Problem Set 1 Solutions

Calvin Walker

Problem 1:

a) Let M be the partial matching of groups A and B. We start with $M = \emptyset$

1. a_1 offers b_3 which is accepted as b_3 is unmatched, $M = \{(a_1, b_3)\}$
2. a_2 offers b_1 which is accepted as b_1 is unmatched, $M = \{(a_1, b_3), (a_2, b_1)\}$
3. a_3 offers b_4 which is accepted as b_4 is unmatched, $M = \{(a_1, b_3), (a_2, b_1), (a_3, b_4)\}$
4. a_4 offers b_1 which is accepted as b_1 prefers a_4 to a_2, $M = \{(a_1, b_3), (a_3, b_4), (a_4, b_1)\}$
5. a_2 offers b_4 which is rejected as b_4 prefers a_3 to a_2, $M = \{(a_1, b_3), (a_3, b_4), (a_4, b_1)\}$
6. a_2 offers b_3 which is accepted as b_3 prefers a_2 to a_1, $M = \{(a_2, b_3), (a_3, b_4), (a_4, b_1)\}$
7. a_1 offers b_2 which is accepted as b_2 is unmatched, $M = \{(a_1, b_2), (a_2, b_3), (a_3, b_4), (a_4, b_1)\}$

The final matching is $M = \{(a_1, b_2), (a_2, b_3), (a_3, b_4), (a_4, b_1)\}$

b) We start with $M' = \emptyset$, but now group B makes the offers to group A

1. b_1 offers a_1 which is accepted as a_1 is unmatched, $M' = \{(b_1, a_1)\}$
2. b_2 offers a_3 which is accepted as a_3 is unmatched, $M' = \{(b_1, a_1), (b_2, a_3)\}$
3. b_3 offers a_3 which is accepted as a_3 prefers b_3 to a_2, $M' = \{(b_1, a_1), (b_3, a_3)\}$
4. b_4 offers a_1 which is rejected as a_1 prefers b_1 to b_4, $M' = \{(b_1, a_1), (b_3, a_3)\}$
5. b_4 offers a_4 which is accepted as a_4 is unmatched, $M' = \{(b_1, a_1), (b_3, a_3), (b_4, a_4)\}$
6. b_2 offers a_4 which is rejected as a_4 prefers b_4 to b_2, $M' = \{(b_1, a_1), (b_3, a_3), (b_4, a_4)\}$
7. b_2 offers a_2 which is accepted as a_2 is unmatched, $M' = \{(b_1, a_1), (b_2, a_2), (b_3, a_3), (b_4, a_4)\}$

The final matching is $M = \{(b_1, a_1), (b_2, a_2), (b_3, a_3), (b_4, a_4)\}$. Compared to part (a), persons b_1, b_2, b_3, and b_4 are happier with the new stable matching M', as they each have a match higher in their preference lists.

c)

Problem 2:

a) (i) $O(n^2)$
   (ii) $O(n \log n)$
   (iii) $O(n^3)$
   (iv) $O(\sqrt{n})$
   (v) $O(n \log n)$

b) $2^{\log\log n}$, $\log n$, $2^{\sqrt{\log n}}$, $2^{2\log n}$, n, $n\log n$, $\frac{n^2}{\log n}$, n^2, $n^{\log n}$, $2^n = 2^{2n}$, $n2^n$

Problem 3:

a) No, it is not possible that everyone in group A is matched with their least preferred partner. Oberve that if some a_i makes an offer to their last choice b_j, then all other people in B except b_j must be unavailable to a_i, and thus matched. There are only $n-1$ other people in A, so b_j must be unmatched, which means that no person in A has made an offer to b_j. So b_j must be a less prefered partner for some already matched person in A, so not everyone in group A is matched with their least preferred partner.

