

## BAB 7

### GRAPH (ALGORITMA SINGLEPATH)

#### 7.1. TUJUAN PEMBELAJARAN

Tujuan dari pembelajaran pada pertemuan *graph*, diantaranya mahasiswa diharapkan mampu:

1. Mengerti algoritma pencarian jalur terpendek untuk *SinglePath*
2. Mampu mengimplementasikan algoritma Dijkstra

#### 7.2. DASAR TEORI

##### 7.2.1. Algoritma Dijkstra

Algoritma Dijkstra ditemukan oleh Edsger Dijkstra yang merupakan algoritma dengan prinsip greedy yang memecahkan masalah lintasan terpendek untuk sebuah graf berarah dengan bobot sisi yang tidak negatif. Sederhananya Algoritma Dijkstra merupakan prosedur untuk menemukan jalur terpendek antara node / tepi grafik yang biasanya dikaitkan dengan optimasi pemecahan masalah. Perlu diketahui graf terpendek dari sebuah pohon dibuat dengan memulai dari simpul sumber ke semua titik lain dalam graf tersebut.

Algoritma Dijkstra bersifat sederhana dan lempang (straightforward). Konsep algoritma greedy ini menerapkan prinsip “hanya memikirkan solusi terbaik yang akan diambil pada setiap langkah tanpa memikirkan konsekuensi ke depan”.

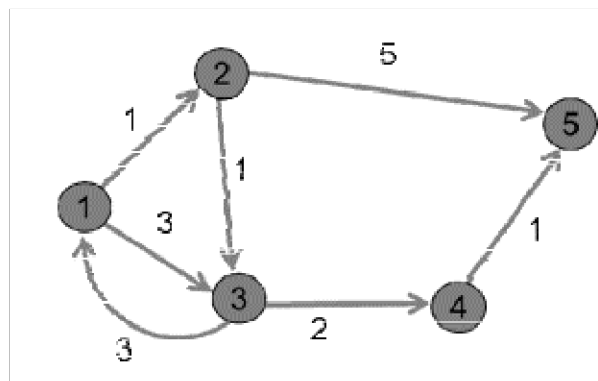
Di dalam algoritma ini terdapat input yang berupa sebuah graph berarah yang berbobot (weighted directed graph)  $G$  dan sebuah sumber vertex  $s$  dalam  $G$  dan  $V$  adalah himpunan semua vertices dalam graph  $G$ . Vertices  $(u, v)$  melambangkan hubungan dari vertex  $u$  ke vertex  $v$ .

Umumnya pada setiap sisi dari graph merupakan pasangan vertices. Himpunan semua tepi disebut  $E$ . Bobot (weights) dari semua sisi dihitung dengan fungsi :  $w: E \rightarrow [0, \infty)$ , jadi  $w(u,v)$  adalah jarak tak- negatif dari vertex  $u$  ke vertex  $v$ . Ongkos (cost) dari sebuah sisi dapat dianggap sebagai jarak antara dua vertex, yaitu jumlah jarak semua sisi dalam jalur tersebut. Untuk sepasang vertex  $s$  dan  $t$  dalam  $V$ , algoritma ini menghitung jarak terpendek dari  $s$  ke  $t$ .

Dalam pencarian jalur terpendeknya algoritma dijkstra bekerja dengan mencari bobot yang paling minimal dari suatu graf berbobot, jarak terpendek akan diperoleh dari dua atau

lebih titik dari suatu graf dan nilai total yang didapat adalah yang bernilai paling kecil. Misalkan  $G$  adalah graf berarah berlabel dengan titik-titik  $V(G) = \{v_1, v_2, \dots, v_n\}$  dan path terpendek yang dicari adalah dari  $v_1$  ke  $v_n$ . Algoritma Dijkstra dimulai dari titik  $v_1$ . Dalam iterasinya, algoritma akan mencari satu titik yang jumlah bobotnya dari titik 1 terkecil. Titik-titik yang terpilih dipisahkan, dan titik-titik tersebut tidak diperhatikan lagi dalam iterasi berikutnya. Langkah-langkah yang digunakan untuk menyelesaikan algoritma Dijkstra, diantaranya :

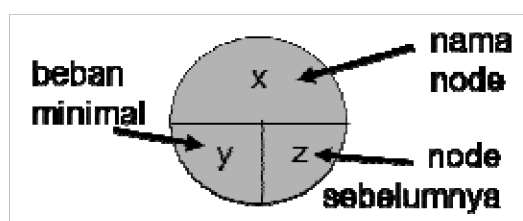
1. Pertama, menentukan titik asal dan titik tujuan sebelum proses
2. Kedua, akumulasikan jarak minimal, lalu simpan ke titik berikutnya.
3. Ketiga, lakukanlah dari titik asal sampai titik tujuan



