**LAPORAN PRAKTIKUM 1**

**ANALISIS ALGORITMA**

****

**DISUSUN OLEH**

**TIVANI SHAKILLA ERVI 140810170023**

**KELAS A**

**PROGRAM STUDI S-1 TEKNIK INFORMATIKA**

**UNIVERSITAS PADJADJARAN**

**JATINANGOR**

**2019**

**Worksheet 01**

Dengan Algoritma Gale-Shapley, cari himpunan stable-matching yang sesuai dengan preferencelists berikut ini.

**Men’s Preferences**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1st | 2nd | 3rd | 4th |
| Victor | Bertha | Amy | Diane | Erika | Clare |
| Wyatt | Diane | Bertha | Amy | Clare | Erika |
| Xavier | Bertha | Erika | Clare | Diane | Amy |
| Yancey | Amy | Diane | Clare | Bertha | Erika |
| Zeus | Bertha | Diane | Amy | Erika | Clare |
| **Women’s Preferences** | |  |  |  |  |
|  |  |  |  |  |  |
|  | 0 | 1st | 2nd | 3rd | 4th |
| Amy | Zeus | Victor | Wyatt | Yancey | Xavier |
| Bertha | Xavier | Wyatt | Yancey | Victor | Zeus |
| Clare | Wyatt | Xavier | Yancey | Zeus | Victor |
| Diane | Victor | Zeus | Yancey | Xavier | Wyatt |
| Erika | Yancey | Wyatt | Zeus | Xavier | Victor |

Loop 1

* **1. (Victor, Bertha)**
* **2. (Wyatt, Diane)**
* **3. (Xavier, Bertha)** Maka (Bertha, Xavier > Victor)

Sehingga ~~(Victor, Bertha)~~

* **4. (Yancey, Amy)**
* **5. (Zeus, Bertha)** Maka (Bertha, Xavier > Zeus)

Sehingga ~~(Zeus, Bertha)~~

**(Zeus, Diane)** Menjadi (Diane, Zeus > Wyatt)

Sehingga ~~(Wyatt, Diane)~~

Loop 2

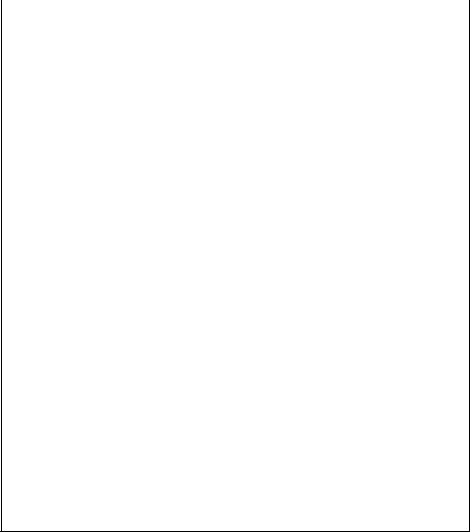
* **1. (Victor, Amy)** Maka (Amy, Victor --> Yancey)Sehingga ~~(Yancey, Amy)~~
* **2. (Wyatt, Bertha)** Maka (Bertha, Xavier --> Bertha)Sehingga ~~(Wyatt, Bertha)~~

**(Wyatt, Amy)** Menjadi (Amy, Victor --> Wyatt)Sehingga ~~(Wyatt, Amy)~~

**(Wyatt, Clare)**

* **3. (Yancey, Diane)** Maka (Diane, Zeus --> Yancey)

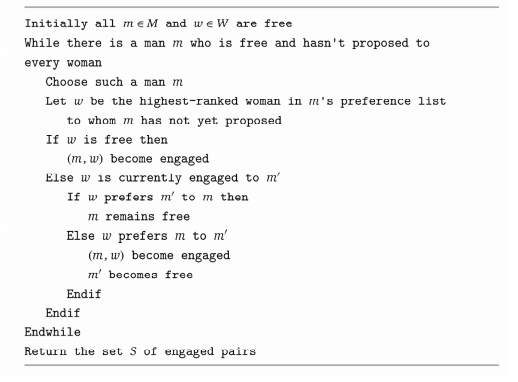
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Sehingga | | ~~(Yancey, Diane)~~ |  |  |  |  |  |
|  |  |  |  | **(Yancey, Clare)** Menjadi (Clare, Wyatt --> Yancey) | | | | | | |  |
|  |  |  |  |  |  | Sehingga | ~~(Yancey, Clare)~~ | |  |  |  |
|  | *(victor,Bertha)* | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  | *(wyatt,Diane)* | | |  |  | **(Yancey, Bertha)** Menjadi (Bertha, Xavier --> | | | | |  |
|  | ***(Xavier,Bertha)*** | | |  |  | Yancey) | | | | |  |
|  | *~~(victor,Bertha)~~* | | *Victor free* | | | Sehingga | | ~~(Yancey, Bertha)~~ | |  |  |
| *(Yancey,Amy)* | | |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  | ***(Zeus,Diane)*** | | |  |  |  |  | **(Yancey, Erika)** | | |  |
|  | *~~(wyatt,Diane)~~* | *Wyatt free* | | | |  |  |  |  |  |  |



