CIS 675 Homework 8

Design and Analysis of Algorithms

Saket Kiran Thombre

SU ID: 899913802

NetID: sthombre

Email: sthombre@syr.edu

Collaborators for Q1, Q2:

1. Saurav Shashank Narvekar

   NetID: sanarvek

2. Chandan Pothumarthi

NetID: cpothuma

Problem 1 Stable matching (Point 6)

Part A: Suppose there are n hospitals and m students. One hospital can match with only one student, and m ≥ n. Each hospital ranks all the students and each student ranks all the hospital from most preferred to least. Definition of unstable pair in this case: Hospital h and student t form an unstable pair if h prefers t to their current student t', or has no student; and t prefers h to their current hospital h' or has no matched hospital. A matching is stable when it has no unstable pairs.

• I am interested about how many stable matchings there can be. Can you show that the upper bound and the lower bounds when m ≥ n are exactly same as when m = n? (2 point)

• Suppose John is assigned to hospital H1 in one stable matching and to hospital H2 in another. If John prefers H2 to hospital H3 and also prefers H3 to H1, must there be a stable matching where John is assigned to hospital h3? Explain. (2 point)

Part B: Let's define a new kind of instability called weakly instability. A weakly instability in a 'perfect matching: M' consists of a hospital h and a student s, such that their partners in matching M are s' and h', respectively. And one of the following holds:

• h prefers s to s' and either s prefers h to h' or is indifferent or

• s prefers h to h', and h either prefers s to s' or is indifferent. That means, the engagement of h and s is either preferred by both or preferred by one while the other is indifferent.

Does there always exist a perfect matching with no weakly instability? Either give an example of a set of hospital and student with preference lists for which every perfect matching has a weakly instability; or give an example where there can be a perfect matching with no weakly instability. (2 point)

In this problem a preference list can have equal-ranks for students or hospitals. For example a student S1 has preference list [1 : H1, 2 : (H2, H3), 3 : H4]. That means, S1 prefers H1 the most. It prefers H2, H3 equally. and prefers H4 the least.

Solution:

Part A (1): I am interested about how many stable matchings there can be. Can you show that the upper bound and the lower bounds when m ≥ n are exactly same as when m = n?

Here we take 2 conditions into consideration,

- m >= n
- m = n

To find the upper bound, we need to calculate the permutations which are needed in the case of hospitals and students together. This permutation comes together to the equation of (m!n!).

- Now if we consider the 2 cases m >= n and m = n, we can say that the permutations for both the cases will be (m!n!).

Similarly, for lower bound, we need to calculate the permutations of the hospitals only. This permutation will be (n!)

Similarly, if we now consider condition 1 and 2 that is m >= n and m = n, we can say that the permutations for them will be the same.

Hence, we can infer that the upper bound will be (m!n!) for both conditions and the lower bound will be (n!) for both the conditions.

This means that it does not matter if m>=n or m=n in the case for finding upper and lower bound in this question.

Part A (2):

Suppose John is assigned to hospital H1 in one stable matching and to hospital H2 in another. If John prefers H2 to hospital H3 and also prefers H3 to H1, must there be a stable matching where John is assigned to hospital h3? Explain.

Here we know that John will prefer hospital H2 over the hospital H3. He will also prefer hospital H3 over hospital H1.

Let us consider if that if John is assigned to a hospitals H1 and H2 in a stable pair condition, that is we will get (John, H1) and (John, H2) as pairs, both of these pairs become unstable. This will directly contradict definition of stable matching with stable pairs.

Hence, we can say that to get a stable pair, there must exist a condition where John will be assigned to hospital H3.

A stable matching is a pair which will have no unstable pairs. It is natural and desirable for both the hospital and student. Individual interests will always prevent any hospital student side deals.

In our case, John will prefer hospital H2 over hospital H3 and also prefer hospital H3 over hospital H1. This will create 2 unstable pairs (John,H1) and (John,H2).

This also means that there will exist a stable matching where John will get is preferred hospital H3.

Part B:

Let us take into consideration an example where we have a set of hospitals and students with their preference lists respectively.

Every perfect match in the set will have a weak instable pair.

Hospitals and their preference list:

Hospital H1:	S1	S2	S3
Hospital H2:	S2	S1	S3

Students and their preference list:

Student S1	Hospital H2	Hospital H1
Student S2	Hospital H1	Hospital H2
Student S3	Hospital H1	Hospital H2

In this example, we can have the following perfect matches:

- (Hospital H1, S1)
- (Hospital H2, S2)

Then we also get a pair of weakly instability pairs:

- (Hospital H1, S2)
- (Hospital H2, S1)

This will happen because Hospital H1 will prefer Student S2 and S2 will also prefer hospital H1 and hospital H2.

Now let us take an example with no weakly instability pairs:

Hospital H1:	S1	S2	S3
Hospital H2:	S2	S1	S3

And

Student S1	Hospital H1	Hospital H2
Student S2	Hospital H1	Hospital H2
Student S3	Hospital H1	Hospital H2

This example will only create stable pairs:

- (Hospital H1, S1)
- (Hospital H2, S2)

This will happen because both Hospital H1 and S2 and Hospital H2 and S1 are going to be indifferent to each other.

Problem 2 Circular Vertex Cover Problem (Point 4)

Given a directed graph G = (V, E), a circular vertex cover is a subset of vertices such that every cycle in G passes through at least one of these vertices. (For example, the graph in Figure 1 has a circular vertex cover of size 2 (grey colored vertices))

Definition of Circular Vertex Cover Problem: Given a directed-graph G and an integer k, does G contain a circular vertex cover of size at most k? Show that Circular Vertex Cover Problem is NP-Hard. Reduce from K size Vertex cover problem. Vertex Cover was for un-directed graph. You need to explain the reduction.

Solution:

```
if Root is a part of the vertex;
         T(R) = 1 + T(R->LT) + T(R->RT)

       if(R->LT)
         T(R) += 1 + T(R->LT->right) + T(R->LT->LT)

       if(R->RT)
         T(R) += 1 + T(R->RT->RT) + T(R->RT->LT)
```

Root = R

Left = LT

Right = RT

One if condition is used when root is part of the vertex cover and other is used when root is not part of it. Then, we check for root->left and root->right.

We use 2 conditions when root will be a part of the vertex:

- `T(R) += 1 + T(R->LT->right) + T(R->LT->LT)`
- `T(R) += 1 + T(R->RT->RT) + T(R->RT->LT)`

Then we check for root in left and right.

Pseudocode

```
void treevertex(vector<int>array1[],vectorInt<int>dpMemory,int SRC,int
ParentVertex){
 for(auto i:adj[SRC])
 {

    if(ParentVertex!=i)
```

```
      treevertex(array1,dpMemory,i,SRC);
}
 for(auto i:array1[SRC])
{
    if(ParentVertex!=i)
    {
      dpMemory[SRC][0] += dpMemory[i][1];
      dpMemory[SRC][1] += min(dp[i][1], dpMemory[i][0]);
    }
}
}
```

Then we check for root in left and right.

Array1 is the adjacency list which will contain all connected vertices.

dpMemory is a 2D array which will store all subproblem results

SRC is Start vertex

The runtime of this algorithm is O(n).