Problem Set 4 Solutions

This problem set is due before lecture on Tuesday, October 26, 2010.

Exercise 4-1. Do Exercise 15.3-4 in CLRS.
Exercise 4-2. Do Exercise 24.3-5 in CLRS.
Exercise 4-3. Do Exercise 24.5-7 in CLRS.
Exercise 4-4. Do Exercise 25.1-9 in CLRS.
Exercise 4-5. Do Exercise 25.3-6 in CLRS.
Exercise 4-6. Do Exercise 26.2-5 in CLRS.
Exercise 4-7. Do Exercise 26.5-3 in CLRS.

Problem 4-1. Smuggling

Suppose a smuggler wants to sneak $n$ units of contraband into a country by hiding them on $k$ trucks that are driving across the border that day. The $i$-th truck can hold $c_i$ pieces of contraband, for $1 \leq i \leq k$.

Each truck has a known probability $p_i$ of being searched at the border (the $p_i$s are publicly known). If a truck is searched any contraband on it will be found.

The smuggler knows $n$, $k$, and both $c_i$ and $p_i$ for $1 \leq i \leq k$. Give an efficient (polynomial time in $n$) dynamic programming algorithm for finding the best way to allocate the contraband so as to minimize the probability that any of it is found. You may assume that $k = O(n)$.

**Solution:** We solve this problem in a bottom-up fashion by recursively looking at the subquestion $HIDE(m, j)$, which is ‘what is the optimal way to smuggle $m$ units of contraband across the border in the first $j$ trucks’.

Let $A$ be an $n$ by $k$ matrix in which we store our results, and let $T$ be the set of trucks. Consider an algorithm that simply computes the best probability (we can easily include enough information to reconstruct the solution, but for the purpose of clarity here we don’t).

For each $m$ in $(1 \ldots n)$, we first check if truck number one can hold $m$ units of cargo. If so, we set $A[m, 1]$ to $1 - p_1$, otherwise we set it to zero. Then, for each $j$ in $(2 \ldots k)$, we have two possible choices for solving the subquestion $HIDE(m, j)$ - either we can use truck $j$ or not. If we don’t,
then our solution is simply the solution to $\text{HIDE}(m, j - 1)$, which is stored in our matrix already. If we do, we should fill truck $j$ as full as possible, and then store the remaining cargo in the first $j - 1$ trucks - which we have already calculated as $\text{HIDE}(m - c_j, j - 1)$. We simply check which of these options has a lower probability of being caught, and store that answer in $A[m, j]$.

**Correctness:** For the first step in the loop, if the first truck can carry all the contraband, our only choice is to load all the contraband into the first truck, so it has a $1 - p_1$ chance of getting through. If we can’t, then the subproblem is unsolvable, so we say it’s certain to fail.

In the inner loop, each time we increase $j$, we have two possible choices: either use the new truck, or don’t. If we don’t, the probability of being caught is just the probability of not being caught with the old $j$. If we do, we load the new truck as fully as possible (there’s no value in leaving extra space), and multiply the probability of that truck making it through by the best probability for getting the remaining cargo through on the first $j$ trucks.

**Analysis:** The inner for loop runs in $O(1)$ time, so the outer for loop runs in $O(1) + O(k) = O(k)$ time (the first check runs in $O(1)$ time, and then we do $O(1)$ work $k$ times), so the entire program requires $O(nk)$ time, which, given our assumption, is polynomial in $n$.

**Problem 4-2. Recitations**

A TA for a popular computer-science class has a problem. It’s the start of the semester, and all $n$ students in the class have filled out their forms, telling him what subset of the $k$ recitations they can attend. The TA wants to know whether or not there is a way to divide the students up evenly among the recitations so that everyone can attend at their assigned time. Help him out by formulating the question as a network flow problem that can be solved in polynomial time with respect to $n$ (you don’t need to produce an assignment, simply say whether or not one exists). You may assume that $k$ is a small, fixed integer, and that $k$ divides $n$ evenly.

