Adv Data Struct & Algorithm: Homework 1

Zimian Li

April 10, 2018

1. Stable matchings.

   (a) In class we established that the worst-case complexity of Boston Pool algorithm is $O(n^2)$. For each value of n describe how to construct an instance with at least $2n$ participants (n for each set in the bipartite graph) that elicit this worst case. Make sure to discuss the order in which proposals are made.

   Suppose these two sets are $M = \{m_1, m_2, ..., m_n\}$ and $W = \{w_1, w_2, ..., w_n\}$.
   The worst-case scenario could be like the preference list of every m_i is $\{w_1, w_2, ..., w_n\}$, and the preference list of every w_i is $\{m_n, m_{n-1}, ..., m_1\}$. And the proposal order should be from m_1 to m_n.

   (b) Let I be an instance of the stable matching problem. Call a pair (m, w) of $Ifeasible$ if (m, w) appears in *some* stable matching for I. Prove that while running the Boston Pool algorithm, a man m is rejected by women that are not feasible for m.

   Proof: Proved by Strong Induction.
   P(k): When the $k - th$ rejection happens, a man m is rejected by women that are not feasible for m.

   Base case P(1):
   When the first rejection happens, the scenario should be like m proposes to w and w has matched with m'. The first rejection could be w rejected m or w rejected m', actually these two are similar. let's assume w rejected m, then w matched with m', which means in $w's$ preference list, m' is prior to m, and this is the first rejection, therefore w is the first choice of m'. Now I need to prove (m, w) is not feasible:
   Assume (m, w) is feasible, then (m, w) appears in some other stable matching, in this certain stable matching, m' is matched with another woman w', however I've already known that w is the first choice of m' such that m' prefers w than w', and w prefers m' than m, so (m', w) is an *unstable* pair. That implies this matching is not stable, which is a contradiction. Hence, (m, w) is not feasible, P(1) holds.

   Assume P(1), P(2), ... P(k), then consider P(k+1):
   When the $(k + 1) - th$ rejection happens, the scenario should also be like m proposes to w and w has matched with m'. w could reject m or m', these two cases are similar.

Let's assume w rejected w. Then I can get that w prefers w' than w. I need to prove (m, w) is not feasible:
Assume (m, w) is feasible, then (m, w) appears in some other stable matching, in this stable matching, m' is matched with another woman w'. Then I get 2 pairs (m, w) and (m', w'). I've already known that w prefer m' to m, and for this is a stable matching, there's no unstable pair, m' must prefer w' over w. But another fact is that after the $(k+1) - th$ rejection happened, m' matched with w. So m' has been rejected by w' in the $i - th(1 \leq i < k+1)$ rejection, according to my inductive hypothesis, (m', w') is not feasible. It's a contradiction! So (m, w) is not feasible.
Thus, P(1), P(2), ..., P(k) $\implies$ P(k+1), and the results follow by strong induction.

(c) We are, obviously, interested in feasible pairs. For person x, let $best(x)$ denote their most desirable and feasible partner, i.e., the highest ranked candidate among the ones that are feasible. Prove that for every man m, the Boston Pool algorithm matches m with $best(m)$.

Proof: Already knew from (b) that while running the Boston Pool algorithm, a man m is rejected by women that are not feasible for m. So for an arbitrary man m, assume the Boston Pool algorithm matches him with w. That means all other women in $m's$ preference list that previous to w all rejected m. Per (b), they are all not feasible for m. So w is highest ranked and feasible one for m, which means $w = best(m)$.

(d) Similarly, let $worst(x)$ denote $x's$ least desirable and feasible partner, i.e., the lowest ranked candidate among the feasible ones. Show that for every woman w, the Boston Pool algorithm matches w with $worst(w)$.

Proof: Note the match as (m, w), I need to prove $m = worst(w)$.
1) If there are only one stable matching, it's obviously right that $m = best(w) = worst(w)$.
2) If there are more than one stable matchings. In any other stable matching, if w gets another different partner m', and m also get another different partner w'. Per what has been proved in (c), (m, w) is matched by Boston Pool algorithm, so $w = best(m)$, which means m prefers w to w'. However, this is a stable matching, therefore w prefers m' to m. Thus, I can get that in all other stable matchings, all other feasible partners of w are higher ranked than m, which means $m = worst(w)$.

(e) We saw in class that there are instances with multiple stable matchings. Can the Boston Pool algorithm return different answers depending on the order in which men are chosen in line 3?

No, it can't.
Proof: As that has been proved in (c), for every man m, the Boston Pool algorithm matches m with $best(m)$. And this $best(m)$ for each m will not change by different orders of looping.
If $best(m)$ would change in different times of Boston Pool algorithm running. Let these

$best(m)$ be $\{w_1, w_2, ..., w_n\}$, they have an order in $m's$ preference list, thus in some runing of Boston Pool algorithm, m is matched with w_i, it has to be rejected other $best(m)$. Per (b), these $best(m)$ are not feasible. That's a contradiction.
Hence, the Boston Pool algorithm will always return the same answer.

(f) Can someone do better (get a better match) by misrepresenting his/her preferences? Justify you claim.

No, he/she cant't.
Proof: Suppose in a stable matching, there are 2 pairs (m, w) and (m', w'). Assume m prefer w' to w, he want to match with w'. But it's useless to misrepresenting $m's$ preference list to make it. Because it depends on the preference list of w'.

(g) Consider the following partial preference list of 4 men and 4 women. Find a matching that is guaranteed to be stable independent of what the unspecified preferences.

