The solution of stable matching problem

陈柯昊

1160300823

2017 年 5 月 24 日

Contents

1 Introduction

1.1 What is stable matching problem

Definition 1 (Stable Matching Problem) *Given M men and W women.Each man ranks all women and every woman ranks all men.We want to order the men and women into pairs so that a. each man and each woman appears in exactly one pair b. there is no (m, w) and(m' , w') pairs such that man m prefers w' to w and w' prefers m to m' , as a picture:*

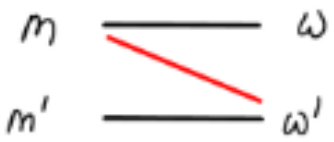

Figure 1: Introduction

If this was the case man m and woman w' would both be better off forming a (m, w') pair instead.

1.2 Applications

Algorithms for finding solutions to the stable marriage problem have applications in a variety of real-world situations, perhaps the best known of these being in the assignment of graduating medical students to their first hospital appointments. In 2012, the Nobel Prize in Economics was awarded to Lloyd S. Shapley and Alvin E. Roth "for the theory of stable allocations and the practice of market design." An important and large-scale application of stable marriage is in assigning users to servers in a large distributed Internet service.Billions of users access web pages, videos, and other services on the Internet, requiring each user to be matched to one of (potentially) hundreds of thousands of servers around the world that offer that service.

A user prefers servers that are proximal enough to provide a faster response time for the requested service, resulting in a (partial) preferential ordering of the servers for each user. Each server prefers to serve users that it can with a lower cost, resulting in a (partial) preferential ordering of users for each server. Content delivery networks that distribute much of the world's content and services solve this large and complex stable marriage problem between users and servers every tens of seconds to enable billions of users to be matched up with their respective servers that can provide the requested web pages, videos, or other services.

2 How to solve a stable marriage problem?

We usually try to solve a stable marriage problem by using Gale-Shapley Algorithm.This is a really tricky algorithm,which can easily solve these problems.

2.1 What is Gale Shapley Algorithm?

The Gale Shapley Algorithm was raised by Gale Shapely in 1962,in order to solve the stable marriage problem.Stable marriage problem is the problem of finding a stable matching between two equally sized sets of elements given an ordering of preferences for each element. A matching is a mapping from the elements of one set to the elements of the other set. A matching is not stable if: a. There is an element A of the first matched set which prefers some given element B of the second matched set over the element to which A is already matched,and b.B also prefers A over the element to which B is already matched. In other words,a matching is stable when there does not exist any match **(A,B)** by which both A and B would be individually better off than they are with the element to which they are currently matched.

2.2 Solution

The Gale-Shapley algorithm involves a number of "rounds".In the first round, first a) each unengaged man proposes to the woman he prefers most, and then b) each woman replies "maybe" to her suitor she most prefers and "no" to all other suitors.She is then provisionally "engaged" to the suitor she most prefers so far,and that suitor is likewise provisionally engaged to her.In each subsequent round, first a)each unengaged man proposes to the most-preferred woman to whom he has not yet proposed(regardless of whether the woman is already engaged), then b)each woman replies"maybe"if she is currently not engaged or if she prefers this guy over her current provisional partner(in this case,she rejects her current provisional partner who becomes unengaged).The provisional nature of engagements preserves the right of an already-engaged woman to "trade up"(and ,in the process,to "jilt"her until-then partner).This process is repeated until everyone is engaged.

2.2.1 Pseudocode

Gale-Shapley Algorithm

Initialize all $m \in M$ *and* $w \in W$ *to free*

1 **while**$\exists$ free man m m
2 // who still has a woman w to propose to
3 w = first woman on man's list to whom m has not yet proposed
4 **if** w is *free*
5 (m,w) become *engaged*
6 **else** some pair(m',w)already exists
7 **if** w prefers m to m'
8 m' becomes *free* (m,w) become *engaged* **else** (m',w) remain *engaged*

2.2.2 Time Complexity

The runtime complexity of this algorithm is $O(n^2)$ where n is number of men or women.

3 Conclusion and analysis

There're plenty of methods to implement the Gale Shapley algorithm,I choose to use linkedlist and array to implement the algorithm.

3.1 Analysis

Below are the results of the two different methods.

