Perbandingan Kompleksitas Algoritma Pohon Merentang Minimum dan Aplikasinya

Arvin Yustin / 13517124
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517124@std.stei.itb.ac.id
arvinyustin99@gmail.com

I. PENDAHULUAN

Matematika diskrit adalah salah satu cabang matematika yang mempelajari objek diskrit (yang dapat dihitung dengan bilangan bulat / diskontinu). Salah satu bahasan yang dipelajari ialah graf. Graf sangat diperlukan dalam berbagai cabang ilmu lainnya, seperti pengaturan lalu lintas, perencanaan pemasangan tiang listrik, pipa air, fiber optik, penyaluran *package* data informasi melalui jaringan internet, dan jalor transportasi umum bawah tanah, seperti MRT (*Mass Rapid Transit*). Begitu juga dengan mesin jaja (*vendor machine*), yang menggunakan teori graf dalam merumuskan bahasa-bahasa yang diterima (dipelajari dalam mata kuliah Teori Bahaa Formal dan Otomata). Gawai yang kita pakai sehari-hari juga merupakan aplikasi dari teori graf, dimana kita menekan sebuah tombol virtual akan mengarahkan tampilan layar yang berbeda.

Dalam biokimia, graf dipakai untuk taksonomi makhluk hidup, dan penggolongan obat-obat farmasetikal.

Penggunaan graf dibutuhkan mengingat banyak hal yang perlu disusun sedemikian sehingga terstruktur, terorganisir, sehingga ketika dicari/atau disusun ulang, tidak menghilangkan / mengurangi / manipulasi data informasi yang bersangkutan.

Selain itu cabang khusus dari graf adalah pohon, dimana sangat berguna dalam bidang *trading* (seperti *blockchain*), kriptografi (penyimpanan kata sandi dalam dunia maya (*hash*), pengiriman pesan yang terenkripsi (*SHA*)), basis data (penyimpan data-data dalam bentuk *list*).

II. TEORI DASAR DAN PENGENALAN ISTILAH

Graf merupakan struktur diskrit yang terdiri dari simpul dan sisi. Lebih formal, Graf merupakan *tuple* dari himpunan tidak kosong simpul (*vertices*) dan himpunan sisi (*edges*)

$$G = (E, V)$$

- *Degree* / *valency* (derajat) simpul pada graf ialah banyaknya sisi yang dapat dibentuk oleh simpul tersebut, baik ke simpul lainnya maupun simpul sendiri
- *Adjacent* (bertetanggaan), dikatakan dua simpul bertetanggaan jika kedua simpul tersebut dihubungkan langsung oleh satu atau lebih sisi.
- *Incident* (bersisian), dikatakan sebuah simpul dan sebuah sisi bersisian jika keduanya terhubung langsung
- *Path* (lintasan), himpunan simpul dan sisi yang membentuk sebuah jalur.
- *Cycle* (sirkuit), lintasan yang tertutup, dimana simpul awalnya juga menjadi simpul terakhir.
- *Connected* (terhubung), dikatakan dua simpul terhubung, jika diantara kedua simpul tersebut mempunyai lintasan paling sedikit satu buah sehingga dari satu simpul dapat mencapai simpul kedua.
- *Subgraph* (subgraf) adalah graf yang sisi dan simpulnya merupakan himpunan bagian dari graf yang lebih besar.

- Subgraf yang simpulnya terdiri dari semua simpul graf disebut *subgraf merentang*
- Graf berbobot (*weighted graph*) adalah graf dengan sisinya yang bernilai bilangan *riil* (*real number*).

Lemma Jabat Tangan (*The Handshaking Lemma*): Jumlah derajat semua simpul pada sebuah graf dua kali jumlah sisi pada graf tersebut

Pohon (*tree*) merupakan graf yang tidak memiliki sirkuit di dalamnya. Setiap simpulnya *terhubung*. Pohon merentang adalah pohon yang dibentuk dari sebuah graf, dimana melibatkan semua simpul pada graf tersebut. Pohon merentang minimum (*minimum spanning tree*) adalah pohon merentang yang memiliki jumlah bobot dari semua sisinya sekecil mungkin.

Dalam kompleksitas algoritma terdapat istilah *big-O notation* atau kompleksitas asimptotik, yang menyatakan batas atas sebuah fungsi ketika *domain* menuju tak hingga. Notasi O besar selalu memakai satu buah suku, dimana suku tersebut merupakan suku yang paling dominan dalam sebuah fungsi (memberikan efek paling besar).