b) Yes, it is possible that everyone in group B is matched with their least preferred partner. Consider the following example.
Group A's preference lists:

a_1: $b_1 \ldots$

a_2: $b_2 \ldots$

$\vdots$

a_n: $b_n \ldots$

Group B's preference lists:

b_1: $\dots a_1$

b_2: $\dots a_2$

$\vdots$

b_n: $\dots a_n$

If we run the Gale-Shapley algorithm with group A making the offers, the resulting stable matching will be $M = \{(a_1, b_1), (a_2, b_2), \dots (a_n, b_n)\}$, so every person in B will be matched with their least preferred partner.

c) The maximum total number of offers made by group A in terms of n is $n^2 - n + 1$. Consider the following example:
Group A's preference lists:

a_1: b_1, b_2, b_3

a_2: b_2, b_1, b_3

a_3: b_1, b_2, b_3

Group B's preference lists:

b_1: a_2, a_3, a_1

b_2: a_3, a_1, a_2

b_3: $a_1, \dots$

In this example, a_1 makes n offers while a_2 and a_3 make $n-1$ offers, for a total of n^2-n+1 offers. This is because if some a_i makes their n'th offer to their last choice b_j, then all other $B\setminus\{b_j\}$ must be unavailable, and therefore have partners. So only one a_i can make n offers, while the $n-1$ others make at most $n-1$ offers, for a total of $(n-1)(n-1)+n = n^2-n+1$.

Problem 4:

a) There are n people in A and n people B, each of whom can make a total of n offers, with each offer being made at most onece, for a running time of $O(n^2)$

b) No, the matching is not guaranteed to be stable. Consider the following example:
Group A's preference lists:

a_1: b_1, b_2

a_2: b_1, b_2

Group B's preference lists:

b_1: a_2, a_1

b_2: a_2, a_1

The sequence of offers and matches would be:

1. a_1 offers b_1 which is accepted as b_1 is unmatched, $M = \{(a_1, b_1)\}$
2. b_2 offers a_2 which is accepted as a_2 is unmatched, $M = \{(a_1, b_1), (a_2, b_2)\}$

Since everyone is matched, the algorithm terminates, and the final matching is $M = \{(a_1, b_1), (a_2, b_2)\}$. However, a_2 and b_1 prefer eachother to their current partners, so the matching is not stable.

c) No the algorithm does not always result in a matching. Consider a more detailed version of the situation provided by the hint:

1. $b_{j'}$ makes an offer to $a_{i'}$ and is rejected
2. Some other $a_k \notin \{i, i'\}$ makes an offer to $b_{j'}$ and is accepted
3. Some other $b_k \notin \{j, j'\}$ makes an offer to some other $a_k \notin \{i, i'\}$
4. a_i makes an offer to $b_{j'}$ and is rejected
5. a_i makes an offer to b_j and is rejected
6. Some other $b_k \notin \{j, j'\}$ makes an offer to a_i' that is accpeted, leaving $b_{j'}$ without a partner

7. Some other $a_k \notin \{i, i'\}$ makes an offer to some other $b_k \notin \{j, j'\}$
8. $b_{j'}$ makes an offer to a_i and is accepted
9. Some other $a_k \notin \{i, i'\}$ to some other $b_k \notin \{j, j'\}$
10. Some other $b_k \notin \{j, j'\}$ makes an offer to b_j's partner that is accpeted, leaving b_j without a partner
11. Some other $a_k \notin \{i, i'\}$ to some other $b_k \notin \{j, j'\}$
12. b_j makes an offer to a_i and is rejected
13. Some other $a_k \notin \{i, i'\}$ to some other $b_k \notin \{j, j'\}$
14. Some other $b_k \notin \{j, j'\}$ makes an offer to $a_{i'}$'s partner that is accpeted, leaving $a_{i'}$ without a partner
15. $a_{i'}$ makes an offer to $b_{j'}$ which is accepted, leaving a_i and b_j without partners, both of whom have been rejected by one another. Thus, the algorithm never terminates.