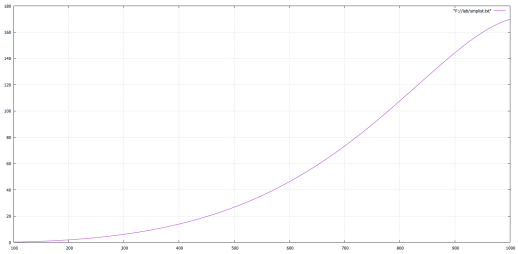

Figure 2: linkedlist

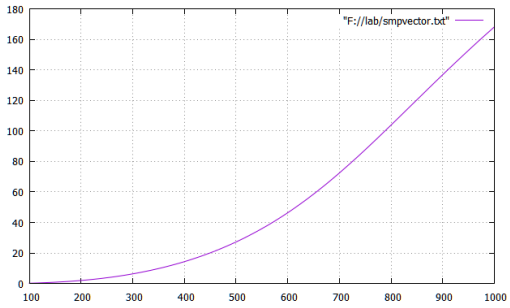

Figure 3: vector

And the next chart is the comparison of two methods.

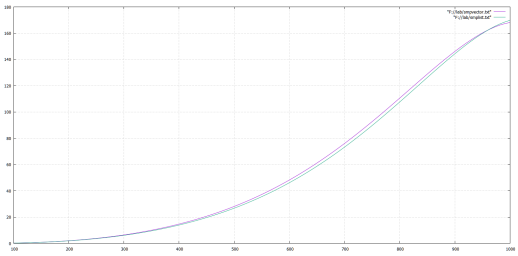

Figure 4: comparison

3.2 Conclusion

During my studying,I found more interesting facts about this algorithm.For example,this algorithm can make every man engage with his best available partner,however women can only get the worst partner. What's more,though the algorithm seems perfect,when put into practice,it may have some potential problems. If a woman knows other women's list, she may refuse the man who should not be refused ,to wait for a better result. And the algorithm also has its limitation,it cannot solve a problem that "2n people(not about gender) stable matching problem". In fact, a simple situation is distributing dormitory.Assuming that each dormitory lives two people, it is known that each student of 2n students evaluates the preferences of the remaining 2n-1 students, and how to find a stable dormitory assignment.At this point, Gale-Shapley algorithm is no longer useful. In fact, the dormitory distribution problem is likely to simply do not exist with a stable match. For example, there are A, B, C, D four people, including A to B in the first, B to C in the first row, C ranked first in the A, and the three of them put the D in the final. It is easy to see that there must be no stable dormitory allocation scheme at this time. If A, D with the dormitory, B, C with the dormitory, then C will think that A is a better roommate (because C to A ranked first), while A will think that C is a better roommate (because he put D In the end). Similarly, B, D with the dormitory or C, D with the dormitory are not enough, so the stability of the dormitory allocation is not there. At this point, redefining the merits of dormitory distribution is a more fundamental

problem.

4 Appendix

4.1 Code

4.1.1 Array

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <stack>
#include <queue>
#include <vector>
using namespace std;
int N,M;
vector<int> random_value(vector<int>&startArray)
{
        vector<int> resultArray;
        for (int i = 0; i<M; i++)
        {
                startArray.push_back(i);
        }
        for(int j=0;j<N;j++)
        {
                for (int i = 0; i < M; i++)
                {
   int seed = rand()%(startArray.size() - i);
resultArray.push_back(startArray[seed]);
swap(startArray[seed],startArray[startArray.size() - i - 1]);
}
}
```

```
cout << "Get it!" << endl;
return resultArray;
}
vector<int> random_value_f(vector<int> &startArray)
{
vector<int> resultArray;
for (int i = 0; i<N; i++)
{
startArray.push_back(i);
}
for(int j=0;j<M;j++)
for (int i = 0; i < N; i++)
{
int seed = rand()%(startArray.size() - i);
 resultArray.push_back(startArray[seed]);
  swap(startArray[seed],startArray[startArray.size() - i - 1]);
  }
cout << "Get it!" << endl;
return resultArray;
}
void input_data(vector<int> &v_1, vector<int> &v_2,
vector<bool> &judge, vector<bool> &male_st, vector<bool> &famale_st)
{
int k;
cout << "please input men's amount,women's amount:" << endl;
cin >> N >> M;
cout << "Get men's list:";\\
v_1 = random_value(v_1);
cout << "then the women's list:";
v_2 = random_value_f(v_2);
for (int i = 0; i<N; i++)
```

```
{
        male_st.push_back(0);
        for (int j = 0; j<M; j++)
        {
        judge.push_back(0);
        }
        }
for (int i = 0; i<M; i++)famale_st.push_back(0);
}
int if_famale(int f,vector<bool> &judge,int i,vector<int> &v_2)
{
int pos=-1;
for(int j=0;j<N-1;j++)
{
if(v_2[f*N+j]==i)
 {
   pos=j+1;
    break;
     }
      }
      if(pos!=-1)
       {
        while(!judge[(pos++)*M+f]&&(pos-1)<N);
        if(pos!=0)
        return pos-1;
        else return 0;
       }
       return -1;
}
bool judge_male_all_engaged(vector<bool> &male_st)
{
```

```
for(int i=0;i<N;i++)
{
        if(!male_st[i])return false;
}
return true;
}
void release(vector<int> &v_2,vector<bool> &judge,
vector<bool> &male_st,int counter,int i)
{
        for(int j=N-1;j>=0;j--)
        {
                int d=v_2[i*N+j];
                 if(judge[d*M+i])
                 {
                         judge[d*M+i]=0;
                         male_st[d]=0;
                         if((--counter)==1)return;
                 }
        }
}

void release_famale(vector<int> &v_2,
vector<bool> &judge,vector<bool> &male_st)
{
        int counter=0;
        for(int i=0;i<M;i++)
        {
             counter=0;
                 for(int j=0;j<N;j++)
                 {
                 if(judge[j*M+i])
```

```
                    {
                    counter++;
                    }
                    }
          if(counter>1)
          {
          release(v_2,judge,male_st,counter,i);
          }
          }
}
void stable_marraige(vector<int> &v_1,vector<int> &v_2,
vector<bool> &judge,vector<bool> &male_st,vector<bool> &famale_st)
{
          int k=0,i,male,famale,ka=0;
          while(1)
          {
if(!male_st[ka])
{
for(i=0;i<M;i++)
{
                    if(v_1[ka*M+i]!=-1)
                    {
                    famale=v_1[ka*M+i];
judge[ka*M+famale]=1;
male_st[ka]=1;
v_1[ka*M+i]=-1;
if(!famale_st[famale])
famale_st[famale]=1;
else
{
male=if_famale(famale,judge,ka,v_2);
```

```
if(male!=-1)
{
judge[male*M+famale]=0;
male_st[male]=0;
}
}
break;
}
}
release_famale(v_2,judge,male_st);
}
if(judge_male_all_engaged(male_st))break;
k++;
ka=k%N;
}
}
void output_marriage(vector<bool> judge)
{
        for(int i=0;i<N;i++)
        for(int j=0;j<M;j++)
        {
if(judge[i*N+j])cout<<"man "<<i<<" engaged with woman "<<j<<endl;
        }
}

int main()
{
        vector<int> v_1,v_2;
        vector<bool> judge,male_st,famale_st;
        input_data(v_1,v_2,judge,male_st,famale_st);
        cout<<endl;
```

```
        stable_marraige(v_1,v_2,judge,male_st,famale_st);
        output_marriage(judge);
        return 0;
}
```

4.1.2 LinkedList

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <stack>
#include <queue>
#include <vector>
#include<time.h>
#include<stdio.h>
using namespace std;

int N,M;
vector<int> random_value(vector<int> &startArray)
{
        vector<int> resultArray;
        for (int i = 0; i<M; i++)
        {
                startArray.push_back(i);
        }
        for(int j=0;j<N;j++)
        {
                for (int i = 0; i < M; i++)
                {
 int seed = rand()%(startArray.size() - i);
 resultArray.push_back(startArray[seed]);
swap(startArray[seed],startArray[startArray.size() - i - 1]);
```

```
 }
     }
 cout << "Get it!" << endl;
return resultArray;
}

vector<int> random_value_f(vector<int> &startArray)
{
        vector<int> resultArray;
        for (int i = 0; i<N; i++)
{
startArray.push_back(i);
}
for(int j=0;j<M;j++)
 for (int i = 0; i < N; i++)
 {
 int seed = rand()%(startArray.size() - i);
 resultArray.push_back(startArray[seed]);
 swap(startArray[seed],startArray[startArray.size() - i - 1]);
 }
        cout << "Get it!" << endl;
        return resultArray;
}
void input_data(vector<int> &v_1, vector<int> &v_2,
vector<bool> &judge, vector<bool> &male_st, vector<bool> &famale_st)
{
        int k;
        cout << "please input men's amount,women's amount:" << endl;
        cin >> N >> M;
        cout << "Get men's list:";
        v_1 = random_value(v_1);
```

```
        cout << "then the women's list:";
        v_2 = random_value_f(v_2);
        for (int i = 0; i<N; i++)
        {
                male_st.push_back(0);
                for (int j = 0; j<M; j++)
                {
                        judge.push_back(0);
                }
        }
        for (int i = 0; i<M; i++)famale_st.push_back(0);
}
int if_famale(int f,vector<bool> &judge,int i,vector<int> &v_2)
{
        int pos=-1;
        for(int j=0;j<N-1;j++)
        {
        if(v_2[f*N+j]==i)
         {
          pos=j+1;
           break;
            }
        }
        if(pos!=-1)
        {
         while(!judge[(pos++)*M+f]&&(pos-1)<N);
         if(pos!=0)
         return pos-1;
         else return 0;
        }
        return -1;
```

```
}
bool judge_male_all_engaged(vector<bool> &male_st)
{
        for(int i=0;i<N;i++)
        {
                if(!male_st[i])return false;
        }
        return true;
}
void release(vector<int> &v_2,vector<bool> &judge,
vector<bool> &male_st,int counter,int i)
{
        for(int j=N-1;j>=0;j--)
        {
                int d=v_2[i*N+j];
                if(judge[d*M+i])
                {
                        judge[d*M+i]=0;
                        male_st[d]=0;
                        if((--counter)==1)return;
                }
        }
}
void release_famale(vector<int> &v_2,
vector<bool> &judge,vector<bool> &male_st)
{
        int counter=0;
        for(int i=0;i<M;i++)
        {
                counter=0;
                for(int j=0;j<N;j++)
```

```
                {
                        if(judge[j*M+i])
                        {
                                counter++;
                        }
                }
                if(counter>1)
                {
                        release(v_2,judge,male_st,counter,i);
                }
        }
}

void stable_marraige(vector<int> &v_1,vector<int> &v_2,
vector<bool> &judge,vector<bool> &male_st,vector<bool> &famale_st)
{
        int k=0,i,male,famale,ka=0;
        while(1)
        {
                if(!male_st[ka])
{
for(i=0;i<M;i++)
{
                if(v_1[ka*M+i]!=-1)
                {
                famale=v_1[ka*M+i];
judge[ka*M+famale]=1;
male_st[ka]=1;
v_1[ka*M+i]=-1;
if(!famale_st[famale])
 famale_st[famale]=1;
```

```
 else
  {
  male=if_famale(famale,judge,ka,v_2);
   if(male!=-1)
   {
   judge[male*M+famale]=0;
    male_st[male]=0;
    }
    }
    break;
         }
         }
         release_famale(v_2,judge,male_st);
         }
         if(judge_male_all_engaged(male_st))break;
         k++;
         ka=k%N;
         }
}
void output_marriage(vector<bool> judge)
{
         for(int i=0;i<N;i++)
                  for(int j=0;j<M;j++)
{
if(judge[i*N+j])cout<<"man "<<i<<" engaged with woman "<<j<<endl;
}
}

int main()
{
clock_t start, finish;
```

```
double duration;
FILE *stream;
stream = fopen("F://lab/smpvector.txt", "a+" );
vector<int> v_1,v_2;
vector<bool> judge,male_st,famale_st;
input_data(v_1,v_2,judge,male_st,famale_st);
cout<<endl;
start = clock();
stable_marraige(v_1,v_2,judge,male_st,famale_st);
//output_marriage(judge);
finish = clock();
duration = (double)(finish - start) / CLOCKS_PER_SEC;
fprintf(stream, "%d %f\n",N,duration );
fclose(stream);
return 0;
}
```