III. PENJELASAN SINGKAT ALGORITMA POHON MERENTANG MINIMUM

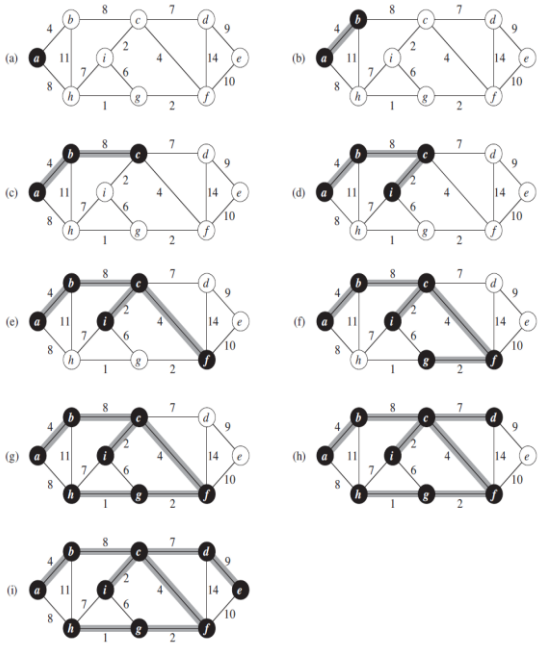
Pada pembahasan ini, akan dikaji empat jenis algoritma dalam mencari pohon merentang minimum, antara lain: algoritma Prim, algoritma Kruskal, algoritma Boruvska, dan algoritma *reverse-delete*.

Kita mengasumsikan bahwa graf yang dipakai merupakan graf sederhana, tidak berulang sisinya (*loop*), dan tidak berarah.

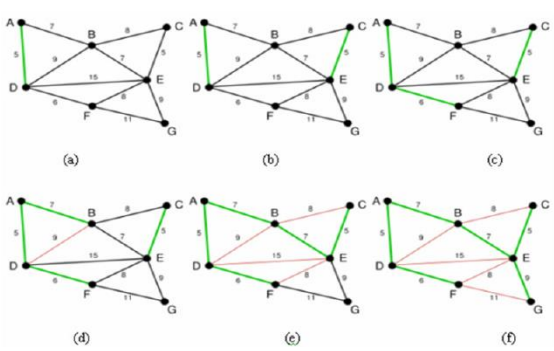
a. Algoritma Prim (*Jarnik's algorithm*)

Algoritma Prim merupakan algoritma visual, yang memberi kebebasan kepada pengguna untuk memilih simpul mulainya. Simpul dapat dimulai dari mana saja, karena pada akhirnya semua simpul akan menjadi pohon. Algoritma Prim awalnya ditemukan oleh *Vojtech Jarnik* (1930), kemudian dikemukakan lagi oleh *Robert C. Prim* (1957) dan *Edsger W. Dijkstra* (1959).

i. Pilih salah satu simpul menjadi pohon awal
ii. Kemudian pilih simpul yang *adjacent* dengan simpul (i) dengan bobot pada sisinya paling kecil
iii. Sekarang kita mempunyai pohon dengan simpul (i) dan simpul (ii). Ulangi langkah (i) dari simpul pohon yang sudah kita bentuk sampai tidak ada simpul yang tertinggal dalam graf tersebut



Gambar 1. Tahap-tahap algoritma Prim
https://i.stack.imgur.com/TTwpR.png



Gambar 2. Tahap-tahap algoritma Kruskal
https://www.researchgate.net/profile/Yael_Jacob/publication/221923505/figure/fig5/AS:305090574471176@1449750670012/An-example-of-how-the-Kruskal-algorithm-can-be-used-in-order-to-find-the-minimal-spanning.png

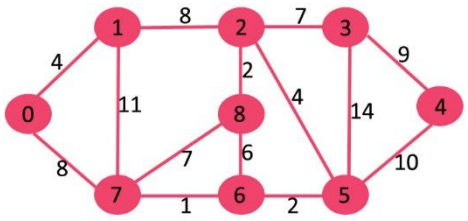
b. Algoritma Kruskal

Algoritma ini mengurutkan sisi graf membesar, kemudian mengutip satu per satu sisi dari yang terkecil, kemudian membentuk sebuah pohon merentang.

i. Buatlah sebuah kumpulan sisi-sisi graf
ii. Urutkan sisi-sisi tersebut secara membesar
iii. Gambarkan semua simpul sesuai graf asli tanpa sisinya
iv. Kutip dari sisi yang bobotnya terkecil
v. Kutip sisi selanjutnya terkecil setelah sisi (iv)
vi. Jika sisi yang dikutip membentuk sebuah sirkuit, maka kutip sisi terkecil selanjutnya
vii. Ulangi langkah (iv) hingga (vi) hingga semua simpul dapat terhubung dengan lainnya maksimal 1 lintasan