* ***(victor,Amy)*** *~~(Yancey,Amy)~~Yancy free*
* ***(Wyatt,Clare)***
* ***(Yancey,Erika)***

**Kesimpulan:**

**(Xavier, Bertha), (Zeus, Diane), (Victor, Amy), (Wyatt, Clare), (Yancey, Erika).**



**Source Code :**

|  |
| --- |
| #include <iostream> |
|  | using namespace std; |
|  |  |
|  | typedef int prefer[5]; |
|  |  |
|  | struct orang{ |
|  | string nama; |
|  | int pasangan; |
|  | int preferences[5]; |
|  | }; |
|  |  |
|  | orang setDataOrang(string nama, prefer prefers); |
|  | void stableMatching(orang pria[5], orang wanita[5]); |
|  |  |
|  | int main(){ |
|  | orang pria[5], wanita[5]; |
|  | prefer prefers; |
|  |  |
|  | //Pria |
|  | prefers[0] = 1; prefers[1] = 0; prefers[2] = 3; prefers[3] = 4; prefers[4] = 2; |
|  | pria[0] = setDataOrang("Victor",prefers); |
|  | prefers[0] = 3; prefers[1] = 1; prefers[2] = 0; prefers[3] = 2; prefers[4] = 4; |
|  | pria[1] = setDataOrang("Wyatt",prefers); |
|  | prefers[0] = 1; prefers[1] = 4; prefers[2] = 2; prefers[3] = 3; prefers[4] = 0; |
|  | pria[2] = setDataOrang("Xavier",prefers); |
|  | prefers[0] = 0; prefers[1] = 3; prefers[2] = 2; prefers[3] = 1; prefers[4] = 4; |
|  | pria[3] = setDataOrang("Yancey",prefers); |
|  | prefers[0] = 1; prefers[1] = 3; prefers[2] = 0; prefers[3] = 4; prefers[4] = 2; |
|  | pria[4] = setDataOrang("Zeus",prefers); |
|  |  |
|  | //Wanita |
|  | prefers[0] = 4; prefers[1] = 0; prefers[2] = 1; prefers[3] = 3; prefers[4] = 2; |
|  | wanita[0] = setDataOrang("Amy",prefers); |
|  | prefers[0] = 2; prefers[1] = 1; prefers[2] = 3; prefers[3] = 0; prefers[4] = 4; |
|  | wanita[1] = setDataOrang("Bertha",prefers); |
|  | prefers[0] = 1; prefers[1] = 2; prefers[2] = 3; prefers[3] = 4; prefers[4] = 0; |
|  | wanita[2] = setDataOrang("Clare",prefers); |
|  | prefers[0] = 0; prefers[1] = 4; prefers[2] = 3; prefers[3] = 2; prefers[4] = 1; |
|  | wanita[3] = setDataOrang("Diane",prefers); |
|  | prefers[0] = 3; prefers[1] = 1; prefers[2] = 4; prefers[3] = 2; prefers[4] = 0; |
|  | wanita[4] = setDataOrang("Erika",prefers); |
|  |  |
|  | stableMatching(pria,wanita); |
|  |  |
|  | cout << "\nHasil Stable Matching :" << endl; |
|  | cout << "-----------------" << endl |
|  | << "| Man \t Woman\t|" << endl |
|  | << "-----------------" << endl; |
|  | for(int i = 0; i < 5; i++){ |
|  | cout << "|" << pria[i].nama << "\t " << wanita[pria[i].pasangan].nama << "\t|" << endl; |
|  | } |
|  | cout << "-----------------" << endl; |
|  | } |
|  |  |
|  | orang setDataOrang(string nama, prefer prefers){ |
|  | orang org; |
|  | org.nama = nama; |
|  | org.pasangan = -1; |
|  | for(int i = 0; i < 5;i++){ |
|  | org.preferences[i] = prefers[i]; |
|  | } |
|  | return org; |
|  | } |
|  |  |
|  | void stableMatching(orang pria[5], orang wanita[5]){ |
|  | bool bebas = true; |
|  | int i=0, j=0, prefSek,prefSuk; |
|  | while(bebas){ |
|  | if(wanita[pria[i].preferences[j]].pasangan = -1){ |
|  | pria[i].pasangan = pria[i].preferences[j]; |
|  | wanita[pria[i].preferences[j]].pasangan = i; |
|  | }else{ |
|  | int k = 0; |
|  | prefSek = -1; prefSuk = -1; |
|  | while(prefSek == -1 || prefSuk == -1){ |
|  | if(i == wanita[pria[i].preferences[j]].preferences[k]){ |
|  | prefSuk = k; |
|  | } |
|  | if(wanita[pria[i].preferences[j]].pasangan == wanita[pria[i].preferences[j]].preferences[k]){ |
|  | prefSek = k; |
|  | } |
|  | k++; |
|  | } |
|  | if(prefSuk<prefSek){ |
|  | pria[wanita[pria[i].preferences[j]].pasangan].pasangan = -1; |
|  | pria[i].pasangan = pria[i].preferences[j]; |
|  | wanita[pria[i].preferences[j]].pasangan = i; |
|  | }else{ |
|  | j++; |
|  | continue; |
|  | } |
|  | } |
|  | bebas = false; |
|  | j=0; |
|  | for(int l = 0; l < 5; l++){ |
|  | if(pria[l].pasangan == -1){ |
|  | i = l; |
|  | bebas = true; |
|  | break; |
|  | } |
|  | } |
|  | } |
|  | } |

