TUGAS

JARINGAN KOMPUTER

Nama : Wahyuni Oktarina

Nim : 09011181419027

Kelas : SK5A

JURUSAN SISTEM KOMPUTER

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SRIWIJAYA

Bellman Ford Algorithm

Ford Algoritma Bellman merupakan kelanjutan dari posting Shortest Path Menggunakan Algoritma Dijkstra. Sambil belajar tentang cara Dijkstra, kita belajar bahwa itu benar-benar efisien algoritma untuk menemukan sumber tunggal jalur terpendek dalam grafik yang disediakan tidak memiliki tepi berat negatif dan tidak ada siklus berat negatif.

Penjelasan - Shortest Path Menggunakan Bellman Ford Algoritma dapat menggunakan Bellman Ford untuk diarahkan serta un-diarahkan grafik. Mari kita memecahkan masalah dengan menggunakan grafik diarahkan di sini. Ide dari algoritma ini adalah cukup sederhana.

- Ia memelihara daftar simpul yang belum dikunjungi
- Ia memilih vertex (sumber) dan memberikan biaya maksimum yang mungkin (yaitu infinity) untuk setiap vertex lainnya.
- Biaya sumber tetap nol karena benar-benar membutuhkan apa-apa untuk mencapai dari sumber titik untuk dirinya sendiri.
- Dalam setiap iterasi berikutnya dari algoritma mencoba untuk bersantai setiap tepi dalam grafik (dengan meminimalkan biaya titik di mana tepi insiden).
- Ini mengulangi langkah 4 untuk | V | -1 kali. Dengan iterasi terakhir kita akan mendapatkan beberapa jalur terpendek dari Sumber ke setiap sudut.

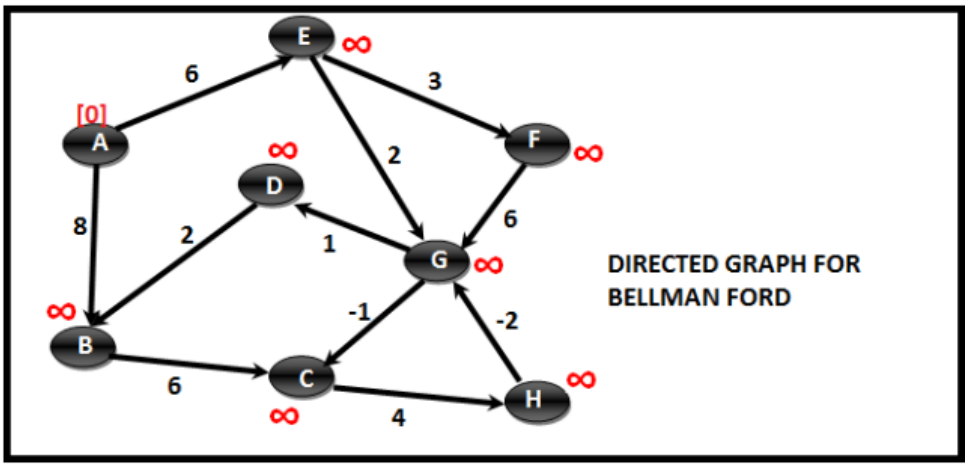
Argumen akan, bahwa jalan terpendek dalam grafik dengan | V | simpul tidak bisa lebih panjang daripada | V | - 1. Jadi, jika kita bersantai semua tepi | V | - 1 kali, kita akan membahas semua kemungkinan santai tepi dan kami akan pergi dengan semua jalur terpendek. Ada banyak bukti oleh Induksi tersedia dalam kasus Anda

lebih tertarik. Atau Anda dapat berharap untuk posting berikutnya tentang "kebenaran Ford Algoritma Bellman".

Demonstrasi

Untuk tujuan demonstrasi, kami akan mempertimbangkan grafik berikut. Langkah1:

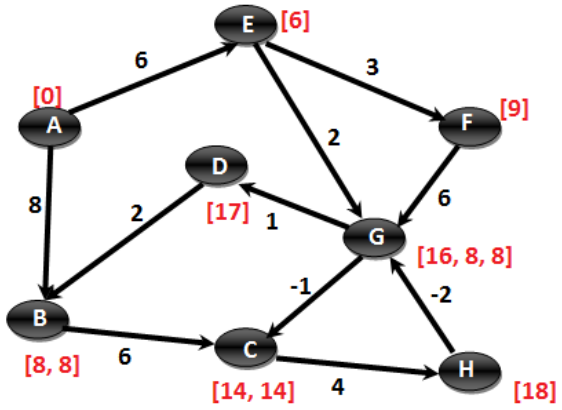
Mengingat A sebagai sumber, menetapkan nol biaya. Tambahkan semua simpul (A, B, C, D, E, F, G, H) untuk daftar. Untuk semua simpul kecuali A menetapkan infinity biaya. Juga, disarankan untuk mempertahankan daftar tepi berguna. Berikut adalah grafik untuk memulai dengan :



Langkah2:

