PENERAPAN ALGORITMA DIJKSTRA PADA PENENTUAN JALUR AKSES TERCEPAT DI BASIS DATA TERSEBAR

Arif Prasetiya-NIM 13508090

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha No.10 Bandung
e-mail: arifwng08@students.itb.ac.id

ABSTRAK

Terdapat banyak algoritma pencarian jalur terpendek, namun yang sering digunakan adalah algoritma Dijkstra. Makalah ini akan mengulas tentang salah satu penerapan algoritma Dijkstra, yaitu penentuan jalur akses tercepat di basis data tersebar khususnya untuk mencari waktu pengambilan data ataupun mengiriman data yang paling cepat. Sistem basis data tersebar ini direpresentasikan dengan sebuah graf. Penerapan algoritma Dijkstra adalah penentuan jalur yang memberikan waktu minimun dari satu site ke site tertentu pada basis data tersebar.

Kata kunci: Teori Graf, Basis Data Tersebar, Algoritma Dijkstra, Jalur akses tercepat.

1. PENDAHULUAN

Teori Graf merupakan teori yang sudah tua usianya namun memiliki banyak terapan sampai saat ini. L.Euler adalah matematikawan asal Swiss yang pertama kali berhasil memodelkan permasalahan jembatan Konissberg menggunakan teori graf. Banyak aplikasi yang berkaitan dengan graf. Di dalam aplikasi tersebut, graf digunakan sebagai alat untuk merepresentasikan atau memodelkan persoalan yang kemudian baru diselesaikan.

Salah satu aplikasi yang menggunakan teori graf adalah persoalan pencarian lintasan terpendek. Graf yang digunakan dalam pencarian lintasan terpendek adalah graf berbobot (*weight graph*). Sampai saat ini sudah banyak algoritma untuk menyelesaikan pencarian lintasan terpendek yang pernah ditulis orang. Algoritma lintasan terpendek yang terkenal adalah algoritma Dijkstra. Algoritma ini menggunakan prinsip Greedy. Prinsip Greedy pada algoritma Dijkstra menyatakan bahwa pada setiap langkah kita memilih sisi yang berbobot minimum dan memasukkannya ke dalam himpunan solusi.

Salah satu contoh persoalan yang dapat direpresentasikan dengan graf adalah basis data tersebar (*distributed database*).

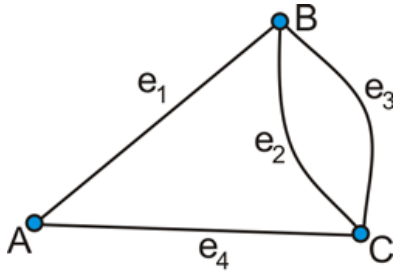
Pada makalah ini akan dibahas tentang penerapan teori graf khususnya pencarian lintasan terpendek pada penentuan jalur akses tercepat dalam pengambilan data/pengiriman data di basis data tersebar. Terlebih dahulu akan diulas sedikit tentang teori graf, basis data tersebar, dan algoritma Dijkstra.

2. GRAF

2.1 TEORI GRAF

Graf (*Graph*) G didefinisikan sebagai pasangan yang dalam hal ini, V merupakan himpunan tidak kosong dari simpul-simpul (vertices / node) = $\{v_1,v_2,...,v_n\}$, dan E adalah himpunan sisi-sisi (edges atau arcs) = $\{e_1,e_2,...,e_n\}$, atau dapat ditulis singkat notasi G = (V,E) [1].

Dapat dikatakan graf adalah kumpulan dari simpul-simpul yang dihubungkan oleh sisi-sisi.