**Solution:** We formulate the network flow problem as follows: make a source node, $S$, and a sink node, $T$. For each student, make a node $s_i$, with an edge from $S$ to $s_i$ that has capacity 1. For each recitation, we make a node $r_i$, that has an edge of capacity 1 coming into it from each student who can attend in that time slot, and an edge of capacity $n/k$ leading out to $T$. If this network has flow $n$, there is a satisfying assignment.

Alternate solution: We formulate the network flow problem as follows: For each recitation slot, we make a node $r_i$. For each of the $2^k$ possible subsets of recitations, we make a node $u_{i_1...,i_j}$. Finally, we add a source node $s$ and a sink node $t$. There is an edge from the source node to each of the subset nodes with capacity equal to the number of students who can attend only that particular subset of the recitations. From each of the subset nodes, an edge goes to each of the recitation nodes that are part of the subset, with effectively infinite capacity. Finally, each recitation node has an edge leading to the sink node, with capacity $n/k$. If this network has flow $n$, there is a satisfying assignment.
Handout 13: Problem Set 4 Solutions

Problem 4-3. Shortest Paths Under Fire

Alyssa P. Skyhacker is commanding a fleet of rebels in a sector of \( n \) planets. Each rebel ship’s crew has landed on a planet, rescued some prisoner or recovered some artifact, and needs to get back to some other different planet as quickly as possible. At least one ship needs to travel from each planet to each other planet. Travel takes place over a set of known hyperspace jumps between planets. Although each hyperspace jump take one hour, you need to be at a specific velocity to enter each jump, so each jump takes a different amount of fuel - we call the fuel needed to jump from system \( i \) to system \( j \) \( c_{ij} \) (measured in space-gallons).

(a) Each rebel has only 500 space-gallons of fuel left. Alyssa decides that she’d prefer if travel went between the safest planets, and has ranked each planet in order of the number of Empire ships currently defending it. Defining the danger of a path as the maximum number of Empire ships that are ever in the same system as the rebel ship (so a path that involves the \( k \) least dangerous systems and no others will always be safer than any plan involving system \( k + 1 \)), Alyssa wants to send each of her ships on the least dangerous path it can manage given its limited fuel. Give an efficient algorithm for Alyssa to use with \( O(n^3) \) performance.

**Solution:** Run Floyd-Warshall and order the nodes in terms of safety. As soon as you find a path with weight \( \leq 500 \), send that ship on its way.

(b) Alyssa has just received intelligence that the Empire is sending in heavy reinforcements to every system in the sector in \( k \) hours. She decides that she needs to risk unsafe planets but has to get every ship to its destination before then. If this isn’t possible, the crew should just stay put for now until their allies can help. Give an efficient \( o(n^3 \cdot k) \) algorithm to use.

**Solution:** Run the matrix-multiplication all-pairs-shortest-paths algorithm, but only multiply out to the \( k \)th power. Repeated squaring gives \( \Theta(n^3 \log k) \) performance.

(c) The Empire’s invasion forces have projected long-range tractor beams in random directions across the sector in an effort to prevent the rebels from reaching their destinations. Each planet is easier to land a hyperspace jump at by some fuel cost \( f_p \), but correspondingly harder to leave from by that jump. Given that your algorithm from part 1 has by this point run to completion, give Alyssa an efficient algorithm to calculate the new shortest paths between all pairs of planets and the running time of that algorithm, and justify its correctness.

**Solution:** For the same reason that Johnson’s reweighting works, this doesn’t affect shortest paths, so just return those paths in \( \Theta(1) \) time.
(d) Unfortunately, the Empire’s reinforcements have arrived early, and each hyperspace
jump now has an independent probability $h_{ij}$ that when traveling from planet $i$ to
planet $j$, the rebel ship will be destroyed. On the bright side, each ship has gotten a
chance to refuel and can therefore take any (simple) path around the sector with fuel
to spare. Alyssa has decided that given this, it’s worth it to have all her ships try to
reach their destinations. Give an efficient algorithm to solve the all-planets least-total-
probability-of-being-blown-up problem.

**Solution:** Take the log of each probability and run your favorite all-pairs shortest-
paths algorithm on the result. The log probability of a ship getting through is the sum
of the log probabilities of getting through each system. We want to maximize this,
but it’s negative, so take the negative of each edge (so that they’re all positive) and
minimize path length.