Problem Set #7

You should try to solve these problems by yourself. I recommend that you start early and get help in office hours if needed. If you find it helpful to discuss problems with other students, go for it. **You do not need to turn these problems in. The goal is to be ready for the in class quiz that will cover the same or similar problems.**

Problem 1: Fractional Knapsack

A thief is robbing a store finds n items; the i^{th} item is worth v_i dollars and weights w_i pounds (both integers). The thief wants to take as valuable a load as possible but can carry only W pounds. However, this thief can take fractions of any item he finds. In other words, the thief doesn't have to take all or none of any particular item.

(a) Explain why this problem displays the optimal substructure property.

Solution

Take w of item j; then we're left with finding an optimal solution for weight $W - w$ given the remaining $n - 1$ items plus $w_j - w$ of item j

(b) Develop an algorithm to maximize the value of thief's load.

Solution

First calculate v_i/w_i for each item. This represents the item's value per pound. Then, sort the items in decreasing order by value per pound (highest value per pound first). Lastly, take items starting from the front of the list and work your way down the list until the knapsack is full.

Problem 2: Minimum Spanning Trees

Let G be a weighted undirected graph, where the edge weights are distinct. An edge is *dangerous* if it is the longest edge in some cycle, and an edge is $useful$ if it does not belong to any cycle in G.

(a) Prove that any MST of G contains every useful edge

Solution

Proof by Contradiction.

Suppose not. That is, suppose there is an MST T of G that does not contain some useful edge e, which connects vertices u and v. Therefore, T must contain some other path from u to v without using edge e. But this implies there is a cycle in the original graph G formed by this path (which goes from u to v) when it is combined with edge e. This contradicts the fact that e is a useful edge.

(b) Prove that any MST of G contains no dangerous edge

Solution

Proof by Contradiction.

Suppose not. That is, suppose there is an MST T of G that contains a dangerous edge e, which is part of some cycle. There is some edge f in the same cycle that is not in T (since a tree has no cycles). f must have a lower weight than e(because e is a dangerous edge). Create a new tree T' that contains f instead of e. T' is a minimum spanning tree with a lower cost than T, which is a contradiction.

Problem 3: Greedy Algorithms

Consider a long, quiet country road with houses scattered sparsely along it. (We can picture the road as a long line segment, with an eastern endpoint and a western endpoint.) Further, let's suppose that despite the bucolic setting, the residents of all these houses are avid cell phone users. You want to place cell phone base stations at certain points along the road, so that every house is within 4 miles of one of the base stations.

(a) Give an efficient algorithm that achieves this goal, using as few base stations as possible.

Solution

Start at the western end of the road and begin moving east until the first moment when there's a house h exactly four miles to the west. We place a base station at this point (if we went any further east without placing a base station, we wouldn't cover h). We then delete all of the houses covered by this base station and iterate on the remaining houses.

For any point on the road, define its *position* to be the number of miles it is from the western end. We place the first base station at the easternmost point (i.e., largest position) s_1 with the property that all houses between 0 and s_1 will be covered by s_1. In general, having placed $\{s_1, \ldots, s_i\}$, we place base station $i+1$ at the largest position s_{i+1} with the property that all houses between s_i and s_{i+1} will be covered by s_i and s_{i+1}.

(b) Prove that the greedy choice that your algorithm makes is the optimal choice.

Solution

Let $S = \{s_1, \ldots, s_k\}$ denote the full set of base station positions that our greedy algorithm places, and let $T = \{t_1, \ldots, t_m\}$ denote the set of base station positions in an optimal solution, sorted in increasing order (i.e., from west to east). We must show that $k = m$.

We do this by showing a sense in which our greedy solution S "stays ahead" of the optimal solution T. Specifically, we claim that $s_i \geq t_i$ for each i, and prove this by induction. The claim is true for $i = 1$ since we go as far as possible to the east before placing the first base station. Assume now that it is true for some value $i \geq 1$; this means that our algorithm's first i centers $\{s_1, \ldots, s_i\}$ cover all of the houses covered by the first i centers $\{t_1, \ldots, t_i\}$. As a result, if we add t_{i+1} to $\{s_1, \ldots, s_i\}$, we will not leave any house between s_i and t_{i+1} uncovered. by the $(i+1)^{st}$ step of the greedy algorithm choose s_{i+1} to be *as large as possible* subject to the condition of covering all houses between s_i and s_{i+1}, so we have $s_{i+1} \geq t_{i+1}$. This proves the claim by induction.

Finally if $k > m$, then $\{s_1, \ldots, s_m\}$ fails to cover all houses. But $s_m \geq t_m$, and so $\{t_1, \ldots, t_m\}$ also fails to cover all houses—a contradiction to it being the optimal solution.

Problem 4: Modifying Dijkstra's Algorithm

We are given an directed graph $G = (V, E)$ on which each edge $(u, v) \in E$ has an associated value $r(u, v)$, which is a real number in the range $0 \leq r(u, v) \leq 1$ that represents the reliability of a communication channel from vertex u to vertex v. We interpret $r(u, v)$ as the probability that the channel from u to v will not fail, and we assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices.

Solution

We create a weight function for the graph where the weight on each edge is the negative log of the reliability, i.e., $w(u, v) = -\log(r(u, v))$. Then we run Dijkstra's algorithm. Because the probabilities of reliability are independent of each other, the probability that a path will not fail is the product of the probabilities that its edges will not fail. We therefore want to maximize $\Pi_{(u,v)\in p} r(u, v)$. Maximizing this value is equivalent to maximizing $\log(\Pi_{(u,v)\in p} r(u, v)) = \sum_{(u,v)\in p} \log(r(u, v))$. (To be careful, we define $\log 0 = -\infty$).