1. (From Jeff Erickson) Suppose you have already computed a maximum flow \( f \) in a flow network \( G \) with integer edge capacities.

(a) Describe and analyze an algorithm to update the maximum flow after the capacity of a single edge is increased by 1.

(b) Describe and analyze an algorithm to update the maximum flow after the capacity of a single edge is decreased by 1.

Your algorithms should be faster than running a standard Max Flow algorithm from scratch.

2. (Problem 9, chapter 7 of Kleinberg and Tardos) Due to large-scale flooding in a region, paramedics have identified a set of \( n \) injured people distributed across the region who need to be rushed to hospitals. There are \( k \) hospitals in the region, and each of the \( n \) people needs to be brought to a hospital that is within a half-hour’s driving time of their current location.

Moreover, it’s desirable that the load on hospitals is balanced: Each hospital receives at most \( \lceil n/k \rceil \) people.

Give a polynomial-time algorithm that takes the given information about peoples’ locations and determines whether it is possible for them to be transferred to hospitals so that the driving time and balanced load constraints are satisfied.

3. In this problem, consider the Stable Matching problem with \( n \) students and \( n \) professors, where each professor needs exactly one student TA and all pairs are acceptable. That is, each professor has all students on its ranked preference list and vice versa.

(a) Give a small problem instance in which the professor-optimal and student-optimal stable matchings are different.

(b) Can you find a problem instance in which there is a stable matching that is neither professor-optimal nor student-optimal?
(c) In class we discussed the following algorithm for Stable Matching. Does it work correctly? If you think that the answer is “yes”, provide a proof, and if you think that the answer is “no”, provide a counter-example.

\[
\begin{align*}
M := & \text{ arbitrary perfect matching} \\
\text{WHILE } & (M \text{ has a blocking pair, say } (p,s')) \\
& \{ \\
& \text{Let } (p,s) \text{ and } (p',s') \text{ be in } M; \text{ remove both from } M \\
& \text{Add } (p',s) \text{ and } (p,s') \text{ to } M \\
& \} \\
\text{RETURN } & M
\end{align*}
\]

4. (From Moore and Mertens) Hall’s Theorem states that a bipartite graph with \( n \) vertices on each side has a perfect matching if and only if every subset \( S \) of the vertices on the left is connected to a set \( T \) of vertices on the right, where \( |T| \geq |S| \). (Hall’s Theorem can be proved using the duality between max flow and min cut; see if you can figure it out yourself, or check Kleinberg and Tardos.)

(a) We say that a graph is \( d \)-regular if every vertex has exactly \( d \) neighbors. Use Hall’s Theorem to show that if \( G = (U, V, E) \) is bipartite and \( d \)-regular (so each vertex in \( U \) is connected to \( d \) vertices in \( V \), and vice versa) then \( G \) has a perfect matching.

(b) Now suppose that we actually want to find a perfect matching. We can do this using an algorithm for Max Flow. But, at least for some values of \( d \), there is a more elegant approach. Suppose that \( d \) is even. Show that we can reduce the problem of finding a perfect matching on \( G \) to the same problem for a \( (d/2) \)-regular subgraph \( G' \), with the same number of vertices, but half as many edges.

(Hint: cover \( G \) with one or more Eulerian paths. Assuming that we can find an Eulerian path in \( O(m) \) time where \( m = dn \) is the number of edges, show that this gives an \( O(m) \) algorithm for finding perfect matchings in \( d \)-regular graphs whenever \( d \) is a power of 2.)