Massachusetts Institute of Technology Handout ??
6.042J/18.062J: Mathematics for Computer Science October 8, 1998
Professors Nancy Lynch and Srinivas Devadas

Supplement to Lecture 6

1 The Dating Game

Suppose that there are an equal number of boys and girls in a town. This lecture concerns the problem of finding a good matching; that is, a good pairing of boys and girls into dating couples. We suppose that each boy has a ranking of all the girls according to how much he likes them, and each girl has a similar ranking of all the boys. If a boy and a girl that we do not match-up each ranks the other above his or her current date, then they are called a *rogue couple*. A matching with no rogue couples is called a *stable matching*. Our goal is to devise and analyze an algorithm that takes the boys' and girls' rankings as input and produces a stable matching as output.

The following notation will be used throughout the lecture. There are N girls and N boys. The set of girls is called G, and the set of boys is called B. A *matching* is a set of pairs (g, b), where $g \in G$ and $b \in B$, such that every girl and boy appears in exactly one pair.

We define $\text{rank}_b(g)$ to be the rank that boy b assigns to girl g. For example, if $\text{rank}_b(g) = 2$, then girl g is boy b's second favorite. Similarly, $\text{rank}_g(b)$ indicates the rank that girl g assigns to boy b. This notation may be confusing because a boy assigns the numerically highest rank to his least-favorite girl. In addition, we define $\text{pref}_b(k)$ to be boy b's k-th favorite girl, and $\text{pref}_g(k)$ to be girl g's k-th favorite boy.

A pair (g, b) is called a *rogue couple* in a matching M if $(g, b') \in M$, $(b, g') \in M$, and the following two conditions hold:

$$\begin{aligned} \text{rank}_g(b) &< \text{rank}_g(b') \\ \text{rank}_b(g) &< \text{rank}_b(g') \end{aligned}$$

A matching that has no rogue couple is called a *stable matching*.

2 The Algorithm

We will use the following algorithm to generate a matching. The input is the set of all boys' and girls' rankings. Generally, each girl pursues one boy at a time, and each boy reject pursuers that he does not want to date. More precisely, a girl always pursues her top-ranked boy that has not yet rejected her. If a boy is currently pursued by more than one girl, then he may reject any of his pursuers other than his favorite among them. The algorithm terminates when no more rejections are possible. (We will show that this implies that the algorithm terminates when every boy has a single pursuer.) The output is the matching that pairs each boy with his final pursuer.

Note that this algorithm is underspecified. One boy may swiftly reject all pursuers other than his favorite, while another boy tries to amass as large a pool of pursuers as possible before issuing any rejections. One might suppose that by carefully timing his rejections, a boy might better his odds of ending up with the girl of his dreams. We will show that this is not the case; the output of the algorithm is independent of the timing of rejections.

3 A State Machine Model

We can model the dating game with a state machine as follows.

A state consists of an assignment to a set of variables indexed by the girls. We denote these variables by cur_g and allow them to take on values in the range 1 to $N+1$. Intuitively, cur_g is the rank of the boy that girl g is currentl pursuing. If $\mathrm{cur}_g = N+1$, then girl g has been rejected by all her pursuers and is doing nothing. (However, we will show below that this can never actually happen.) In the initial state, all the variables cur_g are equal to 1.

Before describing the transitions of the state machine, we introduce some derived variables.

- For each girl g, let $\mathrm{cur\text{-}boy}_g$ be the boy that girl g is currently pursuing, namely $\mathrm{pref}_b(\mathrm{cur}_g)$. If the girl g has been rejected by all suitors, and thus $\mathrm{cur}_g = N+1$, then $\mathrm{cur\text{-}boy}_g$ is defined to be the null value Λ.
- For each boy b, we define best_b to be the rank of boy b's favorite current pursuer. If boy b currently has no pursuers, then $best_b$ is defined to be $N+1$.
- For each boy b, we define $\mathrm{best\text{-}girl}_b$ to be boy b's favorite current pursuer. That is, $\mathrm{best\text{-}girl}_b = \mathrm{pref}_b(\mathrm{best}_b)$. If boy b is not pursued by any girl, then $\mathrm{best\text{-}girl}_b$ is defined to be Λ.