Gambar 1: Graf G

Pada G diatas, graf terdiri dari himpunan V dan E yaitu:

V = {A, B, C}

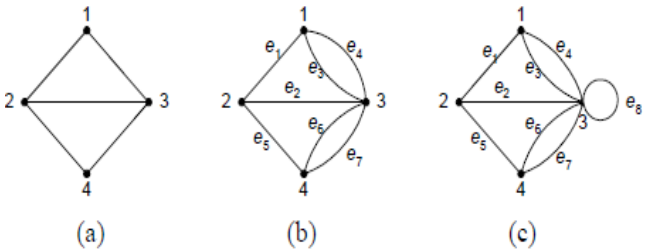
E = $\{e_1, e_2, e_3, e_4\}$

= {(A,B),(B,C),(B,C),(A,C)}

Pengelompokan graf dapat dipandang berdasarkan ada tidaknya sisi ganda atau sisi kalang, berdasarkan jumlah simpul, atau berdasarkan orientasi arah pada sisi.

1. Graf sederhana(*simple graph*)
   Graf yang tidak mengandung gelang maupun sisi-ganda dinamakan graf sederhana.

2. Graf tak sederhana (*unsimple graph*)
   Graf yang mengandung sisi ganda atau gelang dinamakan graf tak-sederhana (*unsimple graph*). Ada dua macam graf tak-sederhana, yaitu graf ganda (*multigraph*) dan graf semu (*pseudograph*). Graf ganda adalah graf yang mengandung sisi ganda. Graf semu adalah graf yang mengandung gelang.



Gambar 2 : Graf berdasarkan ada tidaknya sisi gelang atau sisi ganda, (a). Graf sederhana, (b). Graf ganda, (c). Graf semu

3. Graf berhingga(*limited graph*)
   Graf berhingga adalah graf yang jumlah simpulnya, *n*, berhingga.

4. Graf tak berhingga(*unlimited graph*)
   Graf yang jumlah simpulnya n, tidak berhingga banyaknya disebut graf takberhingga.

5. Graf berarah (*undirected graph*)
   Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah. Kita lebih suka menyebut sisi berarah dengan sebutan busur (*arc*).

6. Graf tak berarah (*directed graph atau digraph*)
   Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah. Pada graf tak-berarah, urutan pasangan simpul yang dihubungkan oleh sisi tidak diperhatikan.

7. Graf berdata bobot (*Weight Graf*)
   Graf berbobot adalah graf yang setiap sisinya diberi sebuah nilai/harga.

2.2. TERMINOLOGI DASAR GRAF

Dalam Teori graf terdapat beberapa terminologi (istilah) yang berkaitan dengan graf yang akan didefinisikan satu persatu sebagai berikut.

1. Bertetangga (Adjacent)
   Dua buah smpul pada graf dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah sisi. Dengan kata lain, v_j bertetangga dengan v_k jika (v_j,v_k) adalah sebuah sisi pada graf G.

2. Bersisian (*Incident*)
   Untuk sembarang sisi $e=(v_j,v_k)$, sisi e dikatakan bersisian dengan simpul v_j dan simpul v_k.

3. Simpul Terpencil (*Isolated Vertex*)
   Simpul terpencil adalah simpul yang tidak mempunyai sisi yang bersisian dengannya. Atau dapat dinyatakan bahwa simpul terpencil adalah simpul yang tidak satupun bertetangga dengan simpul-simpul lainnya.

4. Graf Kosong (*Null Graph* atau *Empty Graph*)
   Graf yang himpunan sisinya merupakan himpunan kosong disebut sebagai graf kosong dan ditulis sebagai N_n, yang dalam hal ini n adalah jumlah simpul.

5. Derajad (*Degree*)
   Derajad suatu simpul pada graf tak berarah adalah jumlah sisi yang bersisian dengan simpul tersebut.
   Pada graf berarah, derajad simpul v dinyatakan dengan $d_{in}(v)$ dan $d_{out}(v)$, yang dalam hal ini $d_{in}(v)$=derajad masuk=derajad busur yang masuk ke simpul ; $d_{out}(v)$=derajad keluar=derajad busur yang keluar ke simpul.
   Untuk sembarang graf G, banyaknya simpul yang berderajad ganjil selalu genap.

6. Lintasan (*Path*)
   Lintasan yang panjangnya n dari simpul awal v_0 ke simpul tujuan v_n di dalam graf G adalah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk $v_0,e_1,v_1,e_2,v_2,e_3,v_3,...,v_{n-1},e_n,v_n$ sedemikian sehingga $e_1=(v_0,v_1)$, $e_2=(v_1,v_2)$,, ... $e_n=(v_{n-1},v_n)$ adalah sisi-sisi dari graf G.

7. Siklus(*Cycle*) atau Sirkuit (*Circuit*)
   Lintasan yang berawal dan berakhir di simpul yang sama disebut sirkuit atau siklus

8. Terhubung (*Connected*)
   Graf Tak Berarah G disebut graf terhubung (*connected graph*) jika untuk setiap pasang simpul v_i dan v_j di dalam himpunan V terdapat lintasan dari v_i ke v_j (yang berarti ada lintasan dari v_i ke v_j).Jika tidak maka G disebut graf tak-terhubung.
   Graf berarah G dikatakan terhubung jika graf tak-berarahnya terhubung (graf tak-berarahnya dari G diperoleh dengan menghilangkan arahnya)

Graf berarah disebut graf terhubung kuat(*strongly connected graph*) apabila untuk setiap pasang simpul sembarang v_i dan v_j di G terhubung kuat. Kalau tidak, G disebut graf terhubung lemah.

9. Upagraf (*Subgraph*) dan Komplemen Upagraf
   Misalkan G=(V,E) adalah sebuah graf. G_1= (V_1,E_1) adalah upagraf (*subgraph*) dari G jika $V_1 \subseteq V$ dan $E_1 \subseteq E$.
   Komplement dari upagraf G_1 terhadap graf G adalah graf G_2= (V_2,E_2) sedemikian sehingga E_2=E-E_1 dan V_2 adalah himpunan simpul yang anggota-anggota E_2 bersisian dengannya.

10. Upagraf Merentang (*Spanning Supgraph*)
    Upagraf G_1 = (V_1,E_1) dari G= (V,E) dikatakan upagraf merentang jika V_1=V(yaitu G_1 mengandung semua simpul dari G).

11. Cut-set
    Cut set dari graf terhubung G adalah himpunan sisi yang bila dibuang dari G menyebabkan G tidak terhubung. Jadi, cut set selalu menghasilkan dua buah komponen terhubung.

12. Graf Berbobot
    Graf Berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot).

3. BASIS DATA TERSEBAR

Basis Data Terdistribusi adalah kumpulan basis-basis data yang saling berhubungan secara logika dan tersebar pada sebuah jaringan komputer [2]. Teknologi Basis Data Tersebar (DDBS: Distributed Database System) merupakan gabungan dari dua pendekatan pengolahan data yang sama sekali berlawanan, yaitu *Database System* dan Teknologi Jaringan Komputer.

Tujuan utama *Database System* adalah mengintegrasikan data dan sentralisasi, sehingga akses (deskripsi, manipulasi dan kontrol) terhadap data sangat terkontrol → *Integration, not Centralization*

Tujuan utama Jaringan Komputer adalah membuat mode kerja yang benar-benar menghindari terjadinya sentralisasi beban kerja.

Teknologi DDBS mengintegrasikan sistem database yang tersebar, untuk menghasilkan suatu teknologi yang hebat dan menjanjikan → *Integration without Centralization.*

DDBS bukan koleksi file yang secara individual disimpan di setiap node pada suatu jaringan komputer. File tidak hanya saling berhubungan secara logis, tapi file-file tersebut harus terstruktur dan pengaksesannya harus melalui interface yang sama.

Distribusi data yang terpisah secara fisik akan menimbulkan masalah yang tidak dijumpai ketika database ada di komputer yang sama. → Inilah yang membedakan antara DDBS dengan Multiprocessor.

DDBS bukan merupakan sistem dimana meskipun ada jaringan komputer, tapi databasenya hanya terletak di satu node dari jaringan tersebut.

Kompleksitas tambahan dengan adanya lingkup kerja yang terdistribusi adalah replikasi data (semua atau sebagian) di jaringan, tapi tidak perlu di semua site. Yang esensial hanya database harus ada di tempat lebih dari satu. Duplikasi ini karena pertimbangan reliability dan efisiensi. DDBS harus memilih satu diantara site yang menyimpan data yang dibutuhkan. Mengupdate setiap site yang menyimpan data yang sudah diupdate. Apabila ada suatu kegagalan (site crash atau link komunikasi putus) selama masa update, sistem harus segera dapat mengupdate site yang tidak tergapai tadi sesegera mungkin (saat sistem tersebut kembali normal). Sinkronisasi transaksi lebih dipertimbangkan lagi dibandingkan sistem yang tersentraliasi (DBS).

3.1. KELEBIHAN BASIS DATA TERSEBAR

Kelebihan dari basis data tersebar antara lain:

- Otonomi Lokal karena data terdistribusi, kelompok user yang bisa menggunakan data tersebut dapat menyimpannya di site di mana dia bekerja, sehingga masing-masing site mempunyai kontrol lokal.
- Performansi Tinggi: karena data yang digunakan umumnya lebih dekat dengan user, maka performansi akses ke database dapat ditingkatkan.
- Kepercayaan: karena ada replikasi data di lebih dari satu tempat, maka kemungkinan crashnya satu node/site, atau hilangnya aksessibilitas karena kegagalan link komunikasi, tidak membuat data tidak dapat diakses.
- Ekonomis:
  - Dari sisi communication cost: Lebih ekonomis dengan membagi aplikasi dan menjalankannya di beberapa situs lokal, dari pada satu aplikasi dipaksa untuk akses ke database yang tersebar.
  - Dari sisi biaya normal: Lebih ekonomis membeli komputer kecil-kecil yang kemampuannya sama dengan satu komputer besar.
- Ekspansinya mudah: Dapat disesuaikan dengan mudah seiring dengan berkembangnya ukuran database. Paling sekitar penambahan pengolahan dan kemampuan penyimpanan di jaringan. Jelas, bahwa kemampuannya tidak akan bertambah secara linier, tapi paling tidak improvisasi dapat dimungkinkan.

- *Shareability*: Untuk merespon gaya organisasi ke arah *distributed fashion*

3.2 KELEMAHAN BASIS DATA TERSEBAR

Kelemahan dari basis data tersebar antara lain:

- Kurangnya pengalaman: minimnya percobaan di lingkup kerja nyata.
- Kompleksitas: menambah masalah baru dari masalah yang ada pada DBS.
- Biaya: menambah biaya hardware (hardware untuk mekanisme komunikasi). Memang Hardware sekarang murah sehingga biaya tidak terlalu signifikan, tapi biaya pengembangan teknologi/software dalam masalah teknis (*distributed debuggers* dll) sangat diperlukan. Selain itu adalah biaya untuk replikasi usaha/tenaga (*manpower*), yang dibutuhkan di beberapa site → Analisa trade-off antara keuntungan dari efisiensi/efektivitas penggunaan informasi dengan biaya penambahan personel.
- Kontrol yang terdistribusi: ini kelebihan yang sudah disebut, tapi sayangnya ini dapat menimbulkan masalah sinkronisasi dan koordinasi
- Keamanan: sekuriti jaringan lebih kompleks dari sekuriti akses database yang tersentralisasi.
- Migrasi yang susah: belum ada tool atau metodologi yang dapat membantu untuk konversi dari DBS ke DDBS.

4. ALGORITMA DIJKSTRA

Algoritma Dijkstra, dinamai menurut penemunya, Edsger Dijkstra, adalah sebuah algoritma rakus (*greedy algorithm*) dalam memecahkan permasalahan jarak terpendek (*shortest path problem*) untuk sebuah graf berarah (*directed graph*) dengan bobot-bobot sisi (*edge weights*) yang bernilai tak-negatif.[4]

Algoritma Dijkstra merupakan salah satu varian bentuk algoritma populer dalam pemecahan persoalan yang terkait dengan masalah optimasi. Sifatnya sederhana dan lempang (*straightforward*). Sesuai dengan arti *greedy* yang secara harafiah berarti tamak atau rakus - namun tidak dalam konteks negatif -, algoritma *greedy* ini hanya memikirkan solusi terbaik yang akan diambil pada setiap langkah tanpa memikirkan konsekuensi ke depan. Prinsipnya, ambillah apa yang bisa Anda dapatkan saat ini (*take what you can get now*!), dan keputusan yang telah diambil pada setiap langkah tidak akan bisa diubah kembali.

Elemen-elemen penyusun prinsip greedy pada Algoritma Dijkstra adalah :

1.Himpunan kandidat

Himpunan ini berisi elemen-elemen yang memiliki peluang untuk membentuk solusi. Pada persoalan lintasan terpendek dalam graf, himpunan kandidat ini adalah himpunan simpul pada graf tersebut.

2. Himpunan solusi

Himpunan ini berisi solusi dari permasalahan yang diselesaikan dan elemennya terdiri dari elemen dalam himpunan kandidat namun tidak semuanya atau dengan kata lain himpunan solusi ini adalah upabagian dari himpunan kandidat.

3. Fungsi seleksi

Fungsi seleksi adalah fungsi yang akan memilih setiap kandidat yang yang memungkinkan untuk menghasilkan solusi optimal pada setiap langkahnya.

4. Fungsi kelayakan

Fungsi kelayakan akan memeriksa apakah suatu kandidat yang telah terpilih (terseleksi) melanggar constraint atau tidak. Apabila kandidat melanggar constraint maka kandidat tidak akan dimasukkan ke dalam himpunan solusi.

5. Fungsi objektif

Fungsi objektif akan memaksimalkan atau meminimalkan nilai solusi. Tujuannya adalah memilih satu saja solusi terbaik dari masing-masing anggota himpunan solusi.

Ada beberapa kasus pencarian lintasan terpendek yang diselesaikan menggunakan algoritma Dijkstra, yaitu:

- pencarian lintasan terpendek antara dua buah simpul tertentu (*a pair shortest path*),
- pencarian lintasan terpendek antara semua pasangan simpul (*all pairs shortest path*),
- pencarian lintasan terpendek dari simpul tertentu ke semua simpul yang lain (*single-source shortest path*),
- pencarian lintasan terpendek antara dua buah simpul yang melalui beberapa simpul tertentu (*intermediate shortest path*).

4.1 PSEUDOCODE

Pseudocode dari algoritma dijkstra adalah sebagai berikut:

```
procedure Dijkstra(input m: matriks, a : simpul awal)
{Mencari lintasan terpendek dari simpul awal a ke semua simpul lainnya.
Masukan: matriks ketetanggaan (m) dari graf berbobot G dan simpul awal a
Keluaran: lintasan terpendek dari a ke semua simpul lainnya}
```

Kamus:
s : array [1..n] of integer
d : array [1..n] of integer
i : integer
Algoritma:
{ Langkah 0 (inisialisasi: }
traversal [1..n]
s_i <- 0
d_i <- m_{ai}
{ Langkah 1: }
s_a<- 1{karena simpul a adalah simpul asal lintasan terpendek, jadi simpul a sudah pasti terpilih dalam lintasan terpendek}
d_a <- ∞ {tidak ada lintasan terpendek dari simpul a ke a}
{ Langkah 2, 3, …, n-1: }
traversal [2..n-1]
cari j sedemikian sehingga s_j= 0 dan
dj = min {d1, d2, …, dn}
sj <- 1 {simpul j sudah terpilih}
perbarui di, untuk i = 1, 2, 3,
sampai dengan n dengan:
d_i(baru) = min{d_i(lama),d_j+ m_{ji}}

4.2 KOMPLEKSITAS WAKTU ALGORITMA

Kompleksitas Waktu Algoritma (*Running time*)

- Algoritma Dijkstra'menggunakan waktu sebesar O(V*log V + E)di mana V dan E adalah banyaknya sisi dan titik.
- Kompleksitas algoritma Dijkstra adalah $O(n^2)$. Sehingga, untuk mencari semua pasangan simpul terpendek, total waktu asimptotik komputasinya adalah: $T(n) = n.O(n^2) = O(n^3)$.

