# Homework 8

Due: Friday, Oct 31, 11:59 PM PT

1. Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible base stations. We'll suppose there are *n* clients, with the position of each client specified by its (*x, y*) coordinates in the plane. There are also k base stations; the position of each of these is specified by (*x, y*) coordinates as well.
   For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways. There is a known range parameter *r* — a client can only be connected to a base station that is within distance *r* which can be checked using all the known coordinates. There is also a load parameter $C_i$ for each base station i — no more than $C_i$ clients can be connected to the base station i. Your goal is to for the following problem. Given all the input above, design a polynomial-time algorithm to decide whether every client can be connected simultaneously to a base station, subject to the constraints described. (20 points)

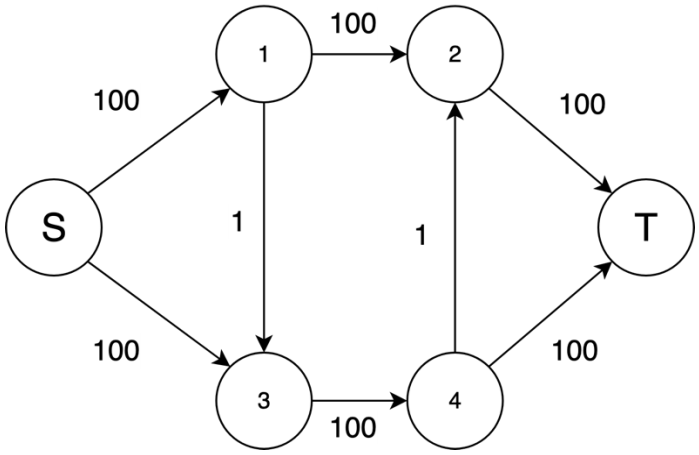
2. The Sushi Express food truck produces a large variety of different lunch menu items. Unfortunately, they can only produce their foods in limited quantities, so they often run out of popular items, making customers sad. To minimize sadness, Sushi Express is implementing a sophisticated lunch-ordering system. Customers text in their acceptable choices before lunch time. Then they can use an algorithm to pre-assign lunches to customers. In general, suppose that, on a given day, Sushi Express has produced m types of food items b1, ..., bm, and the quantity of each type of food item bj is exactly qj. Suppose that n customers a1, ..., an text in their preferences, where each customer ai submits a set Ai of one or more acceptable lunch choices. Customers who do not get one of their choices should receive a $10 voucher. Sushi Express would like to minimize the number of vouchers they give out.
   Give an efficient algorithm for Sushi Express to assign lunches to customers to minimize the number of vouchers. Furthermore, prove the correctness of your algorithm. (20 points)

3. A group of tourists needs to convert all of their USD into various other international currencies. There are n tourists $t_1, t_2, ..., t_n$ and m currencies $c_1, c_2, ..., c_m$. Each tourist $t_k$ has $F_k$ dollars they want to convert, and will accept any mix of currencies, but they have the following constraints: Tourist $t_k$ is willing to trade at most $S_{kj}$ of their dollars for currency $c_j$. (For example, a tourist with 1000 dollars might be willing to convert up to 300 of their USD for Indian Rupees, up to 500 of their USD for Japanese Yen, and up to 400 of their USD for Euros etc.). For each currency cj, the bank can convert at most Bj dollars to cj. Assume that all tourists give their requests (i.e. hand in their available money in dollars) to the bank at the same time.
   (a) Design an algorithm that the bank can use to determine whether all requests can be satisfied. To do this, construct and draw a network flow graph with appropriate source and sink nodes and edge capacities. (10 points)

(b) Prove your algorithm is correct by making an if-and-only-if claim and proving it in both directions. (10 points)

## Ungraded Problems:

1. Based on the following graph, answer the following questions.




   1. What is the max flow from S to T? (1 point)
   2. Find all the min-cuts in the graph. What is the relation between min-cuts and the result from part 1? (2 points)
   3. What is the minimum number of iterations required for the Ford–Fulkerson algorithm to find max flow in this graph? Explain how it happens. (2 points)
   4. What is the maximum number of iterations required for the Ford–Fulkerson algorithm to find max flow in this graph? Explain how it happens. (3 points)
   5. Based on the results from parts 3 and 4, justify the runtime complexity of the Ford–Fulkerson algorithm. (1 point)
   6. How many iterations does Edmonds-Karp require? Does it have the same minimum and maximum number of iterations issue as the Ford–Fulkerson algorithm? (2 points)

2. For a flow network with source S and sink T, determine if the following statements are true or false. For each statement, briefly explain your reasoning. (12 points)

   (a) There always exists a maximum flow that doesn't include a cycle containing positive flow.
   (b) If you have non-integer edge capacities, then you cannot have an integer max-flow value.
   (c) Suppose the maximum s-t flow has value v(f). Now we increase the capacity of every edge by 1. Then the maximum s-t flow in this modified graph will have a value of at most v(f) + 1.

(d) If all edge capacities are multiplied by a positive number k, then the min-cut remains unchanged.

3. Consider a case where there are n tasks, with time requirements $r_1, r_2, \ldots, r_n$ in hours. To perform the tasks, there are k people with time availabilities $a_1, a_2, \ldots, a_k$ in hours. For each task h and person i, we know whether person i has the skills to do task h or not. You are to decide if the tasks can be split up among the people so that all tasks get done. People only execute tasks they are qualified for, and no one exceeds their time availability. Remember that you can split up one task between multiple qualified people in any manner. In addition, there are group constraints. For instance, even if each of you and your two teammates can in principle, spend 4 hours each, you may have decided that between the three of you, you only want to spend 10 hours. Formally, we assume that there are m disjoint groups of people $S_1, S_2, \ldots, S_m$ (where each $S_j \subseteq \{1, \ldots, k\}$), with combined time availability of $t_1, t_2, \ldots, t_m$. That is, any valid solution must ensure, in addition to the previous constraints, that the combined work of all people in group $S_j$ does not exceed $t_j$, for each set j. Disjoint groups mean that one person may belong to at most one constraint set. Give an algorithm with running time polynomial in n, m, k for this problem, and prove that your algorithm is correct. (20 points)