Practice for Midterm 3.

The following instructions will also be on the exam:

Unless stated otherwise, you do not need to repeat any proofs or running time analysis that we saw in class. For example, you may sort $n$ numbers 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. All the running times should use $O(\cdot)$-notation.

Each problem is worth the same number of points. Only your top 5 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 maximum matching in a bipartite graph in time $O(|V|^{1/2}|E|)$.

Problem 1. Is the following claim true or false?

- There exists a flow network $(V, E, s, t, c)$ where all the capacities are 5, 10, and 15, and the max flow is 29.

If the claim is true, provide such an example. If it is false, explain why.

Problem 2. Let $f_n$ be the sequence of Fibonacci numbers: $f_1 = f_2 = 1$. For every $i \geq 3$ we have $f_i = f_{i-1} + f_{i-2}$. The sequence begins with 1,1,2,3,5,8,13,21,...

We have a text file consisting of $n$ symbols, where the $i$'th symbol appears exactly $f_i$ times. Explain how the tree of the Huffman code of the file looks like.

Problem 3. Consider a flow network $(V, E, s, t, c)$ and an edge $e$ in $E$. Describe an algorithm for checking whether $e$ is in every min cut of the network. The running time should be $O(|V||E|^2)$.

Problem 4. Describe an algorithm that receives a flow network $(V, E, s, t, c)$ and finds a minimum cut consisting of as few edges as possible. The running time should be $O(|V||E|^2)$.

(a) Solve the problem assuming that you are allowed to have non-integer capacities.

(b) Solve the problem using only integer capacities.
**Problem 5.** Professor Moriarty creates a maze as follows. A house of width $n$ and length $n$ is split into $n^2$ rooms of size $1 \times 1$. We refer to the $j$’th room in the $i$’th row as $L_{i,j}$. Some adjacent rooms have a door between them. Some rooms on the boundary of the building have a door to the outside. There are no two doors between the same two rooms and no two doors outside from the same room. Professor Moriarty places $k$ people in rooms, such that no room contains more than one person. Every door automatically shuts after one person passes through it.

We wish to check if all $k$ people can escape to the outside. Phrase the problem as a flow network problem. There is no need to solve the flow problem, but don’t forget to state what size of flow we are looking for and why.

**Problem 6.** Recall the longest increasing sequence problem from class. We now wish to find the longest increasing subsequence whose first element is not a multiple of 5. In this case, the above answer $[15,25,37,45]$ is not allowed because it starts with a multiple of 5. Instead, we can return $[37,45]$.

Write a dynamic programming algorithm for the problem. The running time should remain $O(n^2)$.

**Problem 7.** Let $A_1, A_2, \ldots, A_k$ be sets of integers between 1 and $n$. We wish to take as few sets as possible such that each of the integers $\{1,2,\ldots,n\}$ appears in at least one of the sets we chose. Adam suggests the following greedy algorithm:

*Repeat the following until we have sets containing all of the numbers: (i) Take the largest set. (ii) Remove from the remaining sets every number that is in the set we just took.*

For example, assume that we have the sets $A_1 = \{1,2,3,4,7\}, A_2 = \{4,5,6\}, A_3 = \{2,6,7\}$, and $A_4 = \{4,5\}$. At first the greedy algorithm will take the largest set - $A_1$. After removing the numbers of $A_1$ from the remaining sets, we have $A_2 = \{5,6\}, A_3 = \{6\}$, and $A_4 = \{5\}$. We then take the largest remaining set - $A_2$. We are now done!

Create an example where Adam’s greedy algorithm does not find an optimal solution.
**Problem 8.** Adam went to the store to buy $2n$ specific items. The store had a sale: for every pair of items one buys, the cheaper of the two is free. Find a greedy algorithm that pairs up the items Adam wants to buy, to save as much money as possible.

For example, consider the case where Adam buys four items that 10, 9, 8, and 7. The pairs (9,10) and (7,8) will cost 10 + 8 = 18. On the other hand, the pairs (9,7) and (8,10) will cost 9 + 10 = 19.

Briefly explain why your algorithm always finds an optimal solution.

