Be neat and concise, but complete.

1. (5 points) The figure below shows a single Fibonacci heap. The numbers next to the nodes are the key values. The ranks are not shown. Show how the heap structure changes following a deletemin operation. Perform the linking steps from left to right – that is always combine the leftmost pair of trees that are “eligible” to be combined.
2. (10 points) Consider the graph shown below. Suppose we apply the general *scanning and labeling* method for shortest paths to this graph.

If the vertices are scanned in topological order, what is the total number of times that a vertex is scanned?

Suppose the vertices are scanned in the most inefficient possible order. How many times are vertices scanned in this case? Explain.

Generalize the example to show that the scanning and labeling method can take $\Omega(2^n)$ time in the worst-case.
3. (10 points) Suppose we assign labels $\lambda(u)$ to each vertex of a directed graph with a length function defined on its edges. Let $\text{length}'(u,v) = \text{length}(u,v) + (\lambda(u) - \lambda(v))$. For a path $p = u_1, u_2, \ldots, u_r$, let $\text{length}(p) = \text{length}(u_1, u_2) + \text{length}(u_2, u_3) + \ldots + \text{length}(u_{r-1}, u_r)$ and let $\text{length}'(p)$ be defined similarly. Show that for two paths $p$ and $q$ joining the same pair of vertices, that $\text{length}(p) \leq \text{length}(q)$ if and only if $\text{length}'(p) \leq \text{length}'(q)$. 
4. (15 points) Let $G$ be a directed graph with each edge $(u,v)$ having a non-negative capacity, $\text{cap}(u,v)$. The bottleneck capacity of a path in $G$ is the minimum capacity of one of its edges and a best bottleneck capacity path is one with the largest bottleneck capacity. A best bottleneck path tree is a spanning tree of $G$ whose paths are best bottleneck paths.

Let $T$ be a spanning tree of $G$ with root $s$ and let $\text{bcap}(u)$ be the bottleneck capacity of the path from $s$ to $u$ in $T$. Show that if $T$ is a best bottleneck path tree, then $\text{bcap}(v) \geq \min\{\text{bcap}(u), \text{cap}(u,v)\}$, for all edges $(u,v)$ in $G$.

Show that if there is a non-tree path $p$ from $s$ to some vertex $x$, that has bottleneck capacity larger than $\text{bcap}(x)$, then there must be some edge $(u,v)$ for which $\text{bcap}(v) < \min\{\text{bcap}(u), \text{cap}(u,v)\}$.
5. (10 points) Consider an execution of the capacity scaling version of the augmenting path algorithm for maximum flows. Suppose that we're at the start of a scaling phase, in which the scaling parameter is equal to 100, and that $X$ is the set of all vertices that can be reached from the source vertex $s$, by paths with capacity of at least 200. Let $X'$ be the set containing all other vertices.

Give an upper bound on the residual capacity of an edge with one endpoint in $X$ and the other in $X'$. Justify your answer.

Suppose there are 40 edges that cross the cut $(X,X')$ that have residual capacity of at least 100 (from the endpoint in $X$ to the endpoint in $X'$). Give an upper bound on the number of augmenting paths that can be found before the end of this scaling phase. Justify your answer.

Suppose that in the original graph, all edge capacities are multiples of 25 and that the largest edge capacity is 400. Give an upper bound on the number of scaling phases in which at least one augmenting path is found. Justify your answer.
6. (15 points) This problem concerns the performance of the shortest augmenting path algorithm for unit networks. In the analysis of shortest augmenting path for general networks, we defined $\text{level}_i(u)$ to be the number of edges in a shortest path from the source vertex to $u$ with positive residual capacity. We also divided the execution of the algorithm into phases, where a new phase starts whenever $\text{level}_i(t)$ changes. Give an upper bound on the number of phases used by the shortest augmenting path algorithm on unit networks. Your answer may refer to the analysis of Dinic’s algorithm for unit networks in your answer. You need not derive anything that was shown in that analysis.

Suppose that $\text{level}_i(t)=k$ at the start of a phase and that the set $S$ of edges $(u,v)$ with $\text{level}_i(v)=\text{level}_i(u)+1$ and positive residual capacity has $m_S$ edges. Give an upper bound on the number of augmenting paths that are found in the current phase. Explain.

Give an upper bound on the running time for the shortest augmenting path algorithm for unit networks, based on your answers to the first two questions.
7. (10 points) The figure below shows an intermediate state in the execution of Dinic’s algorithm with dynamic trees. In the picture of the flograph, the adjacency lists are indicated by the arcs connecting the edges at each vertex, and the solid circles indicate the position of the nextedge pointers in the adjacency lists. So for example, nextedge(\(c\))=(\(b,c\)) and nextedge(\(g\))=(\(g,i\)). The dynamic trees data structure is shown at right.

What is the flow on edge (\(e,h\))?  
What is the residual capacity of edge (\(f,t\))?  
Show the state of the dynamic trees data structure at the point the next search for an augmenting path reaches the sink. Be sure to show all the node costs.

Show the changes to the flow values in the flograph data structure after flow is added to the augmenting path found by this search. You may just mark the changes on the diagram above.