CS 1510: Homework 8

Zach Sadler zps6@pitt.edu
John Hofrichter jmh162@pitt.edu

September 16, 2013

Greedy Algorithm

Problem 13a

We can construct a counterexample for an input which contains one infected vertex, one neutral vertex, and two cities.
Consider a tree which looks like the following

Then we assign the values $t_1 = 4, t_2 = 1, t_3 = 4$ for the time required to cut edges $1, 2, 3$, and assign $P_1 = 2, P_2 = 2$ to the populations of the two cities. The algorithm will cut the edge which has the minimum t_e to P_c ratio, which is $\frac{t_2}{P_1} = \frac{1}{2}$. Then the algorithm will cut the edge with the next lowest ratio, which is $\frac{t_1}{P_2} = \frac{4}{2}$, so a total of 2.5 for our average time a person has to wait. This is nonoptimal since we could just remove t_1 in the first step and have an average waiting time of 2.

Problem 13b

We can construct a counterexample for an input which contains two infected vertices, no neutral vertices, and two cities. Consider the following picture:

Assign the values $t_e = 1\ \forall e$, since each edge takes unit time to cut. Assign the values $P_1 = 50, P_2 = 10$ to the cities. Then the algorithm will first cut the edge which isolates city 2 because that disconnects 10 people from the infection, then select two more cuts (either of the remaining two first) to disconnect the infection from city 1. Thus our average is $10{\cdot}1 = 10{+}50{\cdot}3 = 10{+}150 = 160$ total time, divided by 60 total people which is $\frac{160}{60}$. Optimally, however we could cut edges 1 and 2 to disconnect city 1, then cut off the last edge for an average of $50{\cdot}2 = 100{+}10{\cdot}3 = 100{+}30 = 130$ total time, divided by 60 total people which is $\frac{130}{60} < \frac{160}{60}$.

Problem 13c

So hear me out. I think this question is actually worded incorrectly. Because as written, it contradicts question 13b. Question 13b says that for any arbitrary number of infected vertices, the algorithm fails if each edge has unit length. But then, we could arbitrarily select 1 as our number of infected vertices, in which case 13b asks us to prove that the algorithm fails for 1 infected vertex and 13c asks us to prove that the algorithm succeeds for 1 infected vertex.

Problem 13d

Randomly disconnect edges according to which one feels right. Hope for the best.
The proof is trivial.

Dynamic Programming

Problem 4

Run out of time to do this problem, then just type nonsense and hope for the best.
The proof is trivial.

Dynamic Programming Problem 5

Optimal Weights

	1	2	3	4	5
1	0.5	0.6	0.85	1.4	2.15
2	0	0.05	0.2	0.55	1.05
3	0	0	0.1	0.4	0.9
4	0	0	0	0.2	0.65
5	0	0	0	0	0.25

Roots

	1	2	3	4	5
1	K_1	K_1	K_1	K_1	K_1
2	X	K_2	K_3	K_4	K_4
3	X	X	K_3	K_4	K_4
4	X	X	X	K_4	K_5
5	X	X	X	X	K_5

Reconstructing the Tree

Look to the upper-right corner to find the overall root of the tree: in this case, K_1. Since K_1 is the root, the left subtree must be empty, and the right

subtree must use the keys 2 through 5. Look at entry (2, 5) to find that the optimal root there is K_4. Since K_4 is the root, the left subtree must use the nodes 2 and 3, and the right child must be K_5. Look to entry (2, 3) to find that the root there is K_3, so K_2 must be the left child. Therefore, the final tree has overall weight 2.15, and looks like this: