's adjacency list has at most 10 edges. Please tell me the running time of each of the following algorithms or quantities. Use the answer sheet.

- **A**: Prim's algorithm for minimum spanning tree (assuming the edges are undirected).
- **B**: Confirming that the graph is connected.
- **C**: The number of subsets of nodes.
- **D**: Finding an augmenting path with Network Flow, using Edmonds-Karp.
- **E**: Depth-first search.
- **F**: Finding the minimum weight path from one node to another.
- **G**: Identifying whether there is a cycle containing a given node in the graph.
- **H**: Breadth-first search.
- **I**: Kruskal's algorithm for minimum spanning tree (assuming the edges are undirected).
- **J**: Finding the shortest-hop path from one node to another.
- **K**: The possible ways to order the nodes in a vector.
- **L**: Topological sort (assuming that the graph is acyclic).

Question 5: Graph Theory

This question concerns the graph represented on the next page. The graph is shown pictorally without edge weights. The edges are specified in two ways -- first, there is a listing of the edges sorted by weight. Second, there is an adjacency matrix.

- **Part A:** Give me the minimum spanning tree of this graph. Do that by specifying the edges in ascending order of weight.

- **Part B:** The shortest path from A to J has a length of 15. To figure that out, I applied Dijkstra's algorithm to the graph. What I want you to do is tell me what the multimap is when Dijkstra's algorithm terminates. Use scratch paper, and make sure your answer is neat. You don't need to include back-edges -- just show me the multimap in the format {(length,node), (length,node)...}.

- **Part C:** Assume that flow can go in either direction on an edge. What is the maximum flow path from node C to node J? Don't tell me the flow -- tell me the path.

- **Part D:** Assume that edges do have direction -- that they always go from smaller letters to bigger letters (i.e. "AB" goes from A to B, not from B to A). Use topological sort to tell me the weight of the maximum weight path from node A to nodes B, C, D, E and F.