MTH 4320: Additional Practice Problems

Instructions at the beginning of the exam:

During this exam you may use the notes and anything you wrote. You may not use any online resources, including search engines and social media. You may not interact with any other people. You may only use your phone to scan your answers and only once you’re completely done writing your solutions.

Unless stated otherwise, you do not need to repeat any proofs or running time analysis that we did in class. For example, you may sort in $O(n \log n)$ time without describing how. You may not use algorithms that we did not learn, such as hash tables or DFS. Running times should use $O(\cdot)$-notation.

Each problem is worth the same number of points. Only your top six answers will be used, so start with the ones that seem easier!

You may use the Edmonds-Karp algorithm to find a max flow in time $O(|V| |E|^2)$. In a 0-1 network the running time is $O(|E|^{3/2})$ and $O(|V|^{2/3} |E|)$. Using flow, one can find a max matching in a bipartite graph in time $O(|V|^{2/3} |E|)$.

**Problem 1.** Let $G = (V, E)$ be a directed graph with edge weights, such that all the weights are numbers from $\{1, 3, 5, 9, 11, 13, 15\}$. Design an algorithm that receives vertices $s, t$ from $V$ and finds a lightest path from $s$ to $t$. The running time should be $O(|V| + |E|)$.

**Problem 2.** Let $G = (V, E)$ be a connected graph and let $e$ be an edge of $E$. Describe an algorithm that checks whether $e$ participates in every BFS tree of $G$. The running time should be $O(|V| + |E|)$.

**Problem 3.** There are $n$ families who plan to attend the wedding of Alice and Bob. The $i$'th family contains $n_i$ members. There are $m$ tables at the wedding, and each table has seats for up to 10 people. To have a social event, Alice and Bob ask that no two people from the same family sit at the same table. We wish to know whether this is possible. For example, if there is a family with $m + 1$ people, then this is impossible.

Phrase the problem as a network flow problem. There is no need to solve the flow problem, but make sure to state what size of flow we are looking for.

**Problem 4.** Let $G = (V, E)$ be a connected graph with edge weights. We know that there are exactly ten cycles in $G$. Describe an algorithm for finding an MST in $G$ in time $O(|V| + |E|)$. 

Problem 5. Let $A$ be a set of $n$ integers. Write an algorithm that checks if there exist four integers $a, b, c, d$ in $A$ such that $a + b + c + d = 1,000$. The values of $a, b, c, d$ are not required to be distinct. The running time should be $O(n^2 \log n)$.

Hint. Let $B$ be the set of all sums of pairs of elements from $A$. Work only with $B$.

Problem 6. Let $G = (V, E)$ be a directed graph with non-negative edge weights. Design an algorithm that receives an edge $e$ of $E$ and checks whether $e$ is the last edge in some lightest path in $G$. For example, the edge $(b, t)$ in the figure is the last edge in the shortest path from $s$ to $t$. The edge $(a, t)$ is not the last edge of any lightest path.

The running time should be $O(|V| \log |V| + |E|)$.

Problem 7. Let $G = (V, E)$ be a connected graph with edge weights. Design an algorithm that checks if the graph has exactly one MST. There is no need to find the exact number of MSTs – only if there is one or more. The running time should be $O(|E| \log |E|)$.

Hint. At each step, consider the set of all edges that have some given weight.

Problem 8. Possibly harder?

Let $G = (V, E)$ be a graph. Recall that a vertex $u$ is reachable from a vertex $v$ if there is a path between $u$ and $v$. We say that a vertex $v$ is central for $G$ if the number of vertices that are reachable from $v$ is larger than the number of vertices that are not reachable from $v$. Describe an algorithm that finds every central vertex of $V$ in time $O(|V| + |E|)$.

Problem 9. In class we solved the problem of finding a longest common subsequence in two DNA strings. We now wish to find the longest common subsequence that contains the letter C at most once. For example, if $S = ACTGACTGACTG$ and $S' = GAACC$, the longest common subsequence cannot be $AACC$ because this contains $C$ more than once. We can take the common sequences $GAA$ or $GAC$.

The running time should be $O(nm)$, as in the original problem. Hint. Define $2mn$ problems, instead of the original $mn$. 
Problem 10. Let $L$ be a list of $n$ numbers. We previously saw how to find one longest increasing subsequence in $L$. We now wish to find how many longest increasing subsequences there are.

For example, in $L = \{0,4,7,3,6,1,2\}$ the longest increasing subsequence is of length three. There are four increasing sequences of length three: $\{0,4,7\}$, $\{0,4,6\}$, $\{0,3,6\}$, $\{0,1,2\}$. The answer is that there are four longest increasing subsequences in $L$.

Write a dynamic programming algorithm that solves the problem in $O(n^2)$ time. Hint. Define subproblems in almost the same way as we did for the original increasing sequence problem.