Modul 4

Pohon

Dr. Nanang Priatna, M.Pd.

PENDAHULUAN

Dalam modul-modul sebelumnya Anda telah mempelajari graph terhubung tanpa sikel, misalnya model graph untuk molekul C_4H_{10}, hierarki administrasi suatu organisasi, pengklasifikasian buku-buku di perpustakaan, pensortiran surat-surat di kantor pos sebelum disampaikan ke alamat masing-masing, dan silsilah raja. Graph semacam ini dikenal sebagai *pohon*.

Kirchoff (1824 - 1887) mengembangkan teori pohon untuk diterapkan dalam jaringan listrik. Selanjutnya **Arthur Cayley (1821 - 1895)** mengembangkan graph jenis ini sewaktu mencacah isomer hidrokarbon jenuh C_nH_{2n+2}. Sekarang pohon digunakan secara luas dalam linguistik dan ilmu komputer.

Modul 4 ini terdiri dari 2 kegiatan belajar. Kegiatan Belajar 1 menguraikan tentang sifat-sifat pohon, pohon rentang, pohon berakar dan pohon jumlah. Sedangkan pada Kegiatan Belajar 2 akan dijelaskan mengenai algoritma Kruskal, algoritma Prim, dan pohon Biner.

Setelah mempelajari modul ini, diharapkan Anda dapat memahami pengertian pohon, pohon rentang, pohon berakar, pohon jumlah, algoritma Kruskal, algoritma Prim, dan pohon Biner. Selain itu, Anda diharapkan dapat menggunakannya dalam kehidupan sehari-hari dan dapat mengajarkannya dengan pendekatan tertentu yang sesuai. Secara khusus tujuan dari penyajian modul ini adalah agar Anda dapat:

a. menjelaskan pengertian pohon pada teori graph,
b. membuktikan sifat-sifat pohon,
c. menentukan pohon rentang dari beberapa graph,
d. menjelaskan pengertian pohon berakar,
e. menentukan pohon jumlah dari suatu graph,
f. menentukan pohon jumlah minimal dengan algoritma Kruskal,
g. menentukan pohon biner optimal dengan algoritma Huffman.

Kegiatan Belajar 1

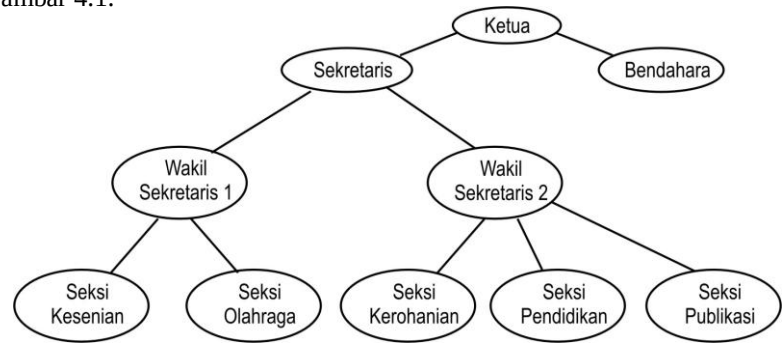
Pohon, Pohon Berakar, dan Pohon Jumlah

Pada bagian ini Anda akan mempelajari graph khusus yang disebut pohon (*tree*). Contoh-contoh dan sifat-sifat penting akan disajikan di bagian awal, sedangkan beberapa permasalahan nyata yang berkaitan dengan pohon diberikan pada bagian akhir.

DEFINISI 1

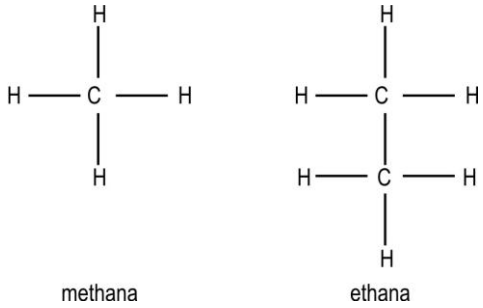
Pohon ialah graph terhubung yang tidak memiliki sikel.

Contoh 1
Hierarki administrasi organisasi OSIS suatu SMA berbentuk seperti pada Gambar 4.1.



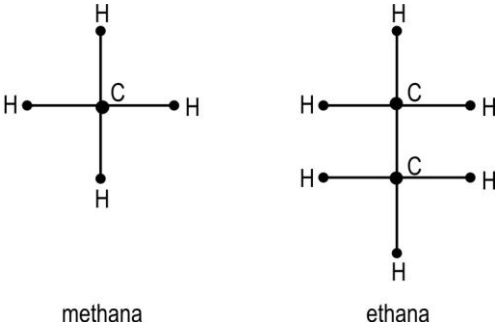
Gambar 4.1

Contoh 2
Pada Tahun 1857, Arthur Cayley mempelajari hidrokarbon, ikatan kimia yang terbentuk dari atom hidrogen dan karbon. Dia mengetahui bahwa atom hidrogen terikat (secara kimia) dengan satu atom yang lainnya, dan setiap atom karbon terikat dengan empat atom lainnya. Perhatikan Gambar 4.2 berikut.



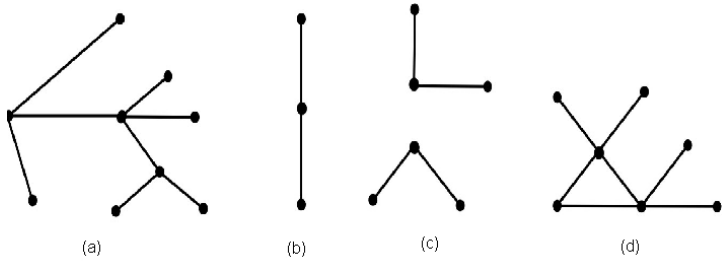
Gambar 4.2

Diagram kimia di atas dapat digambar kembali sebagai graph yang diilustrasikan pada Gambar 4.3.



Gambar 4.3

Contoh 3



Gambar 4.4

Gambar 4.4 (a) dan 4.4 (b) merupakan pohon, sedangkan Gambar 4.4 (c) dan 4.4 (d) bukan pohon, karena Gambar 4.4 (c) graphnya tidak terhubung, sedangkan Gambar 4.4 (d) graphnya memiliki sikel.

Beberapa sifat dasar dari sebuah pohon, akan diuraikan dalam teorema-teorema berikut.

Teorema 1

Jika T pohon, maka untuk setiap dua titik *u* dan *v* yang berbeda di T terdapat tepat satu lintasan (*path*) yang menghubungkan kedua titik tersebut.

Bukti

Misalkan ada lintasan (*path*) berbeda yang menghubungkan titik u dan titik v di T, katakanlah e_1 dan e_2, dengan $e_1 \neq e_2$. Maka e_1 dan e_2 akan menghubungkan titik u dan titik v, sehingga ada dua lintasan yang terhubung pada kedua titik tersebut dan membentuk sikel. Berdasarkan definisi, T tidak memiliki sikel. Dengan demikian, haruslah $e_1=e_2$. Hal ini bertentangan dengan pemisalan bahwa $e_1 \neq e_2$. Jadi, terbukti bahwa setiap dua titik yang berbeda di T memiliki tepat satu lintasan yang menghubungkan kedua titik tersebut.

Teorema 2

Banyaknya titik dari sebuah pohon T sama dengan banyaknya sisi ditambah satu atau ditulis:

Jika T pohon, maka $|V(T)| = |E(T)| + 1$

Bukti

Kita buktikan teorema di atas dengan induksi pada |V(T)|. Jika pohon T mempunyai satu titik, jelas banyak sisi T adalah nol. Jadi teorema benar untuk pohon T dengan satu titik. Asumsikan bahwa pernyataan dalam teorema benar untuk pohon dengan k titik, artinya jika pohon T mempunyai paling banyak k titik, maka $|V(T)| = |E(T)| + 1$.

Akan ditunjukkan bahwa jika pohon T mempunyai k + 1 titik maka $|V(T)| = |E(T)| + 1$. Misalkan T adalah pohon dengan k + 1 titik dan ℓ adalah sebuah sisi T. Maka T - ℓ memiliki tepat dua komponen T_1 dan T_2, dan masing-masing komponen adalah pohon dengan titik kurang dari k + 1. Sehingga menurut asumsi, $|V(T_i)| = |E(T_i)| + 1$; $i = 1,2$.

