CSE 40113: Algorithms
Homework 7

You may complete this homework in groups of 3 or less students. Note that the integrity policy applies: your group should write up your own work, although you're welcome to work on the problems in a larger group. If you have any questions, please re-read both the homework guidelines and the academic integrity policy carefully, and then come discuss any questions or concerns with me.

Required Problems

1. Describe an algorithm to determine whether a given flow network contains a *unique* maximum (s,t) flow. Then, give an example of a network that has a unique maximum s to t flow, but does *not* contain a unique minimum (s,t)-cut.

2. The Department of Commuter Silence is implementing a more flexible curriculum with a complex set of graduation requirements. The department offers n different courses, and there are m different requirements. Each requirement specifies a subset of the n courses and the number of courses that must be taken from that subset. The subsets for different requirements may overlap, but each course can be used to satisfy at most one requirement.

   For example, suppose there are $n = 5$ courses A, B, C, D, E and $m = 2$ graduation requirements:

   - You must take at least 2 courses from the subset $\{A, B, C\}$.
   - You must take at least 2 courses from the subset $\{C, D, E\}$.

   Then a student who has only taken courses B, C, D cannot graduate, but a student who has taken either A, B, C, D or B, C, D, E can graduate.

   Describe and analyze an algorithm to determine whether a given student can graduate. The input to your algorithm is the list of m requirements (each specifying a subset of the n courses and the number of courses that must be taken from that subset) and the list of courses the student has taken.

3. Ad-hoc networks are made up of low-powered wireless devices. In principle, these networks can be used on battlefields, in regions that have recently suffered from natural disasters, and in other hard-to-reach areas. The idea is that a large collection of cheap, simple devices could be distributed through the area of interest (for example, by dropping them from an airplane); the devices would then automatically configure themselves into a functioning wireless network. These devices can communicate only within a limited range. We assume all the devices are identical; there is a distance D such that two devices can communicate if and only if the distance between them is at most D.

   We would like our ad-hoc network to be reliable, but because the devices are cheap and low-powered, they frequently fail. If a device detects that it is likely to fail, it should transmit its information to some other backup device within its communication range. We require each device x to have k potential backup devices, all within distance D of x; we call these k devices the backup set of x. Also, we do not want any device to be in the backup set of too many other devices; otherwise, a single failure might affect a large fraction of the network.

   So suppose we are given the communication radius D, parameters b and k, and an array $d[1..n][1..n]$ of distances, where $d[i, j]$ is the distance between device i and device j. Describe an algorithm that either computes a backup set of size k for each of the n devices, such that no device appears in more than b backup sets, or reports (correctly) that no good collection of backup sets exists.

4. Sample solved problem: Suppose you are given a directed graph $G = (V, E)$, with a positive integer capacity $c(e)$ on each edge e, a designated source $s \in V$, and a designated sink $t \in V$. You are also given an integer maximum s-t flow in G, defined by a flow value $f(e)$ on each edge e. Now suppose we pick a specific edge $e \in E$ and increase its capacity by one unit. Show how to find a maximum flow in the resulting capacitated graph in time $O(|V| + |E|)$.

Solution: The point here is that $O(V + E)$ is not enough time to compute a new maximum flow from scratch, so we need to figure out how to use the flow f that we are given. Intuitively, even after we add 1 to the capacity of edge e, the flow f can't be that far from maximum; after all, we haven't changed the network very much. In fact, it's not hard to show that the maximum flow value can go up by at most 1:

Claim: Consider the flow network G' obtained by adding 1 to the capacity of any edge e. The value of the maximum flow is either $v(f)$ or $v(f) + 1$, where $v(f)$ is the value of the maximum flow for the original graph G.

Proof of claim: The value of the maximum flow in G is at least $v(f)$, since f is still a feasible flow in this network - increasing one edge's capacity cannot cause f to violate the edge constraints. It is also integer-valued. So it is enough to show that the maximum-flow value in G is at most $v(f) + 1$. By the Max-Flow Min-Cut Theorem, there is some s-t cut (A, B) in the original flow network G of capacity $v(f)$. Now we ask: What is the capacity of (A, B) in the new flow network G? All the edges crossing (A, B) have the same capacity in G that they did in G, with the possible exception of e (in case e crosses (A, B)). But $c(e)$ only increased by 1, and so the capacity of (A, B) in the new flow network G is at most $v(f) + 1$.

So we get the following algorithm, which finds if there is an augmenting path in G'_f (our modified graph) and then adds one to ever edge in that path if it exists:

```
IncreaseEdge($G, f, e$):$G' \leftarrow G$
  Modify G': increase edge $e$'s capacity by 1
  Build the residual graph $G'_f$ for $f$ in $G'$
  BFS(s)
  if $t$ is unmarked:
      return $f$
  else
      $x \leftarrow t$
      while $x \neq s$
          $f((p(x), x)) \leftarrow f((p(x), x)) + 1$
          $x \leftarrow p(x)$
```

For BFS code, I used the following version from page 200 of book which stores parent pointers, but modified for directed graphs:

```
BreathFirstSearch(s):
  put $(\emptyset, s)$ in queue Q
  while Q is not empty
      take $(p, v)$ from queue
      if $v$ is umarked
          mark $v$
          parent$(v) \leftarrow p$
          for each edge $(v, w)$
              if $(v, w)$ is unmarked
                  add $(v, w)$ to Q
```

Proof of correctness: Based on the Maxflow-Mincut theorem from the textbook, f will still be the maximum flow precisely if there are no paths in the residual network. If a path does exist in G'_f, we know BFS will find the path, and by our claim above, we know the overall flow will only one until of flow along that path, since otherwise we would exceed e's capacity.

Runtime: We change 1 edge weight, build a residual graph in $O(V + E)$ time, and run BFS in $O(V + E)$ time. The if statement spends at most $O(V)$ time tracing the path and updating the flow values. So, total time is $O(V + E)$.

■