Gambar 7.1 Contoh *graph*

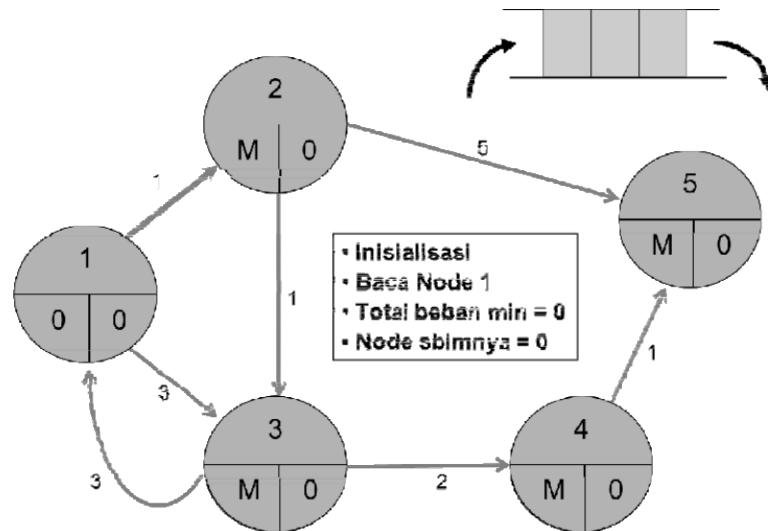
Sebagai ilustrasi, graph pada Gambar 7.1 di atas langkah-langkah nya sebagai berikut :

1. Pada graph di atas, ditentukan titik asal = 1 dan titik tujuan = 5. Selanjutnya, menentukan setiap node mempunyai nilai  $x$  (sebagai nama node), nilai  $y$  (sebagai beban minimal) dan nilai  $z$  (sebagai node sebelumnya). Nilai  $y$  dan  $z$  disimpan sebagai vektor seperti Gambar 7.2.



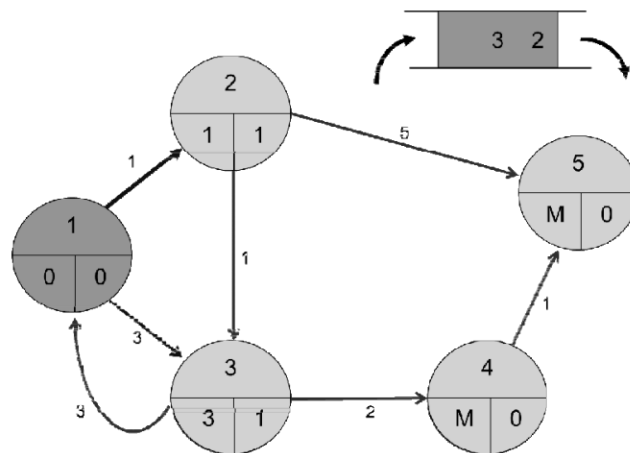
Gambar 7.2 Nilai  $x$ ,  $y$  dan  $z$

2. Siapkan tumpukan untuk menyimpan node yang akan diproses. Sebagai inisialisasi awal, nilai  $y$  bernilai  $M$  (*big integer*) kecuali node awal bernilai 0, sedangkan nilai  $z$  bernilai 0. Node titik awal disimpan dalam tumpukan, dan baca nilai tersebut.



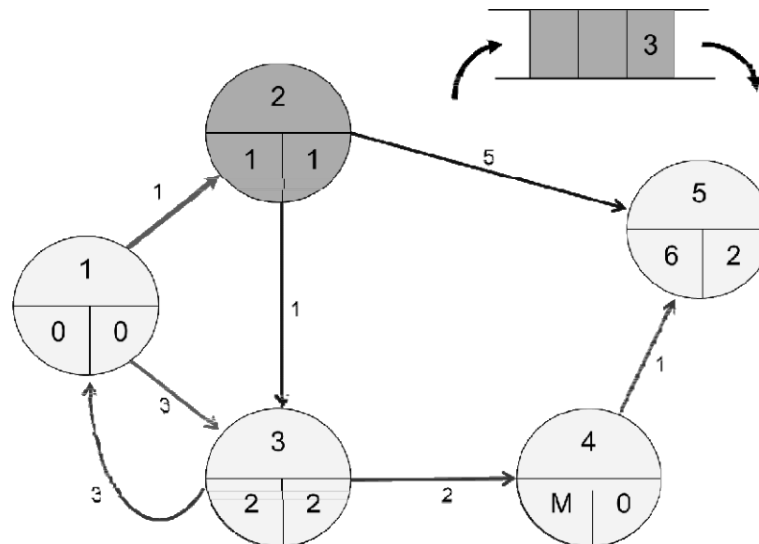
Gambar 7.3 Inisialisai awal

3. Baca node 1, lalu lakukan perubahan nilai y dan z menjadi nilai beban dan node sebelumnya yang terkecil. Masukkan node z pada tumpukan jika node tersebut tidak ada di tumpukan, bukan titik awal dan bukan titik akhir.



