**Tugas Praktikum**

**Analisis Algoritma**

**(*Stable Matching Problem - Gale Shapley Algorithm*)**



Disusun oleh :

Muhammad Islam Taufikurahman

(140810160062)

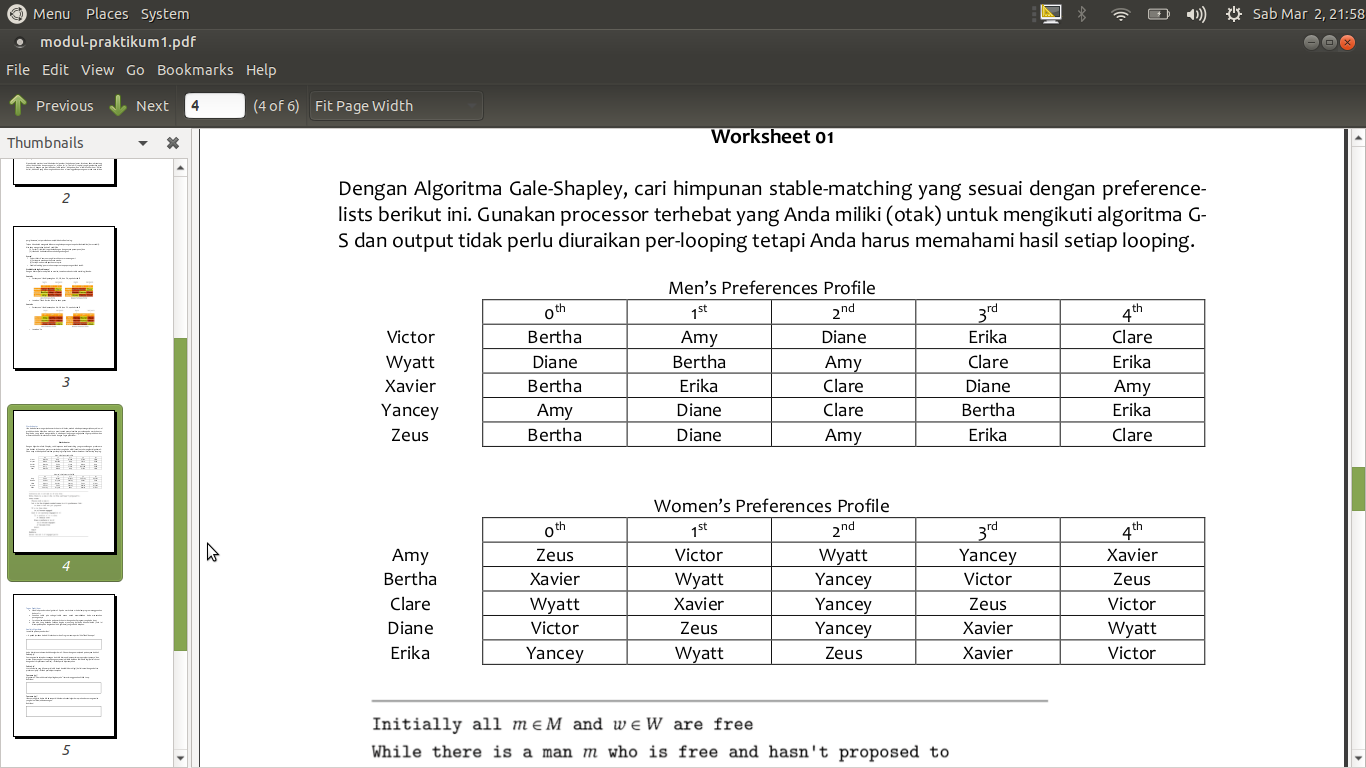
S-1 Teknik Informatika

Fakultas Matematika & Ilmu Pengetahuan Alam

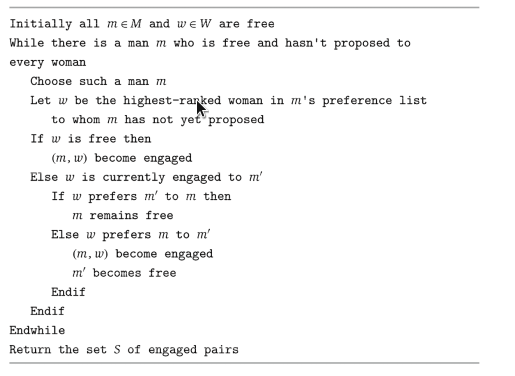
Universitas Padjadjaran

Jalan Raya Bandung - Sumedang Km. 21 Jatinangor 45363

*Stable Matching Problem*



Tabel *Preference*

Ada pula *pseudo code* dari algoritma ini adalah sebagai berikut:

Proses *Matchmaking* secara bertahap:

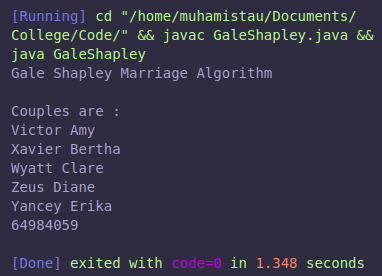
* Victor dengan Bertha
* Wyatt dengan Diane
* Xavier dengan Bertha, karena Bertha lebih memilih Xavier, maka Victor dilepas.
* Yancey dengan Amy
* Zeus dengan Bertha, karena Bertha sudah dengan Xavier dan lebih memilih Xavier. Zeus tidak mendapatkan pasangan
* Victor dengan Amy, karena Amy lebih memilih Victor, Amy melepas Yancey.
* Yancey dengan Diane. Karena Diane lebih memilih yancey, Diane melepas Wyatt
* Zeus dengan Diane. Karena Diane lebih memilih Zeus, Diane melepas Yancey
* Wyatt dengan Clare
* Yancey dengan Erika

Sehingga Pasangan terakhir adalah sebagai berikut:

* Victor Amy
* Xavier Bertha
* Wyatt Clare
* Zeus Diane
* Yancey Erika

Program Java

|  |
| --- |
| /\*\* \* Java Program to Implement Gale Shapley Algorithm \*\*/ /\*\* Class GaleShapley \*\*/ import java.util.concurrent.TimeUnit;  public class GaleShapley {  private int N, engagedCount;  private String[][] menPref;  private String[][] womenPref;  private String[] men;  private String[] women;  private String[] womenPartner;  private boolean[] menEngaged;  /\*\* Constructor \*\*/  public GaleShapley(String[] m, String[] w, String[][] mp, String[][] wp)  {  N = mp.length;  engagedCount = 0;  men = m;  women = w;  menPref = mp;  womenPref = wp;  menEngaged = new boolean[N];  womenPartner = new String[N];  calcMatches();  }  /\*\* function to calculate all matches \*\*/  private void calcMatches()  {  while (engagedCount < N)  {  int free;  for (free = 0; free < N; free++)  if (!menEngaged[free])  break;  for (int i = 0; i < N && !menEngaged[free]; i++)  {  int index = womenIndexOf(menPref[free][i]);  if (womenPartner[index] == null)  {  womenPartner[index] = men[free];  menEngaged[free] = true;  engagedCount++;  }  else  {  String currentPartner = womenPartner[index];  if (morePreference(currentPartner, men[free], index))  {  womenPartner[index] = men[free];  menEngaged[free] = true;  menEngaged[menIndexOf(currentPartner)] = false;  }  }  }   }  printCouples();  }  /\*\* function to check if women prefers new partner over old assigned partner \*\*/  private boolean morePreference(String curPartner, String newPartner, int index)  {  for (int i = 0; i < N; i++)  {  if (womenPref[index][i].equals(newPartner))  return true;  if (womenPref[index][i].equals(curPartner))  return false;  }  return false;  }  /\*\* get men index \*\*/  private int menIndexOf(String str)  {  for (int i = 0; i < N; i++)  if (men[i].equals(str))  return i;  return -1;  }  /\*\* get women index \*\*/  private int womenIndexOf(String str)  {  for (int i = 0; i < N; i++)  if (women[i].equals(str))  return i;  return -1;  }  /\*\* print couples \*\*/  public void printCouples()  {  System.out.println("Couples are : ");  for (int i = 0; i < N; i++)  {  System.out.println(womenPartner[i] +" "+ women[i]);  }  }  /\*\* main function \*\*/  public static void main(String[] args)  {  long startTime = System.nanoTime();   System.out.println("Gale Shapley Marriage Algorithm\n");  String[] m = {"Victor", "Wyatt", "Xavier", "Yancey", "Zeus"};  String[] w = {"Amy", "Bertha", "Clare", "Diane", "Erika"};   String[][] mp = {  {"Bertha", "Amy", "Diane", "Erika", "Clare"},  {"Diane", "Bertha", "Amy", "Clare", "Erika"},  {"Bertha", "Erika", "Clare", "Diane", "Amy"},  {"Amy", "Diane", "Clare", "Bertha", "Erika"},  {"Bertha", "Diane", "Amy", "Erika", "Clare"}  };   String[][] wp = {  {"Zeus", "Victor", "Wyatt", "Yancey", "Xavier"},  {"Xavier", "Wyatt", "Yancey", "Victor", "Zeus"},  {"Wyatt", "Xavier", "Yancey", "Zeus", "Victor"},  {"Victor", "Zeus", "Yancey", "Xavier", "Wyatt"},  {"Yancey", "Wyatt", "Zeus", "Xavier", "Victor"}  };  GaleShapley gs = new GaleShapley(m, w, mp, wp);    long endTime = System.nanoTime();  long durationInNano = (endTime - startTime);    System.out.println(durationInNano);  } } |

Output Program  


1. Apakah jawaban Anda di Worksheet 01 dan Program sama persis? Jika Tidak? Kenapa?

Ya, Jawaban saya sama persis seperti di Worksheet 01

1. Algoritma G-S berakhir setelah paling banyak n2 iterasi menggunakan While Loop. Buktikan!

Di setiap iterasi, satu wanita akan dilamar oleh satu pria sesuai dengan daftar preference pria. Karena ada banyak n pria dan setiap pria memiliki panjang daftar yang juga sebanyak n, maka paling banyak lamaran yang terjadi adalah n2.

1. Jika seorang pria bebas di beberapa titik dalam eksekusi algoritma, maka ada seorang wanita yang belum dia ajak bertunangan. Buktikan!

Jika seorang pria berstatus bebas pada suatu titik dalam eksekusi program, misal dikarenakan oleh dilepasnya pria tersebut oleh wanita yang sebelumnya telah menerima, tetapi wanita tersebut kembali dilamar oleh pria yang memiliki priority lebih tinggi pada list wanita tersebut, maka akan terdapat wanita yang belum dia ajak untuk bertunangan dikarenakan jumlah dari pria dan wanita yaitu sama n. Sehingga dipastikan setiap pria akan memiliki pasangan wanita.

1. Himpunan S yang dikembalikan saat terminasi adalah *perfect matching*. Buktikan!

Himpunan S yang dikembalikan oleh algoritma G-S adalah benar perfect matching dikarenakan berpacu pada setiap list prioritas dari pihak pria maupun wanita

1. Sebuah eksekusi algoritma G-S mengembalikan satu set pasangan S. Set S adalah pasangan yang stabil. Buktikan!

Set Pasangan S yang dikembalikan oleh algoritma G-S adalah benar perfect matching dikarenakan berpacu pada setiap list prioritas dari pihak pria maupun wanita