Algoritma dijkstra lebih menguntungkan dari sisi running time.

5. PENENTUAN JALUR AKSES TERCEPAT

Sekarang kita akan mengulas mengenai penerapan algoritma dijkstra pada penentuan jalur akses tercepat dalam pengambilan data/pengiriman data di basis data tersebar. Persoalan jalur akses tercepat ini adalah persoalan optimasi atau sering disebut dengan penentuan lintasan terpendek. Dalam hal ini maksud "terpendek" jangan selalu diasumsikan secara fisik sebagai panjang minimum namun dapat bermakna lain bergantung tipikal persoalan yang sedang dimodelkan. Pada persoalan penentuan jalur akses tercepat ini berhubungan dengan waktu tempuh yang direpresentasikan dengan banyak data dibagi dengan kecepatan aksesnya.

Jika Basis Data tersebar direpresentasikan secara lojik sebagai graf, maka simpul merepresentasikan lokasi/site unit penyimpanan/pemrosesan data, sedangkan sisi merepresentasikan keberadaan hubungan atau koneksi antara dua buah lokasi. Pada hubungan antar dua buah koneksi ini memiliki laju data transmisi (bandwidth) yang menyatakan jumlah maksimum data yang bisa dilewatkan dalam jaringan (baik untuk proses pengambilan/download data maupun pengiriman/upload data, dalam satuan kilobyte). Diasumsikan bahwa tidak ada simpul terpencil pada graf, atau dengan asumsi bahwa tidak ada site yang tidak mempunyai hubungan ke site yang lain. Dalam hal ini untuk memperoleh waktu akses digunakan rumus jumlah data dibagi dengan bandwidth. Bobot graf pada persoalan ini adalah satu per bandwidth, yang nantinya total harganya akan dikalikan dengan jumlah data yang dikirim atau diambil yang kemudian akan menghasilkan waktu akses yang paling cepat dari sekian banyak kemungkinan jalur yang ada. Sehingga pada pemodelan persoalan ini algoritma Dijkstra bisa digunakan untuk mencari waktu akses minimum dari bobot graf yang ada.

Ketika data akan diakses dari suatu site ke site yang lain dapat dicari jalur mana yang punya nilai minimum dan dapat diketahui juga simpul mana saja yang dilewati. Sehingga waktu upload atau download suatu data dapat lebih efektif dengan penggunaan algoritma dijkstra ini..

6. KESIMPULAN

Algoritma Dijkstra merupakan algoritma pencarian lintasan terpendek yang biasa digunakan pada graf berarah maupun untuk graf tak berarah. Algoritma ini dapat diterapkan dengan tepat pada persoalan penentuan jalur akses tercepat dalam pengambilan data atau pengiriman data dari satu site ke site lainnya pada sistem basis data terpusat yang terlebih dahulu memodelkan sistem basis data terpusat tersebut ke dalam graf berbobot, kemudian algoritma dijkstra bisa dilaksanakan untuk menentukan jalur akses tercepat dari satu site ke site yang lain sehingga untuk mengakses suatu data dapat dengan waktu yang paling minimal sesuai dengan bandwidth jaringan yang ada antar site.

REFERENSI

[1] Munir, Rinaldi. (2008). Diktat Kuliah IF2091 Struktur Diskrit Edisi Keempat. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[2] http://w-h-tono.eduspoke.com/course/category.php?id=3. Tanggal akses 17 Desember 2009 pukul 10.00.

[3] Haragi.wordpress.com/2007/09/16/basis-data-tersebar/ Tanggal akses 18 Desember 2009, pukul 11.15.

[4] Algoritma Dijkstra - Wikipedia bahasa Indonesia, ensiklopedia bebas, http://id.wikipedia.org/wiki/Algoritma_Dijkstra. Tanggal akses 17 Desember 2009, pukul 10.05

[5] http://www.rkasigi.net/php/algoritma-dijkstra-mencari-jarak-terpendek-11.html Tanggal akses 17 Desember 2009 pukul 10.30