Each state machine transition corresponds to a boy rejecting one of his pursuers other than his favorite among them. Formally, from each state where $\mathrm{cur\text{-}boy}_g = b$ and $\mathrm{best\text{-}girl}_b \neq g$, there is a transition to the state the differs only in that cur_g is incremented by 1. Intuitively, this means that if girl g is pursuing boy b and boy b's favorite pursuer is someone else, then he can reject her, forcing her to move down to the next guy on her preference list.

4 Basic Algorithm Properties

We will prove a sequence of invariants about the matching algorithm.

Invariant 1 *If $rank_g(b) < cur_g$ then*

1. *There exists a girl g' such that $cur_{g'} = b$ and $rank_b(g') < rank_b(g)$.*

2. *$best_b < rank_b(g)$*

Intuitively, this says that if girl g ranks boy b better than the boy that she is currently pursuing, then boy b currently has a girl g' that he likes better than g. (Parts 1 and 2 in the invariant are technically different ways of expressing the same intuitive idea.)

The proof of part 1 is an easy induction argument, and the proof of part 2 follows.

Invariant 2 *For each girl g, $cur_g \leq N$.*

In other words, every girl is always pursing some boy; no girl is every rejected by all the boys and left with nothing to do.
Proof.

1. Assume not: in some reachable state, $\text{cur}_g = N + 1$.
2. `FALSE`
   (a) For all boys b, $\text{rank}_g(b) < \text{cur}_g$ (because $\text{cur}_g = N + 1$).
   (b) For all boys b, $\text{best}_b < \text{rank}_b(g)$ (by Invariant 1).
   (c) For all boys b, $\text{best-girl}_b \notin \{g, \Lambda\}$.
   (d) For all pairs of distinct boys b and b', $\text{best-girl}_b \neq \text{best-girl}_{b'}$ (by definition of best-girl).
   (e) Q. E. D. (N boys can not be matched with $N - 1$ girls).
3. Q. E. D. (by contradiction)

■

Theorem 1 *The algorithm terminates.*

Proof. The sum of the cur_g variable increases by 1 under each transtion. The sum of these variables is initially N, and the maximum sum is N^2. Therefore, the maximum number of transitions is $N(N - 1)$. ■

(To do this with a termination function, we would want a decreasing derived variable. We could always define the termination function to be $N^2 - \sum_g \text{cur}_g$. Or we could just realize that this is a variation on the notion of a termination function that also works.)

Now we show that when the algorithm terminates, everyone is paired. Recall that the algorithm terminates when no more rejections are possible; that is, every boy is pursued by at most one girl.

Invariant 3 *If a (reachable) state is terminating, then:*

1. *For every girl g, $cur\text{-}boy_g \neq \Lambda$.*

2. *For every boy b, best-girl$_b \neq \Lambda$*

Proof. The first assertion holds by Invariant 1. For the second assertion, note that in a terminating state every boy is pursued by at most one girl. Since every girl is paired with boy by the first assertion, every boy must be paired with a girl. ■

Finally, we show that the algorithm creates a stable matching.

Theorem 2 *If a (reachable) state is terminating, then the pairing defined by cur$_g$ and best$_b$ is stable.*

Proof.

1. Assume not: There exist $b, b' \in B$ and $g, g' \in G$ such that best-girl$_b = g$, cur-boy$_g = b$, best-girl$_{b'} = g'$, cur-boy$_{g'} = b'$, and b and g form a rogue couple.

2. `FALSE`

   (a) $\text{rank}_b(g') < \text{rank}_b(g) = \text{best}_b$, and $\text{rank}_{g'}(b) < \text{rank}_{g'}(b') = \text{cur}_{g'}$ (by definition of rogue couple).

   (b) $\text{best}_b < rank_b(g')$ (by Invariant 1 applied to the second part of step 2). That is, g' has been rejected by b, so b must have someone he likes better.

   (c) $\text{best}_b > \text{rank}_b(g')$ (by the first part of step 2).

   (d) Q. E. D. (by comparing 3 and 4)

3. Q. E. D. (by contradiction)

■