**Problem 9.** Consider a flow network $(V, E, s, t, c)$ and a vertex $r$ of $V$. We wish to find a flow that goes from $s$ to $t$ and where every part of the flow also passes through $r$. The flow should be as large as possible. Here is an algorithm for the problem:

- Find a max flow $f_1$ from $s$ to $r$, and denote the size of this flow as $k_1$.
- Find a max flow $f_2$ from $r$ to $t$, and denote the size of this flow as $k_2$.
- If $k_1 < k_2$, decrease $f_2$ until it is a flow of size $k_1$. This can be done by repeatedly finding a path in the flow and decreasing the flow through this path.
- If $k_1 > k_2$, decrease $f_1$ until it is a flow of size $k_2$.
- Now that $k_1 = k_2$, combine $f_1$ and $f_2$ into one flow of size $k_1$. Return this flow. Specifically, for every edge $e$ we have $f(e) = f_1(e) + f_2(e)$.

If the above algorithm works, explain why. If not, provide a counterexample.

**Problem 10.** Consider a flow network $(V, E, s, t, c)$ such that every vertex also has a capacity. The flow through a vertex cannot be larger than its capacity. The vertex capacities are also positive integers.

(a) Describe an algorithm for finding a max flow in time $O(|V||E|^2)$.

(b) Can we find a faster algorithm when every vertex capacity is 1? (The edges could have much larger $r$ capacities.)
Problem 11. Frog $A$ and frog $B$ live in a swamp. Frog $A$ is standing on leaf $a_0$ and wishes to visit the leaves $a_1, a_2, ..., a_n$ in this order, by a sequence of direct jumps from leaf to leaf. Similarly, Frog $B$ is standing on leaf $b_0$ and wishes to visit the leaves $b_1, b_2, ..., b_m$ in this order. The two frogs have a strong spiritual bond and cannot be more than 10 feet apart. Also, the two frogs refuse to both jump at the same time.

Describe a dynamic programming algorithm that checks if there exists an order of jumps that satisfies the above restrictions. That is, $m + n$ jumps after which frog $A$ is on leaf $a_n$ and frog $B$ is on leaf $b_m$. The frogs are never more than 10 feet apart. The algorithm is given the location of each leaf, and can compute the distance between two leaves in time $O(1)$. The running time should be $O(mn)$.

Hint. Define $O(mn)$ subproblems.

Problem 12. We are given a list $L$ of $n$ positive integers and another positive integer $M$. Describe a dynamic programming algorithm that checks if there exists a subset of the numbers of $L$ that sums up to $M$.

The running time should be $O(nM)$.

Problem 13. A real-world problem!

We are standing on the bottom left corner of a human-sized $n \times n$ chessboard. Each square has some number of goats standing on it. We wish to travel the board until we get to the top right corner, where at each step we can either move one square to the right or one square up. Each time we get to a square, we give a carrot to each of the goats on it. Describe a dynamic programming algorithm that finds a path that maximizes the number of goats that receive a carrot. The running time should be $O(n^2)$.

Problem 14. There are $n$ doctors working in the pediatrics department of our hospital. We are planning the working schedule for the next $m$ days. The $i$’th doctor deserves to have $n_i$ days off during those $m$ days. Each doctor marks potential days for them to have off. The $i$’th doctor marks more than $n_i$ days and is supposed to get $n_i$ of her choices off. For the department to function, at least half of the doctors must be working every day.

We need to check if we can choose days off for each doctor, such that the above requirements are satisfied. Rephrase the problem as a network flow problem. There is no need to solve the problem.
Problem 15. A company wishes to buy \( n \) specific licenses for cryptographic software. The \( i \)’th license currently costs \( n_i \). Due to national security issues, a company is allowed to buy up to one license per month. Unfortunately, the licenses are also becoming more expensive. Every month the price of each license doubles.

Describe a greedy algorithm that decides which license to buy each month, such that the total cost is minimized. Explain why your algorithm always finds an optimal solution.

For each of the following statements, state whether it is true or false. Explain your answers.

- If a flow network has a flow of size 1000, then the same network must also have a flow of size 800.