15-251 : Great Theoretical Ideas In Computer Science

Fall 2013

Assignment 5 (251 Shark Week)

Due: Thursday, Oct. 10, 2013 11:59 PM

Name: _____

Andrew ID: _____

Question:	1	2	3	4	5	Total
Points:	25	15	35	25	10	110
Score:						

1. 251 Jumps the Shark

(10) (a) I loved my kittens, but I eventually got a little...bored with kittens and decided to find a new, more awesome pet. So I went to the pet store to buy some sharks! The pet store had n sharks, and they were all so adorable that I wanted to buy them all. However, the sharks already have some established (direct) friendships among them. Friendships are mutual. Let's say two sharks u and v are *indirect friends* if there is some chain of sharks such that u is the first shark in the chain, v is the last shark, and for each pair of consecutive sharks in the chain, those two sharks are friends. The problem is, two sharks eat each other if they are not even indirect friends. Prove that if there are at least $\frac{(n-1)(n-2)}{2} + 1$ direct friendships among sharks, every pair of sharks must have indirect friendship, and I can buy all n sharks without any shark eating any other shark.

Solution:

(15) (b) In order to be a responsible shark owner, I need to study the shark food chain. Turns out, the pet store was lying about the sharks. No two of the n sharks I have are even indirect friends, but for each pair of sharks, there is exactly one shark that can eat the other. On the day that I buy them, each shark decides to eat one of the sharks that it can eat. Prove that there is a way for the sharks to choose their preys, and a sequence in which the sharks eat their chosen preys, such that only one shark will remain at the end of the day. [HINT: Think about *directed paths of sharks*.]

Solution:

2. Card Shark

(15) (a) Turns out, my sharks were a special variety of sharks, card sharks, that can play poker. Every Friday night the sharks play a high-stakes game of poker using waterproof cards (one of them, Sylvester Shark, wins a lot, so he has also become a bit of a loan shark...).

One night they decide to play a different card game instead. Suppose that 13 sharks are each dealt 4 cards from a standard 52-card deck. Show that it is possible for each of them to select one of their cards so that no two sharks would have selected cards of the same rank.

Solution:

3. Land Shark

Many years later, my sharks were all grown up and I was planning to sell them to various countries. However, each country only has room for so many sharks, and each country also prefers the sharks in different order! My sharks are also picky about which country they end up in, so I held a meeting with the various world leaders (they have nothing better to do, after all) and we formulated the problem as follows.

Consider our set of n countries $C = \{c_1, \ldots, c_n\}$ and a set of m sharks $S = \{s_1, \ldots, s_m\}$. Each country has a preference list over the sharks, and each shark has a preference list over the countries. Furthermore, each country c_i has t_i slots available so that each country c_i can be assigned at most t_i sharks, while each shark has to be matched with at most one country. Assume that the total number of sharks is equal to the total number of slots available in all countries, i.e., $m = \sum_{i=1}^{n} t_i$. Our goal is to assign every shark to some slot in some country so that all slots in all countries are filled. This looked much like the stable marriage problem so far. However, we realized that a pair (c, s) of country c and shark s can be unstable in *two* different ways now:

1. **UNST1:** If s is *not assigned* to any country, and c prefers s to *some shark* s' that is assigned to one of its slots.
2. **UNST2:** If s is assigned a slot in country c', s prefers c to c', and c also prefers s to *some shark* s' that is assigned to one of its slots.

We call an assignment of sharks to slots in countries a "stable marriage" if it has no unstable pairs of either of the two kinds mentioned above. We came up with the following variant of the Gale-Shapley algorithm that we think always produces a stable marriage between sharks and countries. To understand the algorithm better, imagine the countries as the men who propose, and the sharks as the women who accept/reject the proposals.

- Initially all sharks $s \in S$ are unassigned and all slots at all countries $c \in C$ are open.
- While there is a country c that has at least one open slot and has not "proposed" a slot to every shark already:
  - Choose such a country c.
  - Let s be the highest ranked shark in c's preference list to whom c has not yet "proposed" a slot. Let c propose one of its open slots to s.
  - If s is unassigned, then:
    - (c, s) become assigned and a slot closes at c.
  - Otherwise, if s is currently assigned to a slot in c':
    - If s prefers c' to c, then:
      - The proposal fails and the slot at c remains open, so nothing changes.
    - Otherwise, if s prefers c to c':
      - (c, s) become assigned and a slot closes at c.
      - The previous assignment (c', s) breaks and a slot opens up at c'.

End of While Loop

- Return the set of assigned pairs.

Help us prove the following desired properties of our algorithm. NOTE: Please review each and every detail of the Gale-Shapley algorithm, the proof of its termination, and the proof that it always produces a stable marriage before you start solving this problem. You'll need all those details here.

(15) (a) Following the same steps as for the proof of termination of the Gale-Shapley algorithm, show that the algorithm outlined above also terminates by proving an explicit upper bound on the number of executions of the "while" loop in terms of the number of countries and the number of sharks. Also, show that no shark will be left unmatched at the end of running our algorithm. (Please keep your solution crisp. A good strategy to write concise solutions is to reiterate your solution a couple of times after you find one.)

Solution:

(10) (b) Show that in the solution returned by the algorithm, there are no unstable pairs of type UNST1.

Solution:

(10) (c) Show that in the solution returned by the algorithm, there are also no unstable pairs of type UNST2.

Solution:

4. Sharking Up the Wrong Tree

To remember my sharks, I drew a tree, where each vertex is labelled with the name of the shark, so I have n vertices labelled $s_1, s_2, \ldots, s_n$ (I'm not very creative at naming sharks!).

(5) (a) Show that in the Prüfer sequence of my tree, every s_i appears exactly $deg(s_i) - 1$ times, where $deg(s_i)$ is the degree of the vertex labelled s_i.

Solution:

(10) (b) Fix some integers $d_1, \ldots, d_n$ such that $1 \leq d_i \leq n-1$ for each $i \in [1, n]$ and $\sum_{i=1}^{n} d_i = 2n - 2$. I want to draw my tree such that the degree of s_i is d_i for each $i \in [1, n]$. Use part (a) to calculate the number of possible trees I could draw (note that these will be labelled trees by definition). The final answer should be a closed-form expression in terms of n and d_1 through d_n. It can involve sums or products, but not recurrence relations etc.

Solution:

(10) (c) Suppose instead of fixing the degree of each shark, I wanted exactly three of the n sharks to have degree 1. Use part (b) to calculate the number of trees I could draw in that case. You must give a closed-form for the answer in terms of n.

Solution:

5. Bonus: Pool Shark

(10) (a) After their poker game, Sylvester and the other sharks started playing pool. They liked it enough that they decided to hold a tournament. They formed $2n$ teams among the sharks. They wanted each team to play against every other team exactly once over the course of the tournament, but they also wanted each team to play exactly one match each day. In each match, one of the two participating teams would win and the other would lose, so there are no draws.

They arranged the matches so that there were exactly n matches each day, and the tournament lasted for exactly $2n - 1$ days. I like all my sharks, so I want to give each team a prize to make them feel special. Can I choose a winning team from each day without choosing any team more than once? (This leaves one team out, but they probably weren't special anyway.)

Solution: