Pencarian *Shortest Path* pada *Directed Acyclic Graph*

Unggul Satrio Respationo

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
e-mail: if16062@students.if.itb.ac.id

ABSTRAK

Graf adalah masalah yang klasik dalam dunia informatika. Jenis dari graf ini pun bermacam – macam, beserta dengan persoalannya masing – masing. Salah satu persoalan yang sering dibahas adalah pencarian *shortest path* atau jalur terpendek. Dari berbagai graf dengan persoalan pencarian jalur terpendek ini, makalah ini akan membahas salah satu dari graf tersebut yaitu *DAG (Directed Acyclic Graph)* atau Graf Berarah Asiklik.

Makalah ini akan membahas beberapa algoritma dalam menyelesaikan pencarian jalur terpendek pada DAG. Yaitu dengan menggunakan algoritma *DFS (Depth-First Search) dan Dynamic Programming* atau Pemrograman Dinamis.

Kata kunci: *DAG (Directed Acyclic Graph)*, Graf Berarah Asiklik, *Shortest Path.*

1. PENDAHULUAN

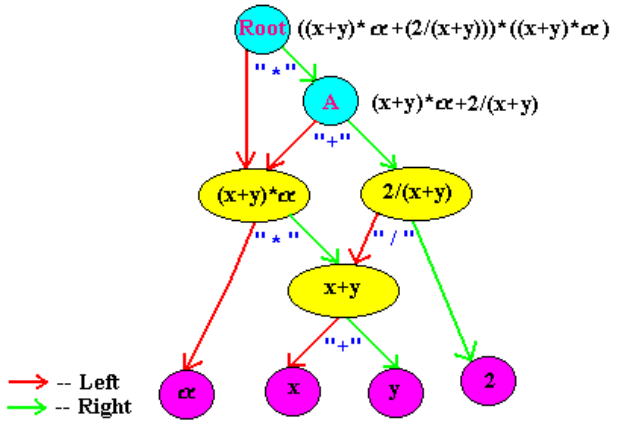
Algoritma adalah cara penyelesaian masalah secara terstruktur yang terbagi dalam langkah per langkah. Dan sering kali algoritma yang ada memiliki skema iteratif di dalamnya. Bahkan tanpa kita sadari dalam kehidupan sehari – hari sudah banyak sekali kita menerapkan algoritma. Pencarian jalur terpendek adalah salah satunya.

Pencarian jalur terpendek sudah menjadi masalah umum dalam berbagai kehidupan. Kenapa tidak, tentunya semua orang ingin untuk mendapatkan hal sebaik mungkin. Dan hak itu bisa kita asosiasikan sebagai suatu bentuk graf. Dimana yang akan dibahas saat ini adalah graf berarah asiklik atau biasa disebut DAG.

Dalam makalah ini akan dibahas dua algoritma yang akan digunakan untuk menyelesaikan masalah tersebut yaitu DFS dan pemrograman dinamis.

2. *DAG (Directed Acyclic Graph)*

DAG adalah sebuah graf berarah tanpa adanya loop di dalamnya. DAG adalah struktur data yang sangat penting karena graf ini bisa dibilang setengah pohon. Penggunaan dari graf ini sangat beragam. Dengan menggunakan DAG banyak persoalan graf menjadi lebih sederhana, salah satunya adalah evaluasi pohon ekspresi (*expression tree evaluation*).



Gambar 1. Aplikasi DAG dalam pohon ekspresi

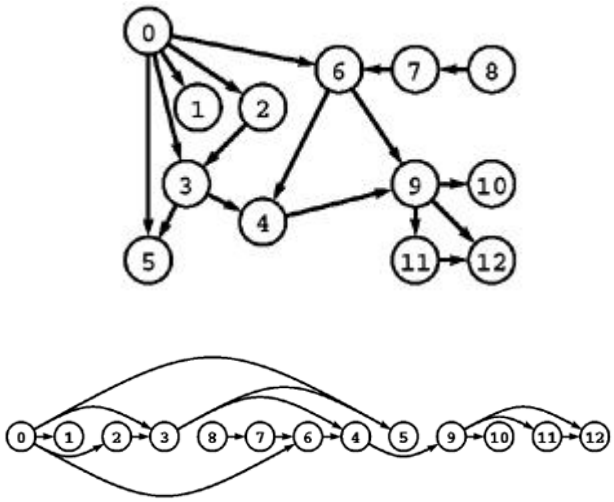
Properti dari DAG yang membuat berbagai persoalan menjadi lebih sederhana adalah kemampuannya untuk diurut secara topologi (*topologically sorted*) menggunakan algoritma DFS. Dua komponen dasar dari DAG adalah :

Akar **:** simpul tanpa sisi masuk
Daun **:** simpul tanpa sisi keluar

Contoh lain dari penggunaan DAG adalah diagram hubungan antar kelas, pohon prasyarat (*prerequisite tree*), pemodelan aktivitas jaringan internet, dan pohon ekspresi.

DAG dalam makalah ini akan membahas tentang algoritma pencarian jalur terpendek di dalamnya. Dalam DAG hampir semua algoritma akan menggunakan sifat khusus dari graf ini yaitu topological sort atau linierization. Untuk itu akan dibahas lebih dahulu topological sort di dalam DAG.

Topological sort adalah pengurutan simpul pada DAG sedemikian sehingga jika terdapat simpul jalur dari simpul *u* ke *v*, maka *v* muncul setelah *u* dalam pengurutan. Sebuah graf siklik (cyclic graf) tidak akan mempunyai urutan topologikal. Topological sort dari suatu graf bisa dipandang sebagai pengurutan simpul – simpulnya dalam satu garis lurus, sehingga semua sisi berarah akan menuju



Gambar 2. DAG dan bentuk liniernya

satu arah. Bisa dibilang topological sort ini menggunakan algoritma DFS dalam penyelesaiannya.Disini kita hanya akan membahas mengenai topologikal sort ini secara singkat, sehingga hanya akan diberikan algoritma yang melakukan pengurutan ini yang dibagia 2 yaitu pengurutan secara terbalik dan pengurutan biasa. Algoritma berikut akandiberikan dalam bahasa Java.

```
class DagTS
{ private Graph G;
  private int cnt, tcnt;
  private int[] pre, post, postI;
  private void tsR(int v)
    {
      pre[v] = cnt++;
      AdjList A = G.getAdjList(v);
      for (int t = A.beg(); !A.end(); t =
A.nxt())
        if (pre[t] == -1) tsR(t);
      post[v] = tcnt; postI[tcnt++] = v;
    }
  DagTS(Graph G)
    { this.G = G; cnt = 0; tcnt = 0;
      pre = new int[G.V()];
      post = new int[G.V()]; postI = new
int[G.V()];
      for (int t = 0; t < G.V(); t++)
        { pre[t] = -1; post[t] = -1;
postI[t] = -1; }
      for (int t = 0; t < G.V(); t++)
        if (pre[t] == -1) tsR(t);
     }
  int order(int v) { return postI[v]; }
  int relabel(int v) { return post[v]; }
}
```

Diatas adalah pengurutan secara terbalik (*reversed topological sort*). Berikut adalah cara yang lain dengan menggantikan prosedur tsR di atas dengan di bawah ini.

```
void tsR(int v)
  {
    pre[v] = cnt++;
    for (int w = 0; w < D.V(); w++)
      if (D.edge(w, v))
        if (pre[w] == -1) tsR(w);
    post[v] = tcnt; postI[tcnt++] = v;
   }
```

Sebetulnya ada banyak sekali cara untuk melakukan topological sort untuk DAG, tetapi tidak akan dibahas lebih lanjut di sini.

3. *SHORTEST PATH* DALAM DAG

Dalam pencarian jalur terpendek tentunya banyak variasi yang dapat digunakan. Dalam makalah ini akan dibahas dua algoritma untuk pencarian jalur terpendek dalam DAG yaitu algoritma DFS dan pemrograman dinamik.

3.1. Algoritma DFS dalam DAG

Dengan menggunakan algoritma DFS bisa didapatkan kompleksitas algoritma yang linier dalam pencarian jalur terpendek. Dengan kunci untuk mendapatkan langkah yang efisien adalah :

Setiap jalur pada DAG, vertex muncul dalam urutan meninggi yang linier.

