1. (10 points) Suppose that $r$ is a root of some tree in a Fibonacci heap. Assume that just before a `deletemin` operation, $r$ has no children and that after the `deletemin` $r$ has 10 children. Let $C$ be the set of children of $r$ after the delete and let $\text{ranks}(C)$ be the set of rank values for the nodes in $C$ (since this is a set, if several nodes in $C$ have the same rank, their rank value appears just once in $\text{ranks}(C)$).

What is the largest value in $\text{ranks}(C)$ right after the `deletemin`? That is what is max $\text{ranks}(C)$?

What is the value of $|\text{ranks}(C)|$ right after the `deletemin`?

Assume that some time later $r$ is still a tree root and has the same set of children. What is the smallest possible value for max $\text{ranks}(C)$ at this point?

What is the smallest possible value for $|\text{ranks}(C)|$ at this point?

Suppose some time later, $r$ is still a tree root but no longer has the same set of children? Let $C'$ be its current set of children. What is the smallest possible value for $|C'|$?
2. (10 points) The diagram below shows an intermediate state in the execution of the round-robin algorithm. The partition data structure $P$ is just shown as a collection of subsets. The leftist heaps are shown as sets of edges. The tree edges are not shown explicitly.

$$\begin{align*}
P: \{a,f\} \ {b,c} \ {e,h,j} \ {d,g,i,k,m} \\
h(a)=\{af,fc,ab,ad\} \\
h(c)=\{cf,bc,ch,ab,bd,be\} \\
h(e)=\{eh,ej,be,eg,ej,ch,jh,ij\} \\
h(d)=\{ad,bd,dg,dk,eg,gi,gk,ij,im,km\}
\end{align*}$$

Which edges are in the partial MST that has been found so far?

In the heap $h(d)$, which edges would be considered “deleted” by the round-robin algorithm?

Suppose that $h(e)$ is the first item on the list used by the round-robin algorithm. What is the next edge added to the tree?

Show the state of the partition data structure after the next edge is added (you may just list the sets in the partition data structure).
3. (10 points) Consider the partial solution to the maximum weighted matching problem shown below.

List all the augmenting paths in this graph and the weight of each path.

Which of these paths would be selected next by Edmonds algorithm?

Suppose that during the execution of Edmond’s algorithm on some other graph, edges \{a,b\}, \{b,c\} and \{a,c\} form an odd blossom with base a. In addition, the tree containing this blossom also includes edges \{x,a\} and \{y,c\}. Which of these five edges are in the matching?

Which of these edges must be equality edges?

Which of these vertices is the parent of the blossom?
4. (10 points) This problem is concerned with Dinic’s algorithm on unit graphs. Let $G$ be a unit graph with 1,000 vertices and 10,000 edges. Let new phases start at times $T_1$ and $T_2$ with $T_1 < T_2$. Suppose that at time $T_1$, $\text{level}(t)=5$ and at $T_2$, $\text{level}(t)=10$ (where $t$ is the sink). If vertex $u$ has 5 incoming and 10 outgoing edges, what is the largest number of times that $\text{nextedge}(u)$ can be incremented between $T_1$ and $T_2$?

Suppose that at time $T_2$ the value of the flow is 23. What is the largest possible value for the maximum flow?

Approximately, what is the maximum possible number of phases after time $T_2$?
5. (10 points) The diagram below shows an implementation of a dynamic tree used by Dinic’s algorithm (what Tarjan calls the “virtual tree”). Draw the corresponding “actual tree” that is implemented by the given virtual tree. Show how the tree is divided into paths (that is, the show the dashed edges) and for each vertex show its actual costs.

What is the augmenting path in the flow graph?

What is the residual capacity of the path?

Which edges are removed from the dynamic trees data structure after flow is added to the path?
6. (15 points). In lab 5, we used an extension on the basic $d$-heap that supports an efficient `addtokeys` operation. There is another way to implement a heap with an efficient `addtokeys` operation using a self-adjusting binary search tree with vertex costs that are represented using the differential representation used in the dynamic trees data structure.

Assuming that the heap were represented in this way, describe an efficient method to implement the method `key(x)`, which returns the key value for node $x$. Keep in mind that to make this efficient, the tree must be restructured in the usual way for self-adjusting trees. You can assume that the underlying rotation methods update the $\Delta_{\text{min}}$ and $\Delta_{\text{key}}$ values as needed.

Describe an efficient algorithm that implements the `findmin` operation on the heap.

Describe an efficient algorithm for implementing the method `insert(u,k,root)` where $u$ is a new node, $k$ is its key value and `root` is the current root of the tree. Hint: note that $u$ can be inserted at any location in the tree.
7. (10 points) Let $T$ be a balanced binary search tree (aka red-black tree) containing a leaf $u$ with $\text{depth}(u)=10$ and a leaf $v$ with $\text{depth}(v)=13$? What is the smallest possible rank for the root of $T$. What is the largest possible rank?

Approximately, what is the smallest possible number of nodes in $T$? The largest?

Show the tree that results from performing the operation $\text{join}(a,p,x)$ on the set of search trees shown below. The numbers shown are the ranks. Key values have been omitted. Don’t forget to rebalance and show the final rank values.
8. (10 points) The figure at right shows what Tarjan calls a virtual tree (that is, the underlying implementation of the dynamic trees data structure). Each triangle represents one solid tree, $b$, $q$ and $y$ are leaf nodes in their trees and the numbers in the triangles represent the number of nodes in those trees. (so, the tree containing $p$ and $q$ has 400 nodes). Label each of the vertices $a$, $b$, $p$, $q$, $x$ and $y$ with its individual weight, total weight and rank.

Suppose we do a splay at node $y$ in the search tree with root $x$. How many new credits do we need to ensure that we can pay for the splay and maintain the credit invariant?

Suppose we do splays at the three nodes $y$, $q$ and $b$? How many new credits do we need to ensure that we can pay for all the splay steps in these splays and maintain the credit invariant?
9. (10 points). Suppose we apply the FIFO variant of the preflow-push algorithm to a graph with 100 vertices and 500 edges. Approximately, what is the maximum number of steps that relabel a vertex?

Suppose $u$ is a vertex with 10 incident edges. Approximately, what is the maximum number of times that $\text{nextedge}(u)$ is advanced during the execution of the algorithm?

Suppose that at the start of some pass, the value of the potential function used in the analysis is 37. Suppose that during this pass, every step concludes by adding flow to an edge without saturating it. What is the largest possible value of the potential function at the end of the pass?

Suppose that during some other pass, the potential function increases from 41 to 47. What is the smallest possible number of rebalancing steps that can occur during this pass?