**ANALISIS ALGORITMA**

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

**Jawab :**

Iya, jawabannya sama.

Anda diminta untuk membuktikan algoritma G-S benar dengan menjawab pertanyaan berikut:

**Fakta (1.1):**

Seorang wanita tetap bertunangan dari titik di mana dia menerima proposal pertamanya; dan urutan mitra yang bertunangan dengannya menjadi lebih baik dan lebih baik lagi (hal ini sesuai dengan daftar preferensi wanita). -> tidak perlu dipertanyakan

**Fakta (1.2):**

Urutan wanita yang dilamar pria lebih buruk dan lebih buruk lagi (hal ini sesuai dengan daftar preferensi pria). -> tidak perlu dipertanyakan

**Teorema (1.3):**

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

Algoritma ini terus membuat kemajuan. Dalam setiap iterasi pengulangan loop, seorang pria lajang melamar wanita berikutnya dalam daftar pilihannya, seseorang yang belum pernah ia ajukan sebelumnya. Karena ada n laki-laki dan setiap daftar preferensi memiliki panjang n, ada paling banyak proposal n2 yang dapat terjadi. Jadi jumlah iterasi yang dapat terjadi paling banyak adalah n2.

Membuktikan bahwa pencocokan yang dikembalikan stabil.

**Teorema (1.4):**

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

Buktinya berdasarkan kontradiksi. Misalkan ada waktu tertentu dalam pelaksanaan algoritma ketika seorang pria lajang, namun telah mengusulkan kepada setiap wanita. Ini berarti, setiap wanita telah diusulkan setidaknya satu kali. Dengan teori 1, mendapatkan bahwa setiap wanita bertunangan. Jadi, kita telah melibatkan n wanita dan karenanya n laki-laki bertunangan, yang menyiratkan bahwa m juga terlibat bertentangan dengan asumsi bahwa m adalah lajang.

**Teorema (1.5):**

Himpunan S yang dikembalikan saat terminasi adalah perfect matching Buktikan!

Ini dikarenakan semua laki-laki dan perempuan sasling berpasangan

**Teorema (1.6):**

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

Menunjukkan bahwa pencocokan yang dikembalikan adalah pencocokan sempurna. Buktinya dengan kontradiksi. Misalkan tidak, maka ada seorang pria yang masih lajang di akhir algoritma. Menurut teori 2, itu berarti m belum melamar beberapa wanita. Tetapi kemudian, algoritma tidak akan keluar dari pengulangan loop, menghasilkan kontradiksi yang diinginkan.

Menunjukkan bahwa pencocokan yang dikembalikan stabil. Lagi-lagi buktinya dengan kontradiksi. Misalkan ada laki-laki m dan m’ dan wanita w dan w’ sehingga (m, w) dan (m’, w’) berada di S, tetapi m lebih suka w’ ke w dan w’ lebih suka m ke m’. Dengan algoritma, w adalah wanita terakhir yang saya ajukan. Karena m lebih suka w’ ke w, m harus sudah mengusulkan ke w’ sebelum usulannya ke w. Pada saat itu, atau nanti, w’ bertunangan dengan seorang pria, katakanlah m’’, yang ia sukai lebih dari m. Pada akhirnya, w’ bertunangan dengan m’. Oleh teori 1, menemukan bahwa w’ lebih memilih m’ daripada m’’ dan lebih memilih m’’ daripada m; ini menyiratkan bahwa w’ lebih suka m’ daripada m, bertentangan dengan asumsi bahwa w’ lebih memilih m daripada m’.