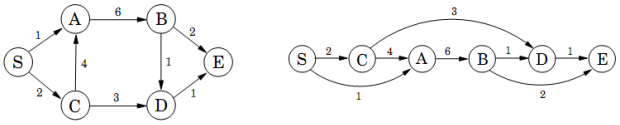
Maka cukup untuk melinierkan (yang dalam kata lain topological sort), sebuah DAG dengan DFS, dan mengunjungi vertex dalam urutan yang terurut. Algoritma dapat dilihat di bawah ini.

```
procedure dag-shortest-paths(G, l, s)
Input:  Dag G = (V,E);
        edge lengths {le : e ∈ E};
        vertex s ∈ V
Output: For all vertices u reachable from
        s, dist(u) is set to the distance
        from s to u.

for all u ∈ V :
   dist(u) = ∞
   prev(u) = nil

dist(s) = 0
Linearize G

for each u ∈ V , in linearized order:
   for all edges (u, v) ∈ E:
      update(u, v)
```

Dapat dilihat bahwa dalam skema di atas maka jika simpul bernilai negatif tidak akan ada masalah. Cara ini juga bisa digunakan untuk mencari jalur terpanjang (*longest path*) dengan cara mengalikan semua simpul dengan (-1).

3.2. Algoritma Pemrograman Dinamis dalam DAG

Seperi sudah dibahas sebelumnya, pencarian jalur terpendek dalam DAG bisa dibilang cukup sederhana. Keistimewaan dari DAG adalah setiap simpulnya dapat dilinierkan, dengan kata lain simpul – simpul dalam graf ini bisa disusun dalam bentuk satu garis.



Gambar 3. Contoh lain DAG dan bentuk liniernya

Untuk melihat kenapa hal ini dapat membantu dalam pencarian jalur terpendek, anggap bahwa kita mau menghitung jarak dari simpul S ke simpul lainnya. Jika kita melihat simpul D maka jalur satu – satunya untuk menuju D adalah melalui *predecessor*-nya yaitu C dan B. Maka untuk mencari jalur terpendek ke D kita hanya perlu membandingkan dua jalur tadi :

$$\text{dist(D)} = \min\{\text{dist(B)} + 1, \text{dist(C)} + 3\} \qquad (1)$$

Hubungan yang sama dapat kita lakukan kepada simpul-simpul lainnya. Jika kita menghitung dist dalam urutan kiri ke kanan dalam gambar 2, sudah pasti bahwa saat kita berada pada simpul v, maka kita sudah mempunyai informasi yang dibutuhkan untuk menghitung dist(v). Maka kita bisa untuk menghitung semua jalur dalam sekal jalan seperti yang dijelaskan dalam algoritma berikut.

```
initialize all dist(·) values to ∞
dist(s) = 0
for each v ∈ V \{s}, in linearized order:
   dist(v) = min(u,v)∈E{dist(u) + l(u, v)}
```

Algoritma ini bisa digunakan dalam menyelesaikan koleksi dari *subproblem*, {dist(u): u ∈ V}. Ini adalah teknik yang sangat general. Setiap node kita menghitung sejumlah fungsi yang nilainya bergantung pada pendahulunya (predecessor). Dalam algoritma kita dikunakan fungsi parsial nilai minimum dari penjumlahan. Tetapi kita juga bisa menjadikannya maksimum jika kita ingin mendapatkan jalur terpanjang. Atau kita juga bisa menggantikannya penjumlahan menjadi perkalian.

4. KESIMPULAN

Algoritma untuk mencari jalur terpendek banyak sekali variasinya. Dalam makalah ini DFS dan pemrograman dinamis salah satunya. DFS bisa dibilang salah satu algoritma yang cukup mudah diterapkan dalam berbagai hal, terutama yang berhubungan dengan graf, seperti pada makalah ini. Adapun pemrograman dinamis sebagai algoritma yang sangat *powerful* yang dimana setiap masalah diselesaikan dengan mengidentifikasi bagian dari upa-masalah dan menyelesaikannya satu – persatu, dimulai dari yang paling kecil.

Tentunya masih banyak lagi algoritma yang bisa digunakan dalam penerapan DAG ini. Penulis berharap makalah ini dapat memberi wawasan untuk terus mengeksplorasi algoritma lain yang bisa digunakan untuk pencarian shortest path, bahkan mungkin menemukan yang lebih mangkus dari yang sudah ada.

REFERENSI

[1] Dasgupta, S. 2006. *Algorithms.* U.C. San Diego: San Diego.

[2] Munir, Rinaldi. 2006. *Diktat Kuliah IF2251 Strategi Algoritmik.* Penerbit ITB: Bandung.

[3] Wilf, Herbert S. 2006. *Algorithm and Complexity.* University of Pennsylvania: Philadelphia.