

Aplikasi Minimum Cost Flow Graph dalam Menyelesaikan Job Assignment Problem

Dewana Gustavus Haraka Otang - 13521173

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

13521173@std.stei.itb.ac.id

Abstrak—Job assignment problem adalah permasalahan dimana kita diberikan n buah pekerja dan n buah pekerjaan dimana setiap pekerja memiliki biaya untuk mengerjakan sebuah pekerjaan. Untuk mencari biaya minimal yang dibutuhkan apabila seorang pekerja hanya dapat mengambil 1 pekerjaan dan sebuah pekerjaan hanya bisa di pilih oleh satu pekerja dapat dicari secara optimal dengan mengubah masalah menjadi permasalahan flow network dan menyelesaikannya dengan algoritma flow dan graf yang lebih optimal.

Kata Kunci—Assignment Problem, Flow Network, Maximum Flow, Shortest Path.

I. PENDAHULUAN

Job assignment problem adalah permasalahan dimana kita diberikan N buah pekerja dan N buah pekerjaan dan Setiap pekerja memiliki biaya untuk mengerjakan sebuah pekerjaan. Kita diminta untuk menentukan biaya minimum yang dibutuhkan apabila seorang pekerja hanya dapat mengambil 1 pekerjaan dan sebuah pekerjaan hanya bisa di pilih oleh satu pekerja.

	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

Worker A takes 8 units of time to finish job 4.

An example job assignment problem. Green values show optimal job assignment that is A-Job2, B-Job1 C-Job3 and D-Job4

Gambar 1.1 Contoh Job Assignment Problem

Sumber : geeksforgeeks.org

Penyelesaian masalah tersebut dapat diselesaikan dengan melakukan bruteforce dengan mencoba seluruh kemungkinan pekerja dipasangkan dengan sebuah pekerjaan yang akan dikerjakan. Metode bruteforce akan memiliki kompleksitas waktu $O(n!)$ dengan melakukan permutasi dari array yang

berisi elemen 1 sampai N dengan nilai array menyatakan pekerjaan dan index menyatakan pekerja yang mengambil pekerjaan pada index tersebut. Terdapat solusi yang lebih efisien dengan cara mengubah permasalahan Job Assignment menjadi permasalahan Minimum Cost Flow yang dapat diselesaikan dengan kompleksitas waktu $O(n^3)$.

II. TEORI DASAR

A. Graf

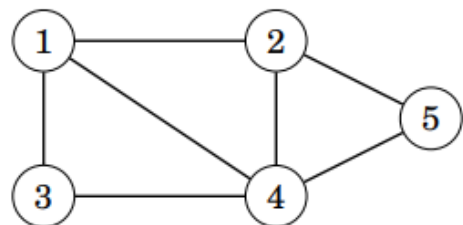
Graf adalah struktur data yang digunakan untuk merepresentasikan objek-objek diskrit beserta hubungan antara objek-objek tersebut. Graf umumnya terdiri atas 2 komponen yaitu :

1. Simpul/Node/Vertex
Simpul biasanya direpresentasikan menggunakan titik yang menyatakan sebuah objek diskrit.
2. Sisi/Edge
Sisi biasanya direpresentasikan menggunakan garis yang menyatakan hubungan antara 2 buah objek.

B. Jenis-Jenis Graf

Terdapat banyak jenis graf, beberapa jenis graf yang dipakai pada makalah ini adalah :

1. Graf tak Berarah
Graf tak berarah adalah graf dimana edge merepresentasikan hubungan antara 2 buah node dan tidak memperhatikan arah, sehingga apabila ada edge dari A ke B maka juga ada edge dari B ke A.

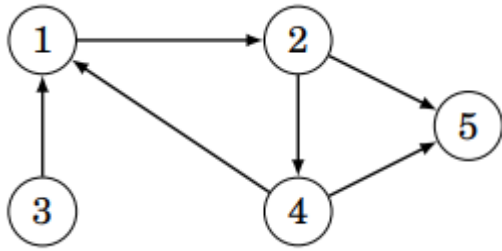


Gambar 2.1 Contoh Graf tak Berarah

Sumber : cses.fi/book/book.pdf

2. Graf Berarah

Graf berarah adalah graf dimana edge merepresentasikan hubungan antara 2 buah node dan memperhatikan arah, sehingga apabila ada edge dari A ke B belum tentu ada edge dari B ke A.