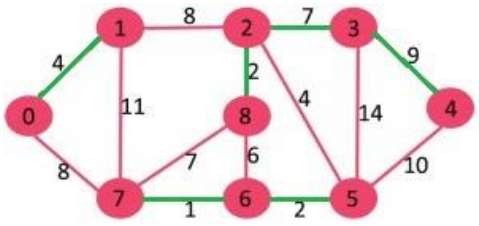
c. Algoritma Boruvska

Pada algoritma ini, masing-masing simpul sebagai sebuah komponen dikerjakan secara bersamaan. Setiap simpul mencari sisi yang *incident* dengan bobot terkecil. Akan terbentuk komponen baru terdiri dari 1 atau lebih simpul.



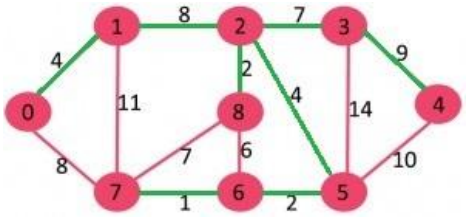
Gambar 3. Graf awal
https://www.geeksforgeeks.org/wp-content/uploads/Fig-0.jpg

i. Tentukan masing-masing sisi dengan bobot terkecil pada masing-masing simpul



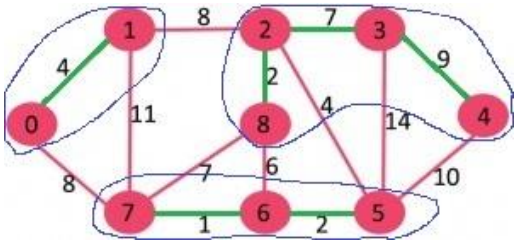
Gambar 4. Subpohon yang ditandai dengan hubungan garis berwarna hijau
https://www.geeksforgeeks.org/wp-content/uploads/13.jpg

ii. Ulangi langkah (i) dengan komponen sebagai sebuah 'simpul'. Jika terdapat sisi yang berbobot sama, cari sisi yang bersisian dengan simpul yang derajatnya lebih tinggi



Gambar 5.
https://www.geeksforgeeks.org/wp-content/uploads/131.jpg

iii. Hentikan pencarian jika komponen tinggal satu buah

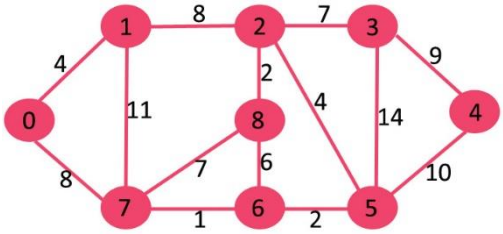


Gambar 6. Terbentuk pohon merentang minimum
https://www.geeksforgeeks.org/wp-content/uploads/14.jpg

d. Algoritma *Reverse-delete*

Algoritma ini berbalikkan dengan algoritma Kruskal. Dimana titik mulainya dari sisi dengan bobot yang paling besar dan membuangnya.

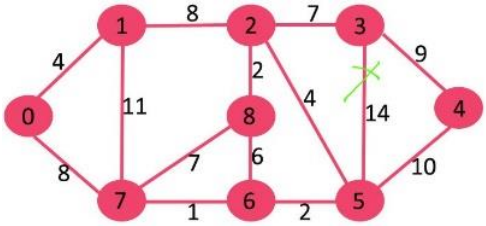
i. Gambarkan simpul sesuai dengan graf awal.



Gambar 7. Graf Awal
https://www.geeksforgeeks.org/wp-content/uploads/Fig-0.jpg

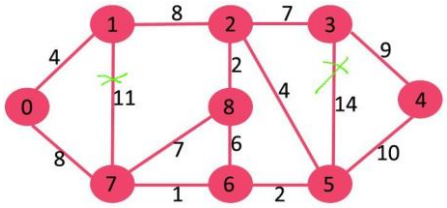
ii. Urutkan sisi-sisi graf mengecil

iii. Buang sisi terbesar, periksa apakah graf awal menjadi terputus. Jika terputus, jangan sisi yang tadi jangan dibuang.

