CS 482 Summer 2005

Homework Assignment #2

Out: July 12 Due: July 15

Question 1

A *matching* in a bipartite graph $G = (U \cup V, E)$ is a set of edges $M \subseteq E$ such that each node appears in at most one edge of M. As mentioned in class, the *bipartite matching problem* looks to find a matching of maximum size. To solve this problem, we propose the following greedy algorithm: Select any edge of the graph to add to the matching. Remove it and all incident edges from the graph, and repeat until no edges remain.

(a) Give a bipartite graph G on which this greedy approach does not work. Show which matching the greedy algorithm finds, and which matching is optimal.

(b) Although (a) shows that this algorithm doesn't solve the bipartite matching problem, we can still say something positive about it. In particular, prove that the given algorithm always finds a matching of size at least half the optimal size.

Question 2

Given a graph with a unique cost on each edge, label the edges such that $c_1 < c_2 < \cdots < c_m$. Now create a new cost function c' such that $c'_i = c_i + i$. Prove that the minimum spanning tree under the cost function c is the same as that under c'.

Question 3

You've recently dropped out of college and moved to Hollywood to pursue a career in the television industry—not as an actor, of course, (you were never really very good at that), but as a writer and technical consultant.

Ironically, you've just landed a gig with the newest spinoff *CSI: Ithaca*. The writers need help with the following plot scenario: In this particular episode, a handful of crimes are committed by what they suspect is a serial killer, who murders his victims by flogging them to death with a piece of string. These crimes occur in a series of cities over the span of a few months, and the lead investigators use their best forensic scientists to determine the dates of the murders, and hence the order in which the killer must have visited the cities. They also identify a group of suspects, and the episode

will focus on how the investigators cleverly eliminate some of them.

For each suspect, they have a list of their recent travel destinations. Due to recent airline industry budget cuts, they cannot determine the exact dates the suspects were in each city, but they *can* determine which order they visited their respective destinations. What the writers want you to figure out is, given the list and order of a suspect's travel destinations, can you determine if they could be the serial killer? In particular, do the murderer's crime cities all appear in the suspect's travel destinations, and in the right order? Figure this out, and they can make their investigators look really smart.

For example, if the murders occur in L.A. then Ithaca, then suspect 1, who travels to Ithaca, L.A., Ithaca, L.A., then Boston, could not be ruled out, whereas suspect 2, who travels to Boston, Ithaca, then L.A., could not have committed the crimes.

Find an algorithm that, given the killer's m destination cities, and a suspect's n destination cities, decides in $O(m + n)$ time whether the killer's sequence of cities appear in order in the suspect's sequence of cities.

Question 4

On second thought, you've decided to see your undergrad degree through to the bitter end. It's come time to graduate, and you're bent on joining the throngs of recent graduates who spend their first post-bachelor summer backpacking through Europe. In true recent-graduate style, you are determined to do this trip on a shoestring budget: staying in hostels, hitchhiking between cities, dining on nutella and baguettes, etc. Therefore, you've come to the conclusion that a dual-entry knapsack is a luxury you can do without. Hence your *ridiculously* huge knapsack has just the one opening at the top.

The time has come to pack all your worldly belongings, and your inner computer scientist wants to do this as efficiently as possible. You want the lesser used items (rain jacket, first aid kit, laundry detergent) and more frequently used items (camera, hat, sweatshirt) in good spots so that you don't spend your whole summer searching through your bag. To be concrete, you have n items to pack. Item i has length $\ell_i \geq 0$ and probability p_i of you wanting to access it. Assuming you can only pack items on top of eachother, and that the time to access the i^{th} item down your pack is directly proportional to the combined lengths of the top i items, determine which order to place the items such that the *expected* access time T is minimized. [$T = \sum_{i=1}^{n} p_i L_i$, where L_i is the combined length of items above and including i in the pack.]

Give an algorithm to solve this problem, prove correctness, and state running time.