CS 3511: Algorithms Honors, Homework 6

Instructor: Prof. Merrick Furst
TAs: Karthik Gopalakrishnan, Pranathi Reddy Tupakula, Jiachen Li

Please submit a LaTeX-formatted PDF containing your solutions on T-Square, by April 13, 3:00 pm.

1 Minimum Spanning Tree: An Alternate Approach

(5 points) Prove that if you pick any cycle in a edge-weighted, undirected graph, and e is the edge with the largest weight in that cycle, there exists a minimum spanning tree that does not contain e.

(10 points) Design an algorithm to find minimum spanning trees by using the above property and analyze its run-time complexity in detail.

2 Greedy: Triathlon Ordering!

You're the organizer of the Georgia Tech triathlon, in which student athletes will swim, run and bike (in that order). Usually all the athletes perform these three activities in this order asynchronously, but unfortunately you were only able to secure the use of one lane in the swimming pool in the CRC. This puts a bottleneck during the swimming leg of the race: only one athlete can swim at a time.

You need to decide on an initial ordering of the athletes, where the first athlete swims first and upon completing the swimming leg, the second athlete can begin swimming (and the first athlete begins running at the same time). But depending on how you order the athletes, the overall triathlon might take too long to finish! Thankfully, you have time-estimates for each athlete for each leg of the triathlon, and you're in CS 3511, so you can devise an algorithm to compute an ordering such that the estimated time for the triathlon to finish is minimized. Note that the triathlon finishes when all athletes have finished.

(15 points) Given n athletes $[1, \ldots, n]$, their corresponding time-estimates for swimming $[s_1, \ldots, s_n]$, running $[r_1, \ldots, r_n]$ and biking $[b_1, \ldots, b_n]$, devise an efficient greedy algorithm to produce an ordering of the athletes such that the estimated time of completion of the triathlon is minimized and prove that the algorithm does indeed give the optimal ordering (hint: use an exchange argument!).

3 Dynamic Programming: Pokémon Wars!

You have two Pokémons: Charizard and Pikachu! During your travels through this beautiful world, you discover a famed mansion known to be filled with N Pokémons. You want to navigate through the mansion! As you navigate, you will encounter each of the N Pokémons in the mansion, one at a time. When you encounter Pokémon i, you can do one of two things:

1. pick either Charizard or Pikachu to fight Pokémon i
2. evade Pokémon i without fighting it

Fighting Pokémon i gives you a reward of r_i, but your Pokémon who fought that battle gets stressed by an amount s_i. Evading Pokémon i gives you no reward, but both Charizard and Pikachu share stress s_i equally among themselves since hey, evasion after encountering is stressful too! (Assume s_i is even.) Both Charizard and Pikachu have a total stress tolerance limit of S each and before entering the mansion, they're both well-rested and have a stress level of 0. A fight will only give you reward if you win, and you win only if that fight doesn't increase the stress level of your chosen fighting Pokémon beyond S.

If at some point, neither fighting nor evading is an option because exercising either option would cause the stress levels of either Charizard or Pikachu (or both) to rise beyond S, you will have to stop your navigation and retreat from the mansion to let them recuperate. You can also spend your collected reward!

For example, suppose the current stress levels of Charizard and Pikachu are 8 and 7, their stress tolerance limit is 10 each, and the next, yet-to-be-encountered Pokémon has an associated stress of 12, then neither fighting nor evading is an option because both actions cause the stress levels of Charizard and Pikachu to exceed their stress tolerance limits. So you absolutely must retreat from the mansion.

You will encounter the N Pokémons in a fixed order known to you before you enter the mansion. All the associated stresses and rewards for the N Pokémons are also known to you before you enter the mansion.

(20 points) Use dynamic programming to compute the maximum total reward you can earn before retreating from the mansion: prove optimal substructure, provide the recurrence relation for the reward, design a top-down approach using memoization and provide the time and space-complexity of your approach.