Selanjutnya $|E(T)| = |E(T_1)| + |E(T_2)| + 1$, sehingga

$$\begin{aligned} |V(T)| &= |V(T_1)| + |V(T_2)| \\ &= |E(T_1)| + 1 + |E(T_2)| + 1 \\ &= (|E(T_1)| + |E(T_2)| + 1) + 1 \\ &= |E\,(T)| + 1 \end{aligned}$$

Dengan demikian teorema terbukti.

Contoh 4

Dinas rahasia Amerika Serikat (CIA) telah membentuk jaringan kerja antara 10 agen rahasia yang bekerja di bidang teknologi industri tingkat tinggi. Penting bagi setiap agen untuk dapat berkomunikasi dengan agen lain secara langsung atau tidak langsung (salah satu) melalui suatu mata rantai. Penentuan lokasi rahasia sulit dilakukan, dan CIA ingin agar ada sesedikit mungkin tempat-tempat pertemuan rahasia itu.

Untuk menjaga kerahasiaan, tidak lebih dari dua agen yang boleh mengetahui tempat pertemuan. Jaringan komunikasi ini dapat disajikan dengan graph yang titiknya menunjukkan agen rahasia dan sisinya menghubungkan dua titik jika agen itu mengetahui tempat pertemuan yang sama. Ternyata graph ini merupakan pohon dengan 10 titik, sehingga semuanya diperlukan sembilan tempat pertemuan rahasia.

Teorema 3

a. Bila suatu sisi dihapus dari pohon (dan titiknya tetap), maka diperoleh graph yang tidak terhubung, dan karenanya graph itu bukan pohon.
b. Bila sebuah sisi ditambahkan pada pohon (tanpa menambah titik baru), diperoleh graph yang memiliki sikel, dan karena itu graph tersebut bukan pohon.

Bukti

Jika sebuah sisi ditambahkan atau dihapuskan dari pohon, graph baru yang diperoleh tidak lagi merupakan pohon, berdasarkan teorema 2. Karena penghapusan sebuah sisi menjadikan graph itu tidak terhubung, dan penambahan sisi membentuk sikel, maka teorema terbukti.

Teorema berikut memberikan cara-cara tertentu untuk mengkarakterisasikan pohon. Pembuktiannya dikerjakan sebagai latihan.

Teorema 4

Pernyataan berikut ini ekuivalen untuk pohon T.

a. T adalah pohon.
b. T terhubung dan banyak titiknya lebih satu dari banyak sisinya.
c. T tidak memiliki sikel dan banyak titiknya lebih satu dari banyak sisinya.
d. Ada tepat satu lintasan (*path*) sederhana antara setiap dua titik di T.
e. T terhubung dan penghapusan sembarang sisi pada T menghasilkan graph yang tidak terhubung.
f. T tidak memiliki sikel dan penambahan sembarang sisi menghasilkan sikel pada graph itu.

Teorema 5

Jika P = $(v_0, v_1, v_2, ..., v_n)$ sebuah lintasan terpanjang di pohon T, maka $d(v_0) = d(v_n) = 1$.

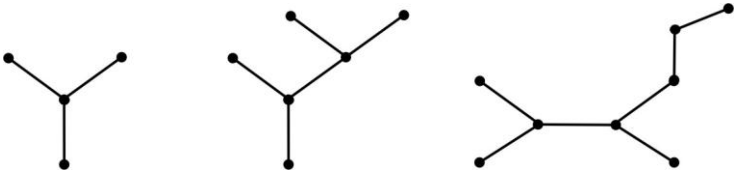
Bukti

Misalkan T adalah sebuah pohon dari P = $(v_0, v_1, v_2, ..., v_n)$ lintasan terpanjang di T. Andaikan $d(v_0) \neq 1$, maka $d(v_0) > 1$. Berarti ada paling sedikit dua sisi T yang terkait (insiden) dengan v_0. Misalkan $e \neq v_0v_1$ adalah sisi T yang terkait dengan v_0. Karena P lintasan terpanjang di T dan T sederhana, sisi e harus menghubungkan v_0 dengan sebuah titik lain di P, katakan titik v_k, $k \neq 0,1$. Akibatnya $(v_0, v_1, v_2, ..., v_k, v_n)$ membentuk sikel di T, ini bertentangan dengan kenyataan bahwa T sebuah pohon. Dengan kata lain $d(v_0) = 1$. Dengan argumen yang serupa, dapat ditunjukkan $d(v_n) = 1$. Teorema terbukti.

