COL 351 Aug 16, 17, 21

TUTORIAL SHEET 3

1. **(KT-Chapter 1)** Decide whether the following statement is true or false: "In every instance of the stable matching problem, there is a stable matching containing a pair (m, w) such that m is ranked first in the preference list for w, and w is ranked first in the preference list of m".

   Solution Sketch: False. It is easy to give instances where there is no such pair.

2. Give an instance of the stable matching problem for which the algorithm discussed in class takes $\Omega(n^2)$ time.

   Solution Sketch: The men have identical rankings of all women (the ranking list of women can be arbitrary.

3. **(KT-Chapter 1)** Gale and Shapley published their paper on the stable marriage problem in 1962; but a version of their algorithm had already been in use for ten years by the National Resident Matching Program, for the problem of assigning medical residents to hospitals.

   Basically, the situation was the following. There were m hospitals, each with a certain number of available positions for hiring residents. There were n medical students graduating in a given year, each interested in joining one of the hospitals. Each hospital had a ranking of the students in order of preference, and each student had a ranking of the hospitals in order of preference. We will assume that there were more students graduating than there were slots available in the m hospitals. The interest, naturally, was in finding a way of assigning each student to at most one hospital, in such a way that all available positions in all hospitals were filled. (Since we are assuming a surplus of students, there would be some students who do not get assigned to any hospital.) We say that an assignment of students to hospitals is stable if neither of the following situations arises.

   - **First type of instability:** There are students s and s', and a hospital h, so that (i) s is assigned to h, (ii) s' is unassigned, and (iii) h prefers s' to s.
   - **Second type of instability:** There are students s and s', and hospitals h and h', so that (i) s is assigned to h and s' is assigned to h', (ii) h prefers s' to s, and s' prefers h to h'.

   So we basically have the stable marriage problem, except that (i) hospitals generally want more than one resident, and (ii) there is a surplus of medical students. Show that there is always a stable assignment of students to hospitals, and give an efficient algorithm to find one. The input size is $\theta(mn)$; ideally, you would like to find an algorithm with this running time.

Solution Sketch: The algorithm is similar to the stable matching algorithm. The hospitals propose to the candidates in order of preference. A candidate, if not already assigned or assigned to a less preferable hospital, accepts the proposal; otherwise rejects it. There can be at most mn proposals, and so we are done. Check that all of the operations can be implemented using arrays.

A common pitfall is that if candidates propose to hospitals, then hospitals need to check if they have a less preferable candidate. Since a hospital can have large number of positions, it is not clear how to perform this operation in constant time.

4. Consider the following algorithm for finding minimum spanning tree: sort all edges in *decreasing order* of weight. Let the edges be $e_1, \ldots, e_m$. Consider edges in this order, and initialize the set T to G, the entire graph. When we consider an edge e_i, we remove it from T if T contains a cycle containing e_i; otherwise we keep e_i. Prove that the final set T will be a minimum spanning tree (assume that G is connected).

   Solution Sketch: You can argue directly as in proof of Kruskal's algorithm, or prove that the tree constructed here will be exactly the one constructed by Kruskal's algorithm. Let us try the latter approach. Suppose this algorithm discards an edge $e = (u, v)$. Let L_e be the set of weight less than that of e. Because the algorithm discards e, it follows that there is already a path from u to v in L_e. Now consider running Kruskal's algorithm. It will consider edges in L_e before looking at e. Note that that if x and y are in the same connected component in L_e, they will also be in the same connected component in Kruskal (why?). So u and v will be in the same connected component when the Kruskal's algorithm considers e. But then, it will not select e.

5. You want to throw a party and is deciding whom to call. You have n people to choose from, and you have made up a list of which pairs of these people know each other. You want to pick as many people as possible, subject to two constraints: at the party, each person should have at least five other people whom they know and five other people whom they don't know. Give an efficient algorithm that takes as input the list of n people and the list of pairs who know each other and outputs the best choice of party invitees. Give the running time in terms of n.

   Solution Sketch: Initialize a set S to the set of all n people. Repeat the following step as long as we are able to shrink the size of S – if there is a person x in S who knows less than 5 other persons *in* S or there are less than 5 people in S who do not know x, we remove x from S.

   In order to prove correctness, let $x_1, x_2, \ldots, x_k$ be the persons discarded by our algorithm (in this order). Now prove by induction on i that no feasible solution can invite $x_1, x_2, \ldots, x_i$.