İ.T.Ü.

Bilgisayar ve Bilişim Fakültesi

Bilgisayar Mühendisliği Bölümü

ANALYSIS OF ALGORITHMS
HW1

Aykut Akın

040080177

23.03.2011

Description of how my program should be compiled and run:

I use Microsoft Visual C++ 2010 Express to develop the program. Program can be compiled and run on Linux/Unix using g++.

You should write a command like that for compile the Man optimal Gale-Shapley algorithm on Linux terminal.

```
g++ GS.cpp Heap.cpp MWP_Gale_Shapley.cpp –o GS
```

You should write a command like that for compile the Woman optimal Gale-Shapley algorithm on Linux terminal.

```
g++ GSW.cpp Heap.cpp MWP_Gale_Shapley.cpp –o GSW
```

You should write a command like that for compile the Man optimal Gale-Shapley algorithm with wealth on Linux terminal.

```
g++ GS.cpp Heap.cpp MWP_Gale_Shapley.cpp –o GSP
```

You should give a command like that for use the Man optimal Gale-Shapley algorithm.

```
./GS –i ppm_5.txt ppw_5.txt –o GS_5_out.txt
```

You should give a command like that for use the Woman optimal Gale-Shapley algorithm.

```
./GSW –i ppm_5.txt ppw_5.txt –o GSW_5_out.txt
```

You should give a command like that for use the Man optimal Gale-Shapley algorithm with wealth.

```
./GSP –i ppm_5.txt ppw_5.txt –w1 mw_5.txt ww_5.txt –o GSP_5_out.txt
```

Note: You should add `<direct.h>` for Windows and exclude `<sys/stat.h>`, `<sys/types.h>` libraries.

My cpp files & header files

This part of the report contains explanation about classes and necessary methods of the program.

__________________________________MWP_Gale_Shapley.h______________________________

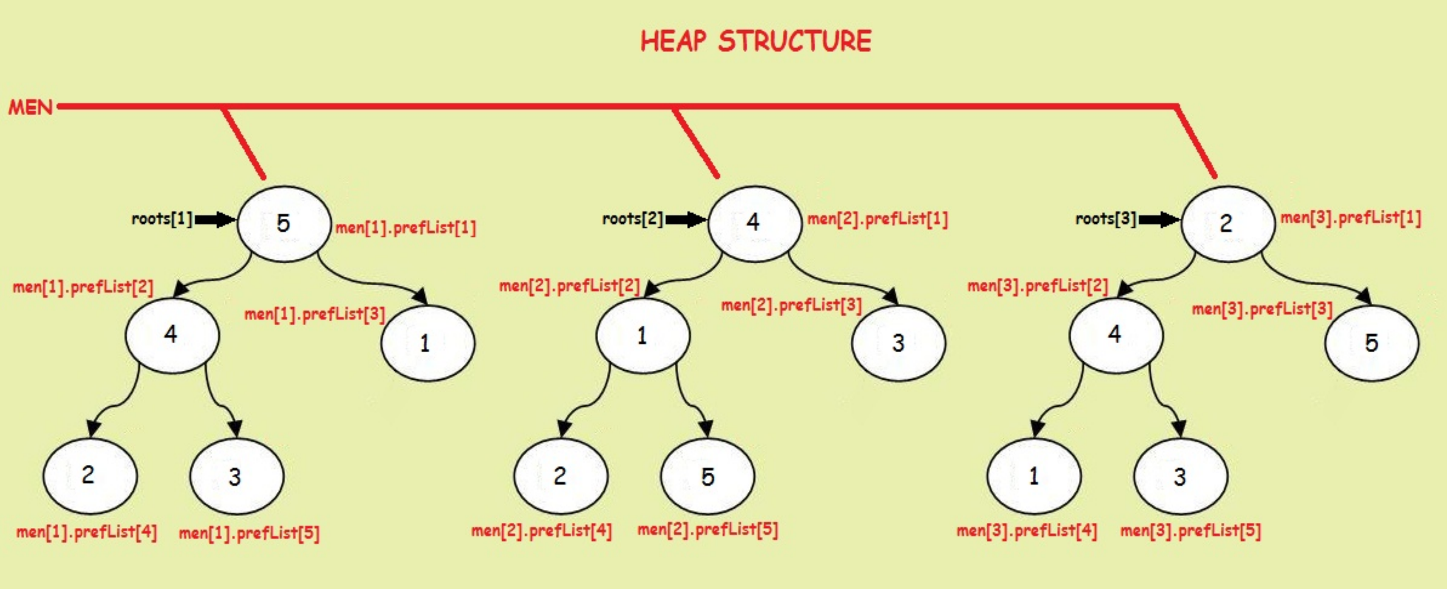
```
#ifndef __GALE_H__
#define __GALE_H__

#include "Heap.h"                               //Libraries

       void gale_shapley(Heap &, Heap &);       //Gale-Shapley algorithm for men optimal
       void w_gale_shapley(Heap &, Heap &);     //Gale-Shapley algorithm for women optimal
       void p_gale_shapley(Heap &, Heap &);     //Gale-Shapley algorithm with wealth
       bool is_free_person(Heap&);              //Check for free person

#endif
```

____

Heap.h

```
#ifndef __HEAP_H__
#define __HEAP_H__

#include <iostream>
#include <iomanip>
#include <cstdlib>       Libraries
#include <fstream>
#include <string>
#include <direct.h>

#define SIZE 100

using namespace std;

class Heap{

      int heapSize;                          //Heap size for a person
      int prefList[SIZE+1][SIZE+1];          //Preference list for every people
      void readData(string fileName);        //Read data files
      void readwData(string wfileName,string num);//Read wealth files

public:

      int engagements[SIZE+1];               //Engagements info for every people
      int *count;                            //Propose number for every men
      int **roots;                           //Root for every heap
      int **inverse;                         //Inverse prefList for quick look(just
                                               for women)
      int *wealth;                           //Wealth for every people

      Heap(string optimal,string fileName, string wfileName=””, string wNum=””); //Constructor
      ~Heap();                               //Destructor

      int getHeapSize();                     //Return heap size
      void writeData(string fileName, int time); //Write output file
};

#endif
```

Example of heap structure for men



Heap.cpp

```
#include "Heap.h"                               //Libraries

Heap::Heap(string optimal,string fileName, string wfileName, string wNum){//Constructor

       if(fileName[4] == '5')                   //Find heapsize
              heapSize = 5;
       if(fileName[6] == '.')
              heapSize = 10;
       if(fileName[4] == '2')
              heapSize = 20;
       if(fileName[7] == '.')
              heapSize = 100;

       for(int i=1; i<=heapSize; i++)           //Every person is single at the start
              engagements[i] = 0;

       roots = new int* [heapSize+1];           //Root pointer for every heap
       readData(fileName);                      //Read data from a file

       char o;
       if(optimal.compare("GSW")==0)            //Woman optimal or Man optimal?
              o = 'w';
       else
              o = 'm';

       if(fileName[2] == o){                    //Get space for count variable
              count = new int[heapSize+1];      //according to who is going to
                                                //propose(according to optimality)
              for(int i=1; i<=heapSize; i++)
                     count[i] = 0;              //Every person propose noone at the start
              inverse = NULL;
       }
       else{
              inverse = new int* [heapSize+1];  //Get space for count variable
              for(int i=0; i<=heapSize; i++)    //according to who is going to
                     inverse[i] = new int [heapSize+1]; //propose(according to
                                                       optimality)
              for(int i=1; i<=heapSize; i++)
                     for(int j=1; j<=heapSize; j++)
                            inverse[i][prefList[i][j]] = j; //Fill inverse array
              count = NULL;
       }

       if(wfileName.compare("")!=0 && wNum.compare("")!=0){
              wealth = new int[heapSize+1];
              readwData(wfileName,wNum);
       }
}

Heap::~Heap(){                                  //Destructor
       if(count)
              delete[] count;
       if(inverse){
              for (int i=0; i<=heapSize; ++i)
                     delete [] inverse[i];
               delete [] inverse;
       }
       delete [] roots;
}

void Heap::readData(string fileName){
       string filename = "data/"+fileName;
       ifstream infile (filename.c_str());
```

```
        int person=1, num=1;

        if(!infile.is_open())
                cout << "Unable to open file"<<endl;
        else{
                while(infile >> prefList[person][num]){ //Fill prefList

                        if(num == 1)  //Root is first preference at the start
                                roots[person] = &prefList[person][num];
                        if(num < heapSize)
                                num++;
                        else{
                                person++;
                                num=1;
                        }
                }

                infile.close();
        }
}

void Heap::readwData(string wfileName,string num){ //Read wealth file and put into wealth
        string filename = "data/wealth"+num.substr(2,1)+"/"+wfileName;
        ifstream infile (filename.c_str());//open file

        int person=1;

        if(!infile.is_open())
                cout << "Unable to open file"<<endl;
        else{
                while(infile >> wealth[person])
                        person++;
                infile.close();
        }
}

void Heap::writeData(string fileName, int time)
{
        mkdir("./output", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
        string filename = "output/"+fileName;
        ofstream outfile(filename.c_str());
        int i=1;
        int countsum=0;

        if (!outfile.is_open())
                cout << "Unable to open file"<<endl;

        else{
                while(i <= heapSize){           //Write to output file
                        countsum += count[i];
                        outfile << "  ";
                        outfile << i << "    ";
                        outfile << engagements[i++] << endl;
                }
                outfile << endl << "time = "<< time << " ms"<<endl;
                outfile <<"count sum = "<< countsum;
                outfile.close();
        }
}

int Heap::getHeapSize(){
        return heapSize;
}
```

MWP_Gale_Shapley.cpp

```cpp
#include "MWP_Gale_Shapley.h"                         //Libraries

void gale_shapley(Heap& men, Heap& women){     //Man optimal Gale-Shapley algoritm

       int number = men.getHeapSize();
       srand ( time(NULL) );
       while(is_free_person(men)){       //While there is a single man
              int i = rand() % number + 1;  //Choose such a man
              if(men.engagements[i] == 0){      //If the man is single
                    bool free = true;
                    while(free && men.count[i] < number){
                           men.count[i]++;
                           int woman = *men.roots[i]; //Next preference from root
                           men.roots[i]++;

                           if(women.engagements[woman]==0){ //Preferred woman engaged?
                                  men.engagements[i] = woman; //If no make engagement
                                  women.engagements[woman] = i;
                                  free = false;
                           }
                                  //If woman is engaged, prefer which man?
                   else if(women.inverse[woman][women.engagements[woman]]>women.inverse[woman][i]) {
                                  men.engagements[women.engagements[woman]] = 0; //If new man
                                  men.engagements[i] = woman;                     //make
                                  women.engagements[woman] = i;                   //engagement
                                  free = false;
                           }
                    }
              }
       }
}

void w_gale_shapley(Heap& women, Heap& men){    //Woman optimal Gale-Shapley algoritm

       int number = men.getHeapSize();
       srand ( time(NULL) );
       while(is_free_person(women)){     //While there is a single woman
              int i = rand() % number + 1;  //Choose such a woman
              if(women.engagements[i] == 0){     //If the woman is single
                    bool free = true;
                    while(free && women.count[i] < number){
                           women.count[i]++;
                           int man = *women.roots[i]; //Next preference from root
                           women.roots[i]++;

                           if(men.engagements[man]==0){      //Preferred man engaged?
                                  women.engagements[i] = man; //If no make engagement
                                  men.engagements[man] = i;
                                  free = false;
                           }
                                  //If man is engaged, prefer which woman?
                           else if(men.inverse[man][men.engagements[man]] > men.inverse[man][i]){
                                  women.engagements[men.engagements[man]] = 0; //If new woman
                                  women.engagements[i] = man;                  //make
                                  men.engagements[man] = i;                    //engagement
                                  free = false;
                           }
                    }
              }
       }
}

void p_gale_shapley(Heap& men, Heap& women){
    int number = men.getHeapSize();
    bool stop=false;
    srand ( time(NULL) );
    while(is_free_person(men) && !stop){            //There is a free man
        int i = rand() % number + 1;
        if(men.engagements[i] == 0){                //If a man is not engaged
            if(men.wealth[i] !=0 ){                 //and he has a wealth
                bool free = true;
                while(free && men.count[i] < number){//Make engagement with proper woman
                    men.count[i]++;
                    int woman = *men.roots[i];
                    men.roots[i]++;

                    if(women.engagements[woman]==0){
                        men.engagements[i] = woman;
                        women.engagements[woman] = i;
                        men.wealth[i]--;
                        free = false;
                    }

        else if(women.inverse[woman][women.engagements[woman]] > women.inverse[woman][i]){
                        if(women.wealth[woman] !=0 ){

        men.engagements[women.engagements[woman]] = 0;
                            men.engagements[i] = woman;
                            women.engagements[woman] = i;
                            women.wealth[woman]--;
                            free = false;
                        }
                    }
                }
            }
        }
        else{ //Check unengament and non-wealth
            for(int j=1; j<=number; j++){
                if(men.engagements[j]==0 && men.wealth[j]!=0)
                    break;
                if(j==men.getHeapSize())
                    stop = true;
            }
        }
    }
}

bool is_free_person(Heap& person){

    bool free = false;

    for(int i=1; i<=person.getHeapSize(); i++){ //Search for every people
        if(person.engagements[i] == 0){          //If there is single person
            free = true;                         //free is true
            break;
        }
    }

    return free;
}
```

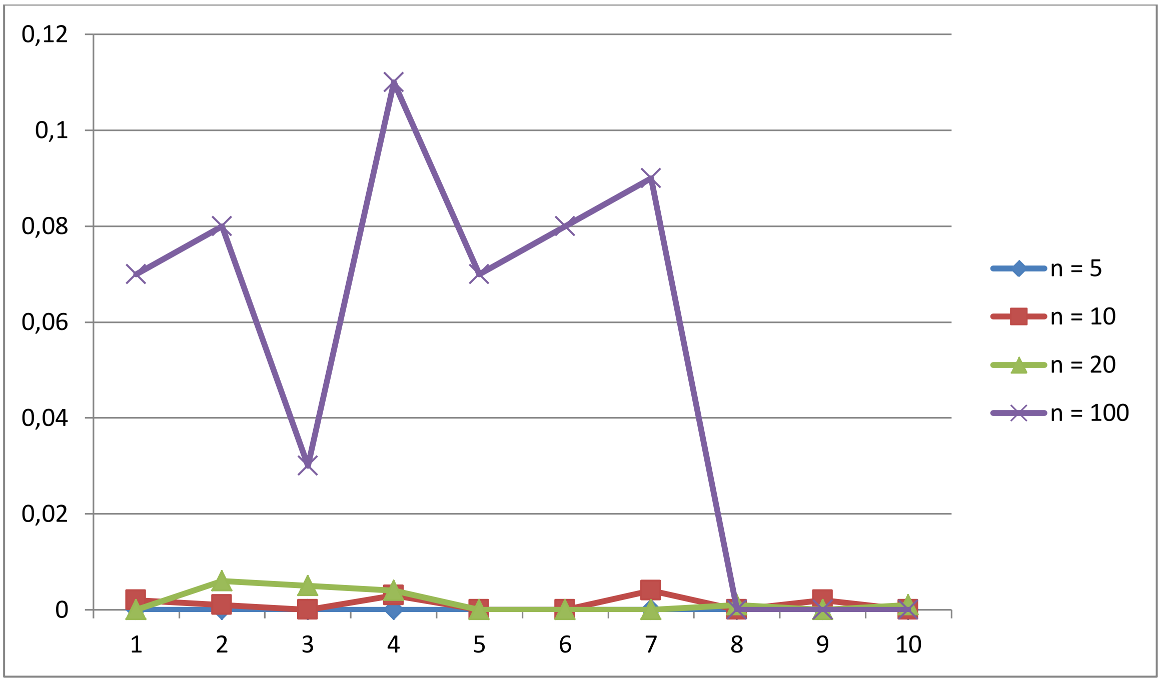
Question 1:

a) GS.cpp
Heap.cpp
MWP_Gale_Shapley.cpp } files are the answer of this question
Heap.h
MWP_Gale_Shapley.h

b) For this part of the homework I made a loop to calculate time. The algorithm works really fast, so we always see 0 ms in the file. Because of that, I made a loop for 1000 times and I divide the result 1000 and calculate the running time of the algorithm.

n = 5	**n = 10**	**n = 20**	**n = 100**
0	0,002	0	0,07
0	0,001	0,006	0,08
0	0	0,005	0,03
0	0,003	0,004	0,11
0	0	0	0,07
0	0	0	0,08
0	0,004	0	0,09
0	0	0,001	0
0	0,002	0	0
0	0	0,001	0

*All the result are millisecond



Question 2:

a) GSW.cpp
Heap.cpp
MWP_Gale_Shapley.cpp files are the answer of this question
Heap.h
MWP_Gale_Shapley.h

b) We can compare summation of count numbers for every men and women. Because count number for each person tells us how many times they made proposal and got married with which preference.

ppm_5.txt – ppw_5.txt
Man optimal count sum = 11
Woman optimal count sum = 11

For this files man optimal solution and woman optimal solution are exact same.

ppm_10.txt – ppw_10.txt
Man optimal count sum = 35
Woman optimal count sum = 35

For this files man optimal solution and woman optimal solution are exact same.

ppm_20.txt – ppw_20.txt
Man optimal count sum = 64
Woman optimal count sum = 51

For this files woman optimal solution is better than man optimal solution.

ppm_100.txt – ppw_100.txt
Man optimal count sum = 486
Woman optimal count sum = 432

For this files woman optimal solution is better than man optimal solution.

Question 3:

a) GSP.cpp
Heap.cpp
MWP_Gale_Shapley.cpp files are the answer of this question
Heap.h
MWP_Gale_Shapley.h

b) I test my code with all the wealth files about 15 times. I wrote different solutions that I found.

Wealth File 1

	Work 1	**Work 2**	**Work 3**
	Woman	**Woman**	**Woman**
Man 1	5	5	5
Man 2	1	3	1
Man 3	0	1	0
Man 4	4	4	4
Man 5	2	2	2
Count	**8**	**11**	**7**

3rd solution is better than 1st solution, 1st solution is better than 2nd solution.

Wealth File 2

	Work 1	**Work 2**	**Work 3**
	Woman 1	**Woman 2**	**Woman 3**
Man 1	5	5	5
Man 2	0	1	1
Man 3	1	0	0
Man 4	4	4	4
Man 5	2	2	2
Count	**10**	**8**	**7**

3rd solution is better than 2nd solution, 2nd solution is better than 1st solution.

Wealth File 3

	Work 1	**Work 2**	**Work 3**
	Woman 1	**Woman 2**	**Woman 3**
Man 1	5	5	5
Man 2	3	0	1
Man 3	1	1	0
Man 4	4	4	4
Man 5	2	2	2
Count	**11**	**10**	**7**

3rd solution is better than 2nd solution, 2nd solution is better than 1st solution.

Wealth File 4

	Work 1	**Work 2**	**Work 3**
	Woman 1	**Woman 2**	**Woman 3**
Man 1	5	5	5
Man 2	1	0	1
Man 3	0	1	0
Man 4	4	4	4
Man 5	2	2	2
Count	**8**	**10**	**7**

3rd solution is better than 1st solution, 1st solution is better than 2nd solution.

Wealth File 5

	Work 1	**Work 2**	**Work 3**
	Woman	**Woman**	**Woman**
Man 1	5	5	5
Man 2	0	3	1
Man 3	1	1	0
Man 4	4	4	4
Man 5	2	2	2
Count	**10**	**11**	**7**

3rd solution is better than 1st solution, 1st solution is better than 2nd solution.