DEFINISI 2

Hutan adalah graph tanpa sikel.

Contoh 5



Gambar 4.5

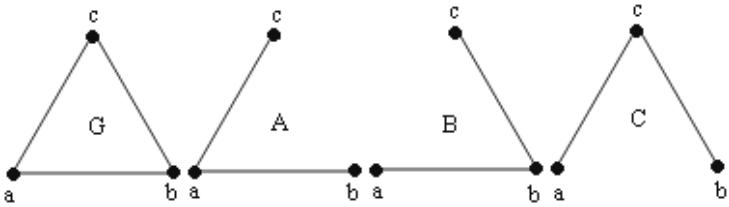
Gambar 4.5 adalah suatu hutan yang terdiri atas 3 komponen.

DEFINISI 3

Misalkan G adalah sebuah graph. Sebuah pohon di G yang memuat semua titik G disebut *pohon rentang (spanning tree*) dari G.

Contoh 6

Misalkan kita mempunyai graph G seperti pada gambar 4.6 di bawah ini. Terdapat 3 pohon rentang dari graph G, yaitu graph A, B, dan C. Tampak jelas bahwa graph A, B, dan C masing-masing memuat semua simpul dari graph G serta mengandung sisi-sisi dari G demikian sehingga tidak terbentuk sikel.



Gambar 4.6

Teorema 6

Graph G terhubung jika dan hanya jika G memuat pohon rentang.

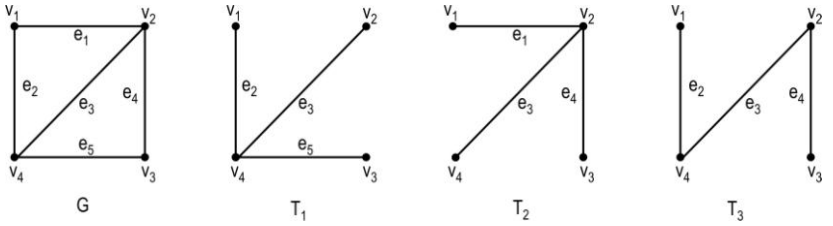
Bukti

Jika graph G memuat pohon rentang, jelas G terhubung. Kita buktikan konvers pernyataan ini dengan induksi pada |E(G)|. Jika G terhubung dan |E(G)| = 0, maka G = K_1, sehingga jelas G memuat pohon rentang.

Asumsikan: setiap graph terhubung dengan k + 1 sisi, maka G memuat pohon rentang.

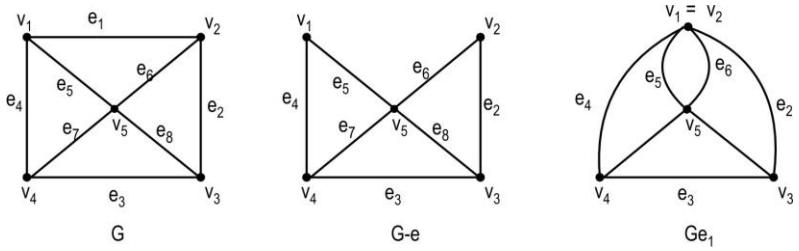
Pandang sebuah graph terhubung G dengan k + 1 sisi. Jika G tidak memuat sikel, maka G sebuah pohon rentang. Jika G memuat sikel, dan misalkan e adalah sebuah sisi dari sikel di G, maka graph G_1 = G - e terhubung dengan k sisi. Sehingga berdasarkan asumsi, G_1 memuat pohon rentang. Sebut T, pohon rentang di G_1. Jelas, T adalah juga pohon rentang dari G. Teorema terbukti.

Sebuah graph terhubung mungkin memuat lebih dari satu pohon rentang, seperti terlihat pada Gambar 4.7. Graph G memuat pohon rentang T_1, T_2, dan T_3.



Gambar 4.7

Selanjutnya τ (G) melambangkan banyaknya pohon rentang pada graph G. Adakah cara yang dapat dipakai untuk menentukan banyaknya pohon rentang di dalam sebuah graph? Jawabnya akan diberikan dalam teorema berikut, namun sebelumnya kita perlu memahami notasi-notasi berikut. Misalkan e adalah sebuah sisi dari graph G. Sebuah graph yang diperoleh dari G dengan menghapus sisi e dari G dan menyatukan kedua titik akhir dari e, dilambangkan dengan G.e (lihat Gambar 4.8).



Gambar 4.8

Perhatikan bahwa banyaknya komponen G sama dengan banyaknya komponen G.e, begitu pula |E(G.e)| = |E(G)| - 1 dan |V(G.e)| = |V(G)| - 1. Perlu dicatat bahwa jika T adalah pohon dan e sisi dari T, maka T.e juga sebuah pohon.

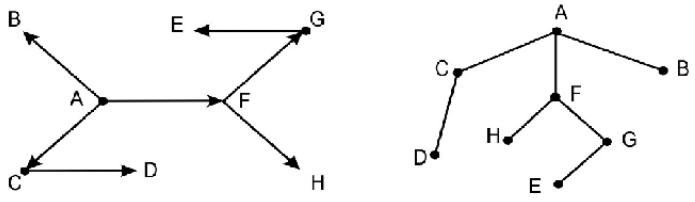
DEFINISI 4

Pohon berakar adalah graph berarah (*digraph*) T yang mempunyai dua syarat:

1. Bila arah sisi-sisi pada T diabaikan, hasil graph tidak berarahnya merupakan sebuah pohon, dan
2. Ada titik tunggal R sedemikian hingga derajat masuk R adalah 0 dan derajat masuk sembarang titik lainnya adalah 1. Titik R disebut akar dari pohon berakar itu.

Contoh 7

Graph pada Gambar 4.9 (a) adalah pohon berakar dengan akar A karena (1) bila arah pada sisinya diabaikan, graph hasilnya merupakan pohon, dan (2) A berderajat masuk 0, dan semua titik lainnya berderajat masuk 1. Cara biasa untuk menggambarkan graph itu ditunjukkan pada Gambar 4.9 (b).

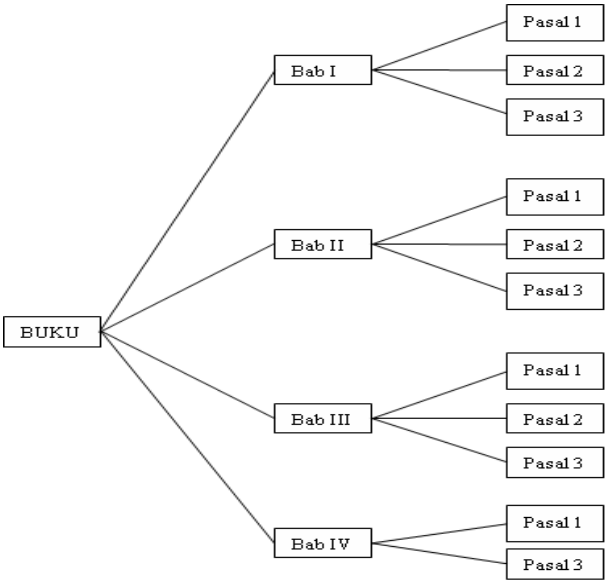


Gambar 4.9

Titik-titik D, H, E, dan B disebut *titik terminal*, yaitu titik dengan derajat keluar 0. Sedangkan titik-titik A, C, F, dan G disebut *titik internal*, yaitu titik yang memiliki derajat keluar yang tidak nol.

Contoh 8

Dalam gambar di bawah ini konsep pohon berakar digunakan untuk menggambarkan hubungan antara pasal-pasal dan bab-bab dalam sebuah buku.



Gambar 4.10

Teorema 7

Pada pohon berakar dengan akar R:

a. Banyak titik lebih satu dari banyak sisi berarah.
b. Tidak ada sikel berarah.
c. Ada *path* sederhana berarah yang tunggal dari R ke setiap titik lain.

Bukti

Pembuktian (a) dan (b) segera diperoleh karena T menjadi pohon bila arah sisinya diabaikan. Kemudian, akan ditunjukkan bahwa