Ambil satu titik pada satu waktu mengatakan A, dan bersantai semua tepi dalam grafik. Titik layak memperhatikan adalah bahwa kita hanya dapat bersantai tepi yang outgoing dari titik A. Sisa tepi tidak akan membuat banyak perbedaan. Jadi, berikut ini adalah sub langkah untuk langkah2

Relax (A, E). Biaya E = MIN (biaya saat E [∞], biaya A [0] + W {A, E} [6]) Biaya (E) menjadi 6.
Relax (A, B). Biaya B = MIN (biaya saat B [∞], biaya A [0] + W {A, B} [8]) Biaya (B) menjadi 8.
Relax (B, C):. Biaya C = MIN (biaya saat C [∞], biaya B [8] + W {B, C} [6]) Biaya (C) menjadi 14.
Relax (C, H): biaya H = MIN (biaya saat H [∞], biaya C [14] + W {C, H} [4]) Biaya (H) menjadi 18..
Relax (H, G): Biaya G = MIN (biaya saat G [∞], biaya H [18] + W {H, G} [-2]) Biaya (G) menjadi 16..
Relax (G, C):. Biaya C = MIN (biaya saat C [14], biaya G [16] + W {G, C} [-1]) Biaya (C) tetap 14.
Relax (G, D):. Biaya D = MIN (biaya saat D [∞], biaya G [16] + W {G, D} [1]) Biaya (D) menjadi 17.
Relax (D, B): biaya B = MIN (biaya saat B [8], biaya D [17] + W {D, B} [2]) Biaya (B) tetap 8..
Relax (E, F):. Biaya F = MIN (biaya saat F [∞], biaya E [6] + W {E, F} [3]) Biaya (F) menjadi 9.
Relax (E, G). Biaya G = MIN (biaya saat G [16], biaya E [6] + W {E, G} [2]) Biaya (G) menjadi 8
Relax (F, G). Biaya G = MIN (biaya saat G [8], biaya F [9] + W {F, G} [6]) Biaya (G) tetap 8.



Untuk menjangkau setiap vertex dapat diperbarui K kali (di mana K adalah jumlah sisi yang masuk ke vertex ini). Ini mungkin bahwa biaya pertama begitu kurang bahwa itu tidak diubah oleh operasi berikutnya. Ada K angka dalam kurung persegi, satu nomor ditambahkan setiap kali tepi masuk santai.

Langkah3:

Mulai dari salah satu simpul, mengatakan A lagi dan bersantai semua tepi seperti di bawah ini:

Relax (A, E). Biaya E = MIN (biaya saat E [6], biaya A [0] + W {A, E} [6]) Biaya (E) tetap 6.

Relax (A, B). Biaya B = MIN (biaya saat B [8], biaya A [0] + W {A, B} [8]) Biaya (B) tetap 8.

Relax (B, C):. Biaya C = MIN (biaya saat C [14], biaya B [8] + W {B, C} [6]) Biaya (C) tetap 14.

Relax (C, H):. Biaya H = MIN (biaya saat H [18], biaya C [14] + W {C, H} [4]) Biaya (H) tetap 18.

Relax (H, G). Biaya G = MIN (biaya saat G [8], biaya H [18] + W {H, G} [-2]) Biaya (G) tetap 8.

Relax (G, C):. Biaya C = MIN (biaya saat C [14], biaya G [8] + W {G, C} [-1]) Biaya (C) menjadi 7.

Relax (G, D):. Biaya D = MIN (biaya saat D [17], biaya G [8] + W {G, D}

[1]) Biaya (D) menjadi 9.

Relax (D, B). Biaya B = MIN (biaya saat B [8], biaya D [9] + W {D, B} [2])

Biaya (B) tetap 8.

Relax (E, F):. Biaya F = MIN (biaya saat F [9], biaya E [6] + W {E, F} [3])

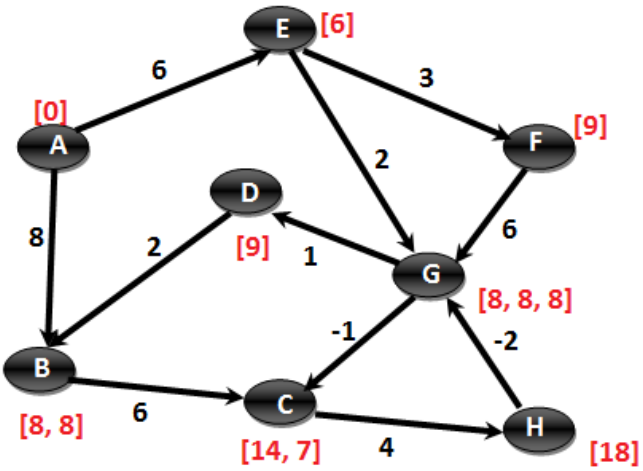
Biaya (F) tetap 9.

Relax (E, G). Biaya G = MIN (biaya saat G [8], biaya E [6] + W {E, G} [2])

Biaya (G) tetap 8.

Relax (F, G). Biaya G = MIN (biaya saat G [8], biaya F [9] + W {F, G} [6])

Biaya (G) tetap 8.