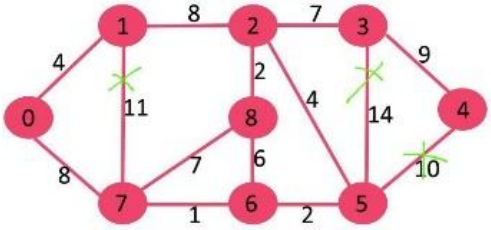


Gambar 8. Hapus Sisi dengan bobot terbesar
https://www.geeksforgeeks.org/wp-content/uploads/ReverseDelete2.jpg

iv. Ulangi langkah (i) hingga masing-masing simpul terhubung dengan lintasan sebanyak 1 buah



Gambar 9. Tahap-tahap algoritma *Reverse-delete*



https://www.geeksforgeeks.org/reverse-delete-algorithm-minimum-spanning-tree/

IV. PERBANDINGAN KOMPLEKSITAS ALGORITMA

Algoritma Prim pada graf sederhana memiliki kompleksitas waktu

$$O(E^2)$$

Karena setiap tahap, algoritma akan memeriksa sisi yang bersisian dengan simpul tersebut.

Sedangkan untuk pada algoritma Kruskal, pertama kita mengurutkan sisi yang berbobot terurut membesar, kita asumsikan menggunakan *quick sort*, kompleksitasnya

$$O(E \log V)$$

Pada algoritma Boruvska, terdapat tahap *traversal* untuk mencari simpul sehingga $O(\log V)$, kemudian setiap simpul mencari satu buah sisi, sehingga, total kompleksitasnya,

$$O(E \log V)$$

Hal yang sama terjadi antara algoritma *Reverse-delete* dan algoritma *Kruskal*, dimana perbedaan keduanya hanyalah jalan mulai dan keberlangsungannya. Kruskal memulainya dari sisi dengan nilai bobot terkecil dan menambahkannya ke pohon, sedangkan *Reverse-delete* menghapus nilai bobot terbesar. Sehingga

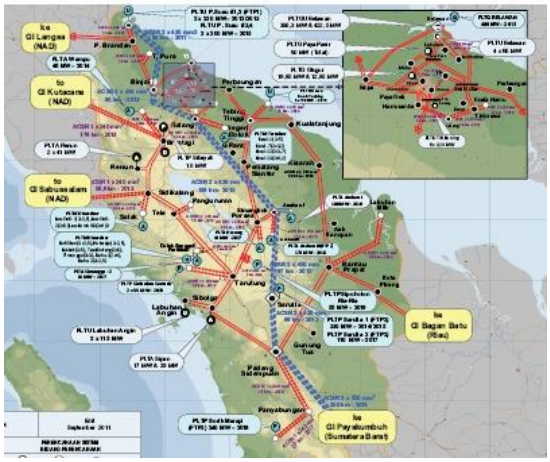
$$O(E \log V)$$

Dari empat buah kompleksitas masing-masing algoritma pohon merentang minimum di atas, algoritma Kruskal, Boruvska, dan *Reverse-delete* mengungguli algoritma Prim. Namun, algoritma Boruvska mengungguli algoritma Kruskal dan *Reverse-delete*. Karena kedua algoritma tersebut harus mengurutkan sisi-sisi pada graf tersebut, sehingga, tidak efisien untuk kasus simpul dan sisi yang berjumlah masif.

V. APLIKASI DAN IMPLEMENTASI ALGORITMA

Implementasi *pohon merentang minimum* dapat ditemukan di sekitar kita. Sebelum memulai pembangunan tiang listrik, perusahaan penyedia listrik akan turun ke lapangan dan meninjau tempat-tempat yang akan dipasangi tiang listrik dan trafo. Mereka akan menghitung lebar jalan, banyak-sedikitnya akses jalan, kepadatan penduduk, dan jarak ke jalan utama. Jika jalanannya sempit dan tidak padat penduduk, maka cukup dipasangkan tiang listrik yang bercabang dari jalan utama.

Jika jalanannya lebar dan padat penduduk, mereka menempatkan trafo di jalan tersebut, sehingga seluruh penduduk setempat dapat menggunakan sediaan listrik.

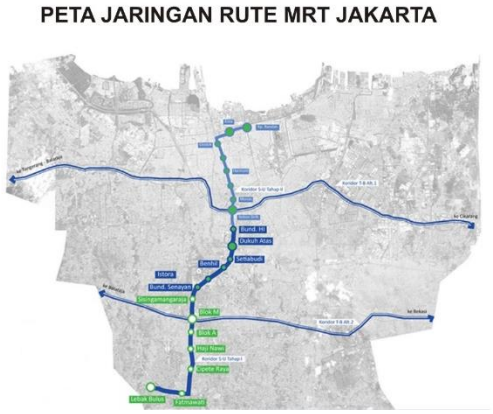


Gambar 10. Distribusi Transmisi Listrik Daya Tinggi di Sumatera Utara
https://repit.files.wordpress.com/2011/05/sumut.jpg

Dalam penerbangan udara, maskapai dan pilot mencari rute paling aman, dan tersingkat, agar cepat sampai tujuan, tanpa melupakan keuntungan yang lebih tinggi (biasanya dihitung dalam bentuk keuntungan per *seat*). Selain itu, mereka juga mempertimbangkan wilayah udara negara yang akan dilalui, apakah rawan gangguan atau ancaman dari daratan, biaya melewati ruang udara negara, dan jika terdapat kejadian darurat, terdapat bandara terdekat untuk mendarat.

