CPSC 320: Intermediate Algorithm Design and Analysis
Assignment #1, due Thursday, January 19th, 2012 at 11:00

[8] 1. The Stable Matching problem, as discussed in class, assumes that every woman and every man has a fully ordered list of preference. In this and the next problem, we consider the situation where we have n women and n men (as before), but where a woman or man may have ties in her/his ranking. For instance, woman w_1 might like man m_3 best, followed by m_1 and m_4 in no particular order (that is, she does not prefer m_1 to m_4, or m_4 to m_1), followed by m_2. In this case, we will say that w_1 is *indifferent* between m_1 and m_4. It is of course possible for a woman or a man to be indifferent between more than two people.

A *strong instability* in a perfect matching consists of a woman w and a man m such that w and m both prefer each other to their current partner. Does there always exist a perfect matching with no strong instability? Either give an algorithm that finds such a matching and prove the correctness of your algorithm, or give an example where every perfect matching has a strong instability.

Solution: The algorithm is simple: we break ties arbitrarily in each person's preference list, and then run the usual Gale-Shapley algorithm. Every strong instability in the matching it returns would be an instability according to the revised preference lists, and we know the matching returned by the Gale-Shapley algorithm does not contain any.

[8] 2. Continuing with the same setup as in the previous question, let us define a *weak instability* as a woman w with partner m and a man m' with a partner w', where either

- m prefers w' to w and w' either prefers m to m' or is indifferent between these two choices, or
- w' prefers m to m' and m either prefers w' to w or is indifferent between these two choices.

Does there always exist a perfect matching with no weak instability? Either give an algorithm that finds such a matching and prove the correctness of your algorithm, or give an example where every perfect matching has a weak instability.

Solution: Consider the following preference lists:

$$\begin{array}{ll} m_1: & w_1, w_2 \\ m_2: & w_1, w_2 \\ w_1: & m_1 = m_2 \\ w_2: & m_1 = m_2 \end{array}$$

If we match m_1 with w_1 and m_2 with w_2, then we have a week instability since m_2 would prefer to be paired with w_1, and w_1 is indifferent between m_1 and m_2. If we match m_1 with w_2 and m_2 with w_1, then we once again have a week instability since m_1 would prefer to be paired with w_1, and w_1 is indifferent between m_1 and m_2. Therefore there does not always exist a perfect matching with no weak instability.

[8] 3. Karaboudjan Shipping Lines Inc. is a shipping company that owns n ships and provides service to n ports over a period of m days. Each of its ships has a schedule that says, for each of the m days, which of the ports it is currently visiting, or whether it is out at sea. You can assume that $m \geq n$. Each ship visits each port for exactly one of the m days. For safety reasons, KSL Inc. has the following strict requirements:

† No two ships can be in the same port on the same day.

The company wants to perform maintenance on all the ships this month, via the following scheme. They want to *truncate* each ship's schedule: for each ship S_i, there will be some day when it arrives in its scheduled port and simply remains there for the rest of the month for maintenance. This means that S_i will not visit the remaining ports on its schedule that month (if any), but this is okay. So the truncation of S_i's schedule will simply consist of its original schedule up to a certain specified day on which it is in a port P. The remainder of the truncated schedule simply has it remain in port P.

The company's question to you is the following: given the schedule for each ship, find a truncation of each so that condition † continues to hold: no two ships are ever in the same port on the same day. Show that such a set of truncations can always be found, and give an algorithm to find them.

Example: suppose we have two ships, two ports, a four day schedule, and that the ships have the following schedules:

Ship	Day 1	Day 2	Day 3	Day 4
S_1	port P_1	at sea	port P_2	at sea
S_2	at sea	port P_1	at sea	port P_2.

The only way to choose truncations would be to have the first ship remain in port P_2 starting on day 3, and have the second ship remain in port P_1 starting on day 2.

Solution : This can be solved directly using the Gale-Shapley algorithm, by setting the preference lists of each ship and port as follows:

- The ship S_i's preference list contains the ports that S_i visits, in the order in which S_i visits them. That is, S_i "likes" an earlier port better than a later one.
- The port P_j's preference list contains the ships that visit P_j, in reverse order from the order in which they visit P_j. That is, P_j "likes" an earlier ship less than a later one.

We know that the perfect matching returned by the Gale-Shapley algorithm contains no instability. Let us prove by contradiction that the truncated schedules do not result in more than one ship being in the same port at the same time. Suppose such a conflict occurred. Because the initial schedules do not conflict, this means there are two ships S_i, S_j such that S_i stops in port P_a, and S_j visits P_a after S_i has stopped there, before stopping at a port P_b. Notice that

- Since S_i visits P_a before S_j, ship S_j occurs before S_i in port P_a's preference list.
- Since S_j visits P_a before stopping at P_b, port P_a occurs before P_b in S_j's preference list.

But this is an instability in the matching returned by the Gale-Shapley algorithm, which we know can not exist. Therefore, the truncated schedules can not contain a conflict.

[8] 4. There is a class of folk songs and holiday songs in which each verse consists of the previous verse, with one extra line added on. "The twelve days of Christmas" has this property; for example, when you get to the fifth verse, you sing about the five golden rings and then, reprising the lines from the fourth verse, also cover the four calling birds, etc. all the way to the partridge in the pear tree. The Aramaic song "Had gadya" from the Passover Haggadah and the French song "Y'a qu'un cheveu sur la tête à Mathieu" work like this as well, as do many other songs.

These songs tend to last a long time, despite having relatively short scripts. In particular, you can convey the words plus instructions for one of these songs by specifying just the new line that is added in each verse, without having to write out all the previous lines each time. So the phrase "five golden rings" only has to be written once, even though it will appear in verses five and onward.

There is something asymptotic that can be analyzed here. Suppose, for concreteness, that each line has a length that is bounded by a constant number of words c, and suppose that the song, when sung out loud, runs for n words total. Show how to encode such a song using a script that has length in $O(f)$, for a function $f(n)$ that grows as slowly as possible. You should specify what the function $f(n)$ is, and prove that your encoding's length is in $O(f)$.

Solution: Let us first compute the length of the song as a function of the length of its encoding. If each line contains c words, there is a p-word prefix to each verse (as there are in many such songs, think of "On the first day of Christmas, my true love gave to me..."), and there are l lines, then the length n of the song will be

$$n = \sum_{i=1}^{l} p + ci = pl + \frac{cl(l+1)}{2}.$$

To determine f, we first need to solve for l as a function of n. By rearranging terms, we find that

$$cl^2 + (c+2p)l - 2n = 0.$$

By applying the quadratic formula, and knowing that $l \geq 0$, we obtain

$$l = \frac{-(c+2p) + \sqrt{(c+2p)^2 + 8nc}}{2c}$$

which means that $l \in \Theta(\sqrt{n})$ (note that all of the other ugly-looking terms do not depend on n, and hence are constants). Assuming that we encode the song by first writing the prefix, and then the l lines, the length of the encoding will be $p + cl$, which is therefore also in $\Theta(\sqrt{n})$.