Langkah4:

Mulai dari salah satu simpul, mengatakan A lagi dan bersantai semua tepi seperti di bawah ini:

Relax (A, E). Biaya E = MIN (biaya saat E [6], biaya A [0] + W {A, E} [6])

Biaya (E) tetap 6.

Relax (A, B). Biaya B = MIN (biaya saat B [8], biaya A [0] + W {A, B} [8])

Biaya (B) tetap 8.

Relax (B, C):. Biaya C = MIN (biaya saat C [7], biaya B [8] + W {B, C} [6]) Biaya (C) menjadi 7.

Relax (C, H):. Biaya H = MIN (biaya saat H [18], biaya C [7] + W {C, H} [4]) Biaya (H) menjadi 11.

Relax (H, G). Biaya G = MIN (biaya saat G [8], biaya H [11] + W {H, G} [-2]) Biaya (G) tetap 8.

Relax (G, C): biaya C = MIN (biaya saat C [7], biaya G [8] + W {G, C} [- 1]) Biaya (C) tetap 7..

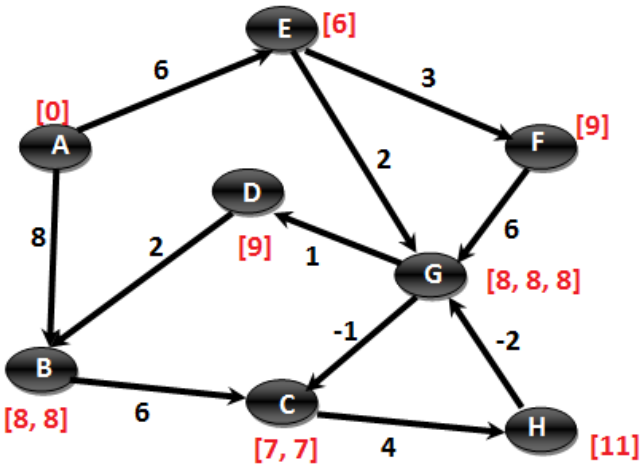
Relax (G, D):. Biaya D = MIN (biaya saat D [9], biaya G [8] + W {G, D} [1]) Biaya (D) tetap 9.

Relax (D, B). Biaya B = MIN (biaya saat B [8], biaya D [9] + W {D, B} [2]) Biaya (B) tetap 8.

Relax (E, F):. Biaya F = MIN (biaya saat F [9], biaya E [6] + W {E, F} [3]) Biaya (F) tetap 9.

Relax (E, G). Biaya G = MIN (biaya saat G [8], biaya E [6] + W {E, G} [2]) Biaya (G) tetap 8.

Relax (F, G). Biaya G = MIN (biaya saat G [8], biaya F [9] + W {F, G} [6]) Biaya (G) tetap 8.



Mungkin tidak melihat adanya perubahan dalam biaya untuk setiap titik dalam iterasi kemudian. Akan lebih baik untuk berhenti di titik itu untuk membuat algoritma yang efisien.

Kita mungkin tanpa henti melihat perubahan biaya dari satu atau titik lain di kemudian iterasi. Ini akan menjadi kasus siklus berat negatif. Oleh karena itu, kita perlu berhenti di beberapa titik dan itu akan menjadi setelah | V | - 1 untuk alasan yang disebutkan di bagian atas.

Setelah kita keluar dari iterasi, kita perlu melakukan ikuti langkah-langkah sekali lagi untuk menemukan jika biaya masih mendapatkan berkurang. Jika kita menemukan pengurangan biaya, kita dapat mengatakan bahwa pasti ada siklus berat negatif dan karenanya tidak ada jalan terpendek ada.

Source Code - Shortest Path Menggunakan Bellman Ford Algoritma

```
public boolean shortestPath(Vertex source, List<Edge> edges, int vertexCount) {

        source.min = 0;


        for (int i = 0; i < vertexCount; i++) {

                for (Edge e : edges) {

                        int edgeWeight = e.weight;

                        int sourceMin = e.start.min;

                        int currentMin = e.end.min;

                        int tempDistance = edgeWeight + sourceMin;

                        if (tempDistance < currentMin) {
```

```
                    e.end.min = tempDistance;
                    e.end.previous = e.start;
                }
            }
        }

        boolean negativeCycle = false;
        for (Edge e : edges) {
            int edgeWeight = e.weight;
            int sourceMin = e.start.min;
            int currentMin = e.end.min;
            int tempDistance = edgeWeight + sourceMin;
            if (tempDistance < currentMin) {
                negativeCycle = true;
                break;
            }
        }

        return negativeCycle;
    }
```

Analisis Algoritma :

Inisialisasi loop berjalan | V | waktu.

Untuk loop berjalan | V | - 1 kali.

Loop batin berjalan | E | kali untuk setiap iterasi dari loop luar.

Loop terakhir berjalan untuk | E | waktu.

Langkah 1 membutuhkan O (| V |) waktu,

Langkah 2 membutuhkan O (| V |. | E |) kali dan

langkah 3 membutuhkan O (| E |) kali. Maka waktu berjalan akan didominasi dengan istilah O (| V |. | E |).