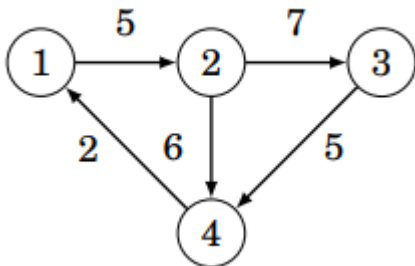


Gambar 2.2 Contoh Graf Berarah

Sumber : cses.fi/book/book.pdf

3. Graf Berbobot

Graf berbobot adalah graf yang edgenya memiliki sebuah nilai yang memiliki arti, contoh: bobot menunjukkan jarak antara 2 buah node. Graf berbobot dapat memiliki arah atau tidak memiliki arah.



Gambar 2.3 Contoh Graf Berbobot

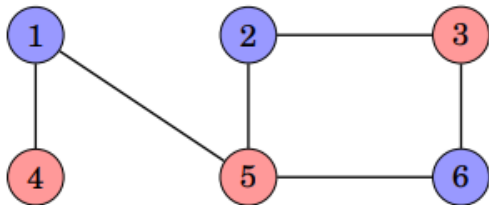
Sumber : cses.fi/book/book.pdf

4. Graf tak Berbobot

Graf tak berbobot adalah graf yang seluruh edgenya hanya memiliki bobot 1.

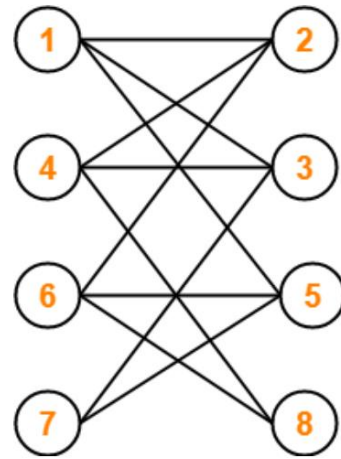
5. Graf Bipartit

Sebuah graf dapat dikatakan sebagai graf bipartite apabila ada cara untuk mewarnai node pada graf dengan 2 buah warna dan tidak ada 2 buah node yang bertetangga memiliki warna yang sama.



Gambar 2.4 Contoh Graf Bipartit

Sumber : cses.fi/book/book.pdf

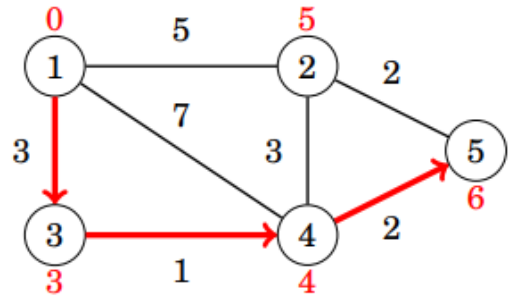


Gambar 2.5 Contoh Graf Bipartit

Sumber : slide materi kuliah

C. Shortest Path

Shortest path merupakan sebuah permasalahan pada graf berbobot dan berarah dimana kita diminta untuk mencari jarak terpendek antara 2 buah node.



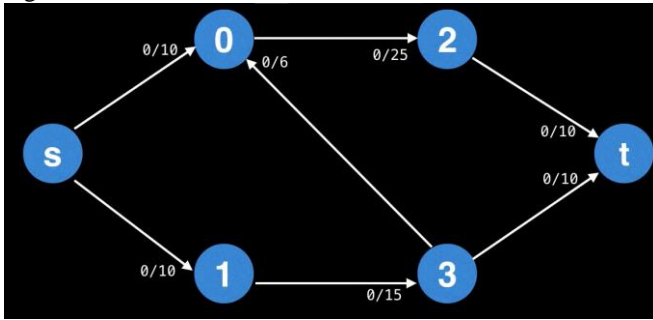
Gambar 2.6 Contoh Permasalahan Shortest Path, nilai berwarna merah pada node menunjukkan jarak minimum dari node 1, panah berwarna merah adalah contoh jalan terpendek dari node 1 ke node 5

Sumber : cses.fi/book/book.pdf

Terdapat banyak algoritma untuk mencari shortest path seperti Floyd-Warshall algorithm, Dijkstra algorithm, pada makalah ini penulis akan memakai Bellman-Ford algorithm khususnya variasi Bellman-Ford algorithm yaitu Shortest Path Faster Algorithm (SPFA). Algoritma SPFA dapat digunakan untuk mencari jarak terpendek dimulai dari sebuah node ke seluruh node lain (biasa disebut sebagai Single Source Shortest Path (SSSP)). Cara kerja algoritma SPFA adalah dengan menyiapkan sebuah antrian node yang akan diproses untuk dicari jarak terpendek yang baru dan akan dilakukan berulang-ulang sampai antrian kosong. Kompleksitas waktu algoritma SPFA adalah $O(nm)$ dimana n menyatakan banyak node, dan m menyatakan banyak edge.

D. Flow Network

Flow network adalah variasi dari graf berbobot dimana nilai pada edge merepresentasikan nilai kapasitas aliran yang dapat mengalir.

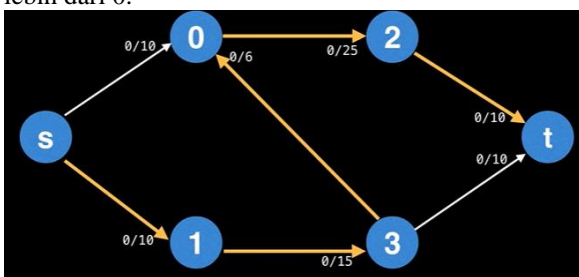


Gambar 2.7 Contoh Flow Network

Sumber : Youtube William Fiset

Beberapa istilah pada flow network yang sering dipakai pada flow network :

1. Kapasitas
Kapasitas adalah nilai pada edge yang menyatakan berapa banyak aliran yang dapat alirkan pada sebuah edge.
2. Source dan Sink
Source adalah node awal pada flow network yang akan memberikan aliran. Sink adalah node akhir pada flow network yang akan menerima aliran.
3. Graf Residu
Graf residu adalah graf pada flow network yang sudah di modifikasi dengan kapasitas pada edgenya sudah dikurangi dengan aliran yang sudah mengalir sehingga memudahkan dalam perhitungan kapasitas yang tersisa.
4. Augmenting Path
Augmenting path adalah sebuah jalur pada graf residu yang dimulai dari source dan berakhir pada sink dengan memakai edge yang memiliki sisa kapasitas lebih dari 0.

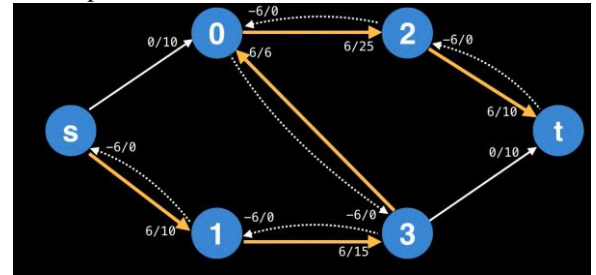


Gambar 2.8 Contoh Flow Network dengan Augmenting Path

Sumber : Youtube William Fiset

5. Bottleneck Value
Bottleneck value adalah nilai edge terkecil pada augmenting path pada contoh augmenting path diatas bottleneck valuenya adalah 6.
6. Edge residu
Edge residu adalah edge dengan arah terbalik yang dipasang setelah mengisi aliran pada augmenting path. Edge residu berfungsi untuk meng-undo aliran yang

tidak optimal.



Gambar 2.9 Contoh Edge Residu pada Flow Network, edge residu direpresentasikan dengan garis putus-putus

Sumber : Youtube William Fiset

III. IMPLEMENTASI SOLUSI JOB ASSIGNMENT PROBLEM

A. Solusi Bruteforce

Salah satu cara untuk menyelesaikan permasalahan job assignment adalah dengan mencoba seluruh kemungkinan cara seorang pekerja memilih pekerjaan dan mengambil nilai minimalnya.

```
#include <bits/stdc++.h>
using namespace std;

const int INF = INT32_MAX; // inialisasi nilai infinity
const int n = 4; // jumlah pekerja dan pekerjaan
// matrix biaya, matrix[i][j] menyatakan pekerja i membutuhkan biaya matrix[i][j] untuk melakukan pekerjaan j
int matrix[n][n] = {{9,2,7,8},
                  {6,4,3,7},
                  {5,8,1,0},
                  {7,6,9,4}};

int main(){
    int mincost = INF; // inialisasi nilai biaya dengan nilai infinity
    int pickjob[n]; // pickjob[i] menyatakan pekerjaan apa yang dipilih pekerja ke-i
    for(int i=0;i<n;i++){
        pickjob[i] = i; // inialisasi nilai permutasi 0 sampai n-1
    }

    do{ // iterasi seluruh kemungkinan pemilihan 0(n!)
        int cost = 0;
        for(int i=0;i<n;i++){
            cost += matrix[i][pickjob[i]]; // hitung biaya yang diperlukan
        }
        mincost = min(mincost, cost); // update nilai biaya minimal
    }while(next_permutation(pickjob.begin(), pickjob.end()));

    printf("biaya minimal yang diperlukan adalah : %d\n", mincost);

    return 0;
}
```

Gambar 3.1 Implementasi Solusi Bruteforce dalam C++

Sumber : Dokumentasi Pribadi

Solusi bruteforce memiliki kompleksitas waktu $O(n!)$ dikarenakan mencoba seluruh permutasi sehingga membutuhkan waktu yang sangat banyak sehingga kita perlu mencari solusi yang lebih optimal tanpa mencoba seluruh kemungkinan.

B. Solusi Minimum Cost Flow

Solusi lain untuk menyelesaikan permasalahan job assignment adalah dengan mengubah masalah menjadi permasalahan graf yang nantinya dapat diselesaikan dengan algoritma Minimum Cost Maximum Flow (MCMF).

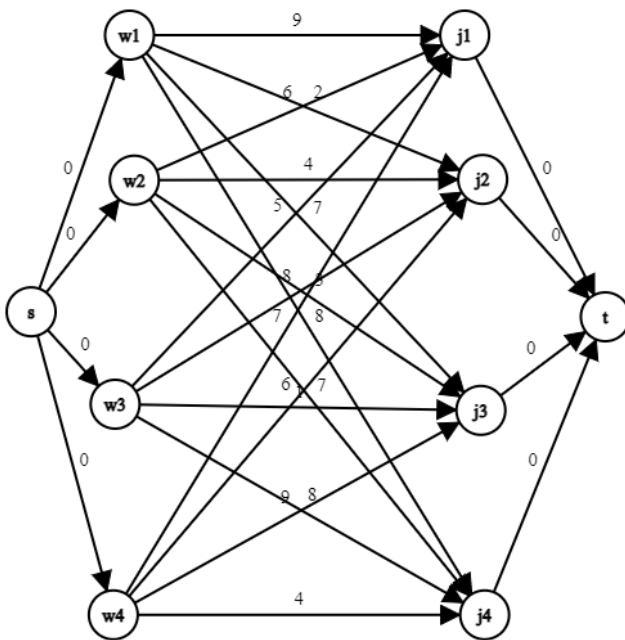
	Job 1	Job 2	Job 3	Job 4
A	9	2	7	8
B	6	4	3	7
C	5	8	1	8
D	7	6	9	4

Gambar 3.2 Contoh Matrix Permasalahan Job Assignment

Sumber : geeksforgeeks.org

Graf flow network permasalahan job assignment dapat dibuat dengan langkah-langkah seperti berikut :

1. Buat graf bipartit dengan node seperti contoh pada gambar 2.5 dengan node di kiri adalah node pekerja dan node di kanan adalah node pekerjaan.
2. Berikan edge dari node kiri ke seluruh node kanan dengan kapasitas flow 1 dan biaya sesuai dengan matrix biaya seorang pekerja melakukan sebuah pekerjaan.
3. Buat node source di sebelah kiri node pekerja dan berikan edge dari source ke seluruh node pekerja dengan kapasitas flow 1 dan biaya 0.
4. Buat node sink di sebelah kanan node pekerjaan dan berikan edge dari seluruh node pekerjaan ke sink dengan kapasitas flow 1 dan biaya 0.