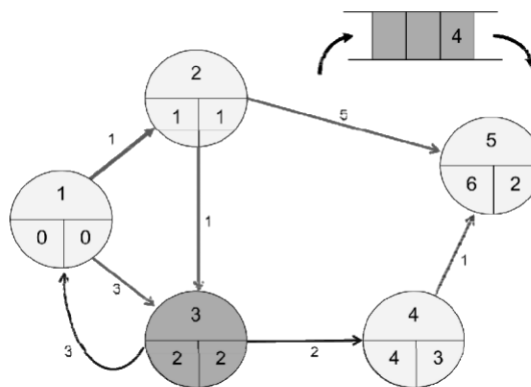
Gambar 7.4 Perhatikan node 1

4. Enqueue node pada tumpukan, baca node 2 dan lakukan hal yang sama dengan langkah 3.



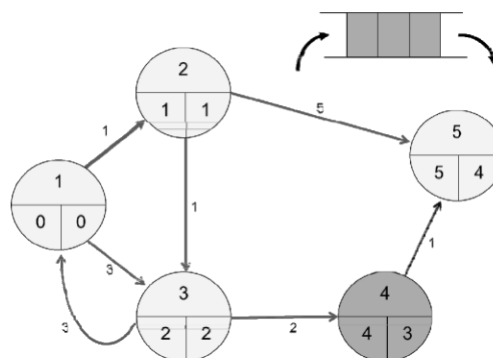
Gambar 7.5 Perhatikan node 2

5. Enqueue node pada tumpukan, baca node 3 dan lakukan hal yang sama dengan langkah 3.



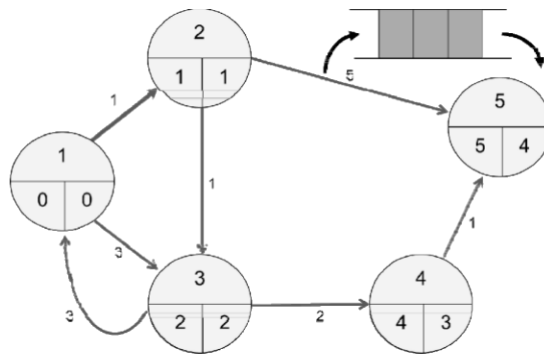
Gambar 7.6 Perhatikan node 3

6. Enqueue node pada tumpukan, baca node 4 dan lakukan hal yang sama dengan langkah 3.



Gambar 7.7 Perhatikan node 4

7. Proses dinyatakan selesai apabila terdapat tumpukan kosong, sehingga menghasilkan vektor bebandan rute minimal.



Gambar 7.8 Tumpukan kosong, proses selesai.

### 7.3. PERCOBAAN

1. Buatlah workspace menggunakan Replit.
2. Buatlah project baru GRAPH2 yang berisi file C++ source untuk algoritma dijkstra
3. Cobalah untuk masing-masing percobaan di bawah

#### Percobaan 1 : Implementasi graph dengan queue

```
#include <iostream>
#include <vector>
#include <queue>

#define M 1000

using namespace std;

const int N = 5;

void dijkstra(int graph[N][N], int source, int
destination) {
    vector<pair<int, int>> adj[N];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (graph[i][j] != M) {
                adj[i].push_back(make_pair(j, graph[i][j]));
            }
        }
    }
}
```

```

int Q[N], R[N];
    for (int i = 0; i < N; i++) {
        Q[i] = M;
        R[i] = -1;
    }
    priority_queue<pair<int, int>, vector<pair<int,
int>>, greater<pair<int, int>>> pq;
    pq.push(make_pair(0, source - 1));
    Q[source - 1] = 0;
    while (!pq.empty()) {
        int u = pq.top().second;
        pq.pop();

        for (int i = 0; i < adj[u].size(); i++) {
            int v = adj[u][i].first;
            int w = adj[u][i].second;

            if (Q[v] > Q[u] + w) {
                Q[v] = Q[u] + w;
                R[v] = u;
                pq.push(make_pair(Q[v], v));
            }
        }
    }

    cout << "Beban = ";
    for (int i = 0; i < N; i++) {
        cout << Q[i] << " ";
    }
    cout << endl;
    cout << "Rute = ";
    for (int i = 0; i < N; i++) {
        cout << R[i] << " ";
    }
    cout << endl;

```

```

    }
int main() {
    int graph[N][N] = {{0, 1, 3, M, M},
                        {M, 0, 1, M, 5},
                        {3, M, 0, 2, M},
                        {M, M, M, 0, 1},
                        {M, M, M, M, 0}};

    int source, destination;
    cout << "Masukkan node asal : ";
    cin >> source;
    cout << "Masukkan node tujuan : ";
    cin >> destination;

    dijkstra(graph, source, destination);

    return 0;
}

```

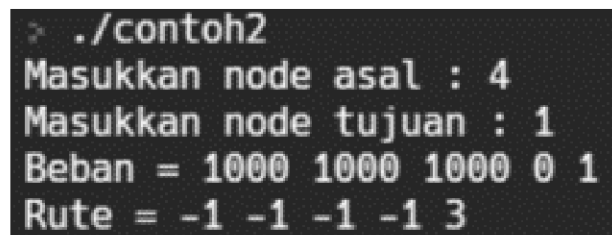
## OUTPUT

Masukkan node asal : 4

Masukkan node tujuan : 1

Beban = 1000 1000 1000 0 1

Rute = -1 -1 -1 -1 3



```

> ./contoh2
Masukkan node asal : 4
Masukkan node tujuan : 1
Beban = 1000 1000 1000 0 1
Rute = -1 -1 -1 -1 3

```

## Percobaan 2 : Mencari rute terpendek pada graph

```

#include <bits/stdc++.h>

#define MAX 100005

#define INF INT_MAX

```

```
using namespace std;

vector<pair<int, int> > adj[MAX];
bool vis[MAX];
int dist[MAX];

void dijkstra(int start)
{
    memset(vis, false, sizeof vis);
    for (int i = 0; i < MAX; i++)
        dist[i] = INF;
    dist[start] = 0;

    priority_queue<pair<int, int>, vector<pair<int,
int> >, greater<pair<int, int> > > pq;
    pq.push({0, start});

    while (!pq.empty())
    {
        pair<int, int> p = pq.top();
        pq.pop();

        int x = p.second;
        if (vis[x])
            continue;
        vis[x] = true;

        for (int i = 0; i < adj[x].size(); i++)
        {
            int e = adj[x][i].first;
            int w = adj[x][i].second;
            if (dist[x] + w < dist[e])
            {
```



```

        dist[e] = dist[x] + w;
        pq.push({dist[e], e});
    }
}

}

}

int main()
{
    adj[1].push_back({2, 3});
    adj[2].push_back({1, 3});
    adj[2].push_back({3, 2});
    adj[3].push_back({2, 2});
    adj[3].push_back({4, 4});
    adj[3].push_back({5, 5});
    adj[4].push_back({3, 4});
    adj[5].push_back({3, 5});

    dijkstra(1);

    cout << "Jarak terpendek dari node 1 ke node 5
adalah " << dist[5] << endl;

    return 0;
}

```

## OUTPUT

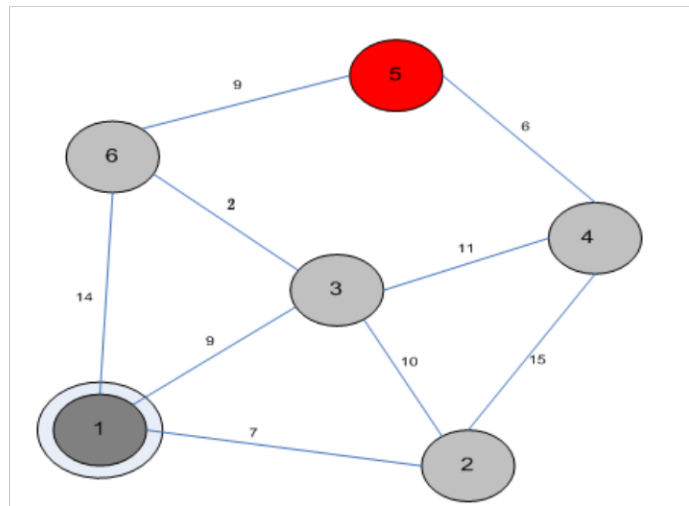
Jarak terpendek dari node 1 ke node 5 adalah 10

```

% ./main
Jarak terpendek dari node 1 ke node 5 adalah 10
%

```

#### 7.4. TUGAS DAN LATIHAN



Gambar 7.9 Soal nomor 1

Dari gambar di atas carilah rute terpendek, gambarkan seperti contoh yang ada di dasar teori, serta buatlah program C++ nya di replit.