$\{(Vic, Amy), (Will, Beth), (Xavi, Demi), (Yan, Cora)\}$

(h) Explain how to define a set of preferences among n men and n women that admits at lest $2^{n/2}$ different stable matchings.
Let's discuss 2 base cases first.
1) If n = 2, it could be like:
Men's preference list:

X	A	B
Y	B	A

Women's preference list:

A	Y	X
B	X	Y

In this case, there are 2 stable matchings, and $2 \geq 2^{\frac{2}{2}} = 2$.
2) If n = 3, it could be like:
Men's preference list:

X	A	B	C
Y	B	C	A
Z	C	A	B

Women's preference list:

A	Z	Y	X
B	X	Z	Y
C	Y	X	Z

In this case, there are 3 stable matchings, and $3 \geq 2^{\frac{3}{2}} \approx 2.83$.
Actullay, what I've listed is not the only case for n = 2 or 3, others are similar.
Then let's discuss if other complex cases.
1) If n is even and n > 2. In this case, the preference list of men and women should be the combination of cases that n = 2. To construct it, I need to take every 2 men and 2 women in one group, the preference lists should be like:
Men's preference list:

X_1	A_1	B_1 ...
Y_1	B_1	A_1 ...
...	...	...
$X_{n/2}$	$A_{n/2}$	$B_{n/2}$ ...
$Y_{n/2}$	$B_{n/2}$	$A_{n/2}$ ...

Women's preference list:

A_1	Y_1	X_1 ...
B_1	X_1	Y_1 ...
...	...	...
$A_{n/2}$	$Y_{n/2}$	$X_{n/2}$ ...
$B_{n/2}$	$X_{n/2}$	$Y_{n/2}$ ...

The number of stable matchings is $2^{\frac{n}{2}}$.
2) If n is odd and n > 3. The other $n - 3$ items in men and women's preference list should be same as the $n = 2$ case, but the rest 3 items should be like the $n = 3$ case. So the preference lists should be like:

Men's preference list:

X_1	A_1	B_1	...
Y_1	B_1	A_1	...
...	...	...	...
$X_{(n-3)/2}$	$A_{(n-3)/2}$	$B_{(n-3)/2}$	...
$Y_{(n-3)/2}$	$B_{(n-3)/2}$	$A_{(n-3)/2}$	...
U	D	E	F ...
V	E	F	D ...
W	F	D	E ...

Women's preference list:

A_1	Y_1	X_1	...
B_1	X_1	Y_1	...
...	...	...	...
$A_{(n-3)/2}$	$Y_{(n-3)/2}$	$X_{(n-3)/2}$	...
$B_{(n-3)/2}$	$X_{(n-3)/2}$	$Y_{(n-3)/2}$	...
D	W	V	U ...
E	U	W	V ...
F	V	U	W ...

The number of stable matchings is $3 \cdot 2^{\frac{n-3}{2}} > 2^{\frac{3}{2}} \cdot 2^{\frac{n-3}{2}} = 2^{\frac{n}{2}}$.

(i) In class we showed that the matching problem always has a stable solution when matching n boys to n girls. We consider now the problem of partitioning the $2n$ students in class into teams of two members, irrespective of gender. Each student ranks the other $2n-1$ students in class. Show that there are input instances with no stable solution.

If n = 2, one instance is like:

preference list:

A	B	D	C
B	D	A	C
C	D	A	B
D	A	B	C

For all 3 perfect matching:
1) $\{(A,B),(C,D)\}$. The unstable pair is (B,D).
2) $\{(A,C),(B,D)\}$. The unstable pair is (A,D).
3) $\{(A,D),(B,C)\}$. The unstable pair is (A,B).
So there are no stable solution.

2. Let A be a $m \times n$ matrix of integers, both positive and negative. Describe an algorithm to make all rows sums and all column sums nonnegative by changing the signs of all the values in any row(s) or any column(s) as the only operation that the algorithm can use to change the contents of A. Discuss correctness and termination.

```
1: procedure PROCESS(A, m, n)
2:     Initialize an array B[m+n] with all rows sums and all columns sums
3:     while some element of B, B[i] < 0 do
4:         if i < m then
5:             change the signs of all values in row i
6:             update B[i] and B[m] to B[m+n-1] with new sums
7:         else
8:             change the signs of all values in column (i-m)
9:             update B[i] and B[0] to B[m-1] with new sums
10:        end if
11:    end while
12: end procedure
```

Proof of correctness and termination:

Correctness: The condition of termination of my algorithm is that all rows sums and columns sums of the input matrix are nonnegative. That can make sure the correctness.

Termination: 3 facts I can get by observation:

1) The sum of all rows' sums and the sum of all columns' sums are always same. They are all equal to the sum of all elements of this matrix.

Note it as $\sum_{i=1}^{m} r_i = \sum_{j=1}^{n} c_j = \sum_{i=1,j=1}^{m,n} a_{ij} = S$.

2) Every time I get one row or column's sum from negative to positive, the sum of all elements S increases by a certain integer.

3) There's a upper bound of S, which is the sum of all elements' absolute values. Actually if there are negative values, the upper bound should be smaller than that, because we can't decrease the number of negative values by the only operation we can do.

Note it as $S \leq max(S) = \sum_{i=1,j=1}^{m,n} |a_{ij}|$.

Finally, we can conclude that after doing every operation, S increases by a certain integer, and S has an upper bound, hence, after some steps, this algorithm would terminate.