Gambar 3.3 Flow Network untuk Matrix Contoh

Sumber : Dokumentasi Pribadi

Setelah flow network dibuat kita dalam menjalankan algoritma Minimum Cost Maximum Flow untuk mencari total biaya minimum.

```

#include <bits/stdc++.h>
using namespace std;

const int INF = INT32_MAX; // inisialisasi nilai infinity
const int n = 4; // jumlah pekerja dan pekerjaan
const int m = n * 2 + 2; // nilai maximum node, 0..n-1 pekerja, n..2n-1 pekerjaan, 2n source, 2n+1 sink
// matrix biaya, matrix[i][j] menyatakan pekerja i membutuhkan biaya matrix[i][j] untuk melakukan pekerjaan j
int matrix[n][n] = {{0,7,8},
                  {6,4,3,7},
                  {5,8,1,8},
                  {7,6,9,4}};

int flow[m][m] = {}; // nilai flow bernilai -1,0,1
int previous[m]; // vertex sebelum vertex[i] setelah memasang flow
vector<int> dist(m); // jarak dari source ke sebuah vertex
bool in_queue[m]; // mencatat apakah vertex i sudah ada dalam queue atau tidak
queue<int> q; // queue untuk node yang akan diproses dalam flow

```

Gambar 3.3 Implementasi Solusi Minimum Cost Flow dalam C++ bagian 1, Inisialisasi Variabel

Sumber : Dokumentasi Pribadi

Sebelum menjalankan algoritma ada beberapa variabel yang perlu di inisialisasi seperti variabel yang menampung jarak, antrian node untuk di proses, flow, dan matrix biaya

```

void reset(){
    // me-reset ulang seluruh variabel previous, dist, dan queue
    for(int i=0;i<m;i++){
        previous[i] = 0;
        dist[i] = INF;
        in_queue[i] = false;
    }
    dist[source] = 0; // jarak dari source adalah 0
    previous[source] = -1; // tidak ada vertex sebelum source
    q.push(source); // masukkan source kedalam queue untuk di proses
}

void pushqueue(int curr, int before, int new_distance){
    // prosedur memasukkan sebuah vertex kedalam queue dan memperbarui nilai previous dan dist
    dist[curr] = new_distance;
    previous[curr] = before;
    if (!in_queue[curr]) {
        q.push(curr);
        in_queue[curr] = true;
    }
}

int popqueue(){
    // memberikan nilai terdepan pada queue dan menghapusnya untuk di proses
    int value = q.front();
    q.pop();
    in_queue[value] = false;
    return value;
}

```

Gambar 3.4 Implementasi Solusi Minimum Cost Flow dalam C++ bagian 2, Implementasi Fungsi dan Prosedur

Sumber : Dokumentasi Pribadi

Ada juga beberapa fungsi dan prosedur yang perlu di implementasikan seperti fungsi me-reset state, memasukkan node ke dalam queue, mengeluarkan node dari queue.

```

while(!q.empty()) // lakukan pencarian shortest path flow untuk seluruh pekerja
    reset(); // me-reset seluruh state
while (is_sink()) // lakukan pencarian shortest path sampai tidak ada vertex yang perlu di proses
    if (vertex == source) // operasi yang diproses adalah source
        for(int worker=pekerjaan[source]; // iterasi seluruh pekerja
            IF (flow[worker][pekerjaan[source]] < 1) // apakah belum ada flow, tentukan jumlah queue
                pushqueue(worker, source, 0); // jarak dari source ke pekerja adalah 0
        }
    }
    else if (vertex < n) // operasi yang diproses adalah pekerja
        for(int job=pekerjaan[vertex]; // iterasi seluruh pekerjaan
            IF (flow[vertex][job] < 1 && dist[job] > dist[vertex] + matrix[vertex][job-1]) // apakah belum ada flow dan mempunyai jarak yang lebih kecil
                pushqueue(job, vertex, dist[vertex] + matrix[vertex][job-1]); // tambahkan path baru dari pekerja ke pekerjaan
        }
    }
    else if (vertex > n) // operasi seluruh pekerja (path dari pekerjaan ke pekerja adalah path yang sudah dibuat)
        for(int worker=pekerjaan[vertex]; // iterasi seluruh pekerja (path dari pekerjaan ke pekerja adalah path yang sudah dibuat)
            IF (flow[vertex][worker] < 1 && dist[worker] > dist[vertex] - matrix[worker][vertex-1]) // apakah ada path yang sudah dibuat dan jaraknya lebih kecil
                pushqueue(worker, vertex, dist[vertex] - matrix[worker][vertex-1]); // tambahkan path baru dari pekerjaan ke pekerja
        }
    }
}

```

Gambar 3.5 Implementasi Solusi Minimum Cost Flow dalam C++ bagian 3, Algoritma Shortest Path

Sumber : Dokumentasi Pribadi

Setelah seluruh variabel, fungsi dan prosedur di siapkan Langkah pertama dalam algoritma adalah mencari augmenting path dengan algoritma SPFA.

```
int curcost = 0; // mencari nilai terkecil
for(int job=0; job<n; job++) // iterasi seluruh pekerjaan
    if (flow[job][sink] == 0 && dist[job] < curcost) // apabila belum ada flow ke sink dan biayanya lebih kecil
        curcost = dist[job]; // ubah biaya terkecil
    previous[sink] = job; // siapkan jalan dari pekerjaan yang biayanya terkecil ke sink
if (curcost == INF) break; // apabila nilai terkecilnya masih infinity maka seluruh pekerja sudah memilih pekerjaan dan pencarian akan
```

Gambar 3.6 Implementasi Solusi Minimum Cost Flow dalam C++ bagian 4, Menentukan Path dengan Biaya Terkecil

Sumber : Dokumentasi Pribadi

Setelah itu kita perlu mengecek path mana yang memiliki biaya terkecil dan apakah kita menemukan path baru, jika tidak menemukan path baru maka algoritma selesai.

```
cost += curcost; // tambah total biaya dengan biaya baru
int cur = sink; // lakukan jalan mundur dari sink ke source
while(cur != -1) { // mundur sampai source
    int prev = previous[cur]; // mundur
    if (prev != -1) { // apabila bukan di source
        flow[prev][cur] = 1; // berikan flow
        flow[cur][prev] = -1; // siapkan flow dengan arah terbalik
    }
    cur = prev; // mundur
}
}
```

Gambar 3.7 Implementasi Solusi Minimum Cost Flow dalam C++ bagian 5, Membuat Graf Residu

Sumber : Dokumentasi Pribadi

Setelah kita mencari path dengan biaya terkecil kita akan memperbarui graf residu dan edge residu dengan memberikan aliran dan menyiapkan edge residu dengan arah terbalik untuk meng-undo aliran yang tidak optimal.

```
int pick[n]; // jawaban permasalahan dengan pick[i] menyatakan pekerjaan apa yang dipilih pekerja ke
for(int i=0; i<n; i++) // iterasi seluruh pekerja
    for(int j=0; j<n; j++) // iterasi seluruh pekerjaan
        if (flow[i][j] == 1) // apabila ada flow dari pekerja ke pekerjaan
            pick[i] = j; // pekerja memilih pekerjaan tersebut

// bagian output
printf("matrix job assignment\n");
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        printf("%d ", matrix[i][j]);
    }
    printf("\n");
}
printf("biaya total yang paling minimum adalah : %d\n", cost);
printf("dengan cara pemilihan adalah :\n");
for(int i=0; i<n; i++){
    printf("pekerja ke-%d memilih pekerjaan %d dengan biaya %d\n", i+1, pick[i], matrix[i][pick[i]]);
}
return 0;
```

Gambar 3.8 Implementasi Solusi Minimum Cost Flow dalam C++ bagian 6, Output

Sumber : Dokumentasi Pribadi

Setelah seluruh langkah dijalankan kita akan mendapatkan total biaya yang paling minimum beserta cara pemilihan pekerjaan. Langkah-langkah yang dilakukan diatas dapat dipastikan akan memberikan solusi dengan total biaya yang minimum karena algoritma akan terus mengulang pencarian shortest path sampai tidak lagi menemukan augmenting path yang lebih kecil. Algoritma yang digunakan juga memastikan setiap pekerja hanya memilih 1 pekerjaan dan sebuah pekerjaan hanya akan dipilih oleh 1 pekerja karena kita perlu memaksimalkan flow terlebih dahulu dan flow yang maksimal adalah sebanyak n (jumlah pekerja dan pekerjaan) dengan source memberikan flow sebesar 1 ke n buah pekerja dan n buah pekerjaan akan memberikan flow sebesar 1 ke sink.

```
matrix job assignment
9 2 7 8
6 4 3 7
5 8 1 8
7 6 9 4
biaya total yang paling minimum adalah : 13
dengan cara pemilihan adalah :
pekerja ke-1 memilih pekerjaan 2 dengan biaya 2
pekerja ke-2 memilih pekerjaan 1 dengan biaya 6
pekerja ke-3 memilih pekerjaan 3 dengan biaya 1
pekerja ke-4 memilih pekerjaan 4 dengan biaya 4
```