Gambar 11. Tampilan *Waypoint* yang ditandai dengan bulatan warna biru
http://www.vfrflight.org/img/sshots/1.jpg

Dalam komputasi graf pada penerbangan udara, setiap *waypoint* atau titik tujuan, bandar udara bertindak sebagai simpul, dan rute atau jalur bertindak sebagai sisi.



Gambar 12. Peta Jaringan Rute MRT Kota Jakarta
https://siranselhitam.files.wordpress.com/2015/03/peta-rute-mrtj.jpg

Dinas perhubungan kotamadya bertugas untuk mencari tujuan-tujuan transportasi umum sehingga, rute-rute yang mereka lalui tidak searah ataupun saling memotong, tidak melalui jalur yang sama, dapat dijangkau oleh warga (dalam hal ini, jarak), dan memerlukan biaya operasional seminim mungkin. Jika dilihat dengan seksama, dalam sebuah kota, akan terdapat pangkalan utama dimana semua rute yang berbeda akan saling bertemu.

VI. KESIMPULAN

Algoritma pencarian pohon merentang minimum sangat esensial dalam kehidupan sehari-hari, maupun dalam cabang ilmu lain. Dari perbandingan kompleksitas, terlihat algoritma Boruvska sedikit mengungguli algoritma *Kruskal* dan *Reverse-delete*. Dan dibawahnya terdapat algoritma *Prim*. Meskipun begitu, sampai saat ini algoritma *Kruskal* lebih populer ketimbang algoritma *Boruvska* karena algoritma *Kruskal* dapat ditelusuri secara traversal dan lebih mudah dimengerti. Pada kenyataannya, algoritma *Boruvska* yang kelihatan cepat karena '*multi-tasking*' ketika dikomputasikan, akan sama cepatnya dengan kedua algoritma lainnya. Selain itu, algoritma *Kruskal* dan *Reverse-delete* akan berjalan lebih cepat dengan *prekondisi* semua elemen disusun sudah terurut.

VII. UCAPAN TERIMA KASIH

Puji Tuhan atas berkat yang diberi-Nya sehingga saya bisa menyelesaikan makalah ini tepat waktu. Tidak lupa juga kepada pengajar mata kuliah Matematika Diskrit IF2120 K1, yaitu Bapak Rinaldi Munir atas bimbingannya selama ini. Mohon maaf jika terdapat kekurangan baik selama di kelas maupun isi di dalam makalah ini. Dan kepada pihak yang tidak dapat saya sebutkan satu per satu, terima kasih telah membantu saya dalam proses pembuatan makalah ini.

VIII. PUSTAKA

[1]. https://www.ics.uci.edu/~eppstein/161/960206.html (diakses pada tanggal 6 Desember 2018)

[2]. https://www.tutorialspoint.com/discrete_mathematics/graph_and_graph_models.htm (diakses pada tanggal 6 Desember 2018)

[3]. http://www.uniksharianja.com/2015/06/mengenal-matematika-diskrit-atau-matematika-informatika.html (diakses pada tanggal 8 Desember 2018)

[4]. https://www.thecrazyprogrammer.com/2014/03/kruskals-algorithm-for-finding-minimum-cost-spanning-tree.html (diakses pada tanggal 9 Desember 2018)

[5]. Rosen, K.H., *Discrete Mathematics and Its Applications*, New York: McGraw-Hill, 2003, edisi kelima

[6]. Rosen, K.H., *Discrete Mathematics and Its Applications*, New York: McGraw-Hill, 2012, edisi ketujuh.

[7]. http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Graf%20(2015).pdf (diakses pada tanggal 9 Desember 2018)

[8]. http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf (diakses pada tanggal 8 Desember 2018)

[9]. http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2015-2016/Kompleksitas%20Algoritma%20(2015).pdf (diakses pada tanggal 8 Desember 2018)

PERNYATAAN

Dengan ini, saya menyatakan bahwa makalah yang saya tulis adalah hasil dari tulisan saya sendiri, bukan salinan, terjemahan maupun plagiasi dari orang lain

Bandung, 9 Desember 2018

13517124