Gambar 3.9 Hasil Program pada Matrix Contoh

Sumber : Dokumentasi Pribadi

```
matrix job assignment
19 41 45 47 36 17 20 12 44
29 49 47 13 40 40 36 12 47
22 17 33 27 43 18 45 44 45
23 12 29 31 25 40 10 50 50
13 31 32 15 28 29 13 25 10
37 40 47 39 30 43 26 10 40
35 41 49 49 30 34 30 50 23
49 13 34 25 28 25 36 27 19
22 22 28 32 13 50 34 44 19
biaya total yang paling minimum adalah : 145
dengan cara pemilihan adalah :
pekerja ke-1 memilih pekerjaan 6 dengan biaya 17
pekerja ke-2 memilih pekerjaan 4 dengan biaya 13
pekerja ke-3 memilih pekerjaan 3 dengan biaya 33
pekerja ke-4 memilih pekerjaan 7 dengan biaya 10
pekerja ke-5 memilih pekerjaan 1 dengan biaya 13
pekerja ke-6 memilih pekerjaan 8 dengan biaya 10
pekerja ke-7 memilih pekerjaan 9 dengan biaya 23
pekerja ke-8 memilih pekerjaan 2 dengan biaya 13
pekerja ke-9 memilih pekerjaan 5 dengan biaya 13
```

Gambar 3.10 Hasil Program pada Matrix Acak 9x9

Sumber : Dokumentasi Pribadi

IV. KESIMPULAN

Solusi sebuah permasalahan yang optimal bisa saja bukan memakai pendekatan yang terlihat dengan jelas. Seperti pada permasalahan job assignment solusi yang terlihat jelas adalah solusi bruteforce yang memiliki kompleksitas waktu yang sangat buruk. Terdapat solusi permasalahan job assignment yang optimal tetapi tidak terlihat dengan jelas yaitu dengan mengubah permasalahan menjadi masalah graf dan dapat diselesaikan dengan algoritma graf yang sudah ditemukan algoritma yang optimal.

V. UCAPAN TERIMA KASIH

Pertama-tama, puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena berkat rahmat-Nya penulis dapat menyelesaikan makalah ini dengan baik. Penulis juga berterimakasih kepada orang tua, serta teman-teman yang selalu memberikan semangat dan dukungan kepada penulis sehingga makalah ini dapat terselesaikan. Penulis juga tak lupa berterimakasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T., Ibu Fariska Zakhralativa Ruskanda, S.T.,M.T., selaku dosen mata kuliah matematika diskrit yang telah memberikan banyak ilmu dan motivasi dalam kegiatan perkuliahan. Terakhir, penulis memohon maaf apabila dalam penulisan makalah ini terdapat

kesalahan baik disengaja maupun tidak disengaja. Penulis berharap makalah ini dapat bermanfaat bagi banyak orang.

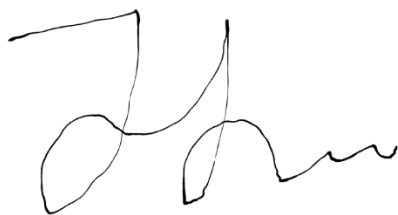
REFERENSI

- [1] Laaksonen, Antti. Guide to Competitive Programming. Switzerland: Springer, <https://cses.fi/book/book.pdf>, diakses tanggal 5 Desember 2022.
- [2] <https://www.geeksforgeeks.org/job-assignment-problem-using-branch-and-bound/>, diakses tanggal 5 Desember 2022.
- [3] <https://cp-algorithms.com/graph/Assignment-problem-min-flow.html>, diakses tanggal 6 Desember 2022.
- [4] https://cp-algorithms.com/graph/min_cost_flow.html, diakses tanggal 7 Desember 2022.
- [5] https://cp-algorithms.com/graph/bellman_ford.html, diakses tanggal 8 Desember 2022.
- [6] <https://www.youtube.com/watch?v=LdOnanfc5TM&list=PLDV1Zeh2NRsDj3NzHbbFIC58etjZhiGcG&index=2>, diakses tanggal 8 Desember 2022.
- [7] <https://www.topcoder.com/thrive/articles/Assignment%20Problem%20and%20Hungarian%20Algorithm>, diakses tanggal 10 Desember 2022.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2022



Dewana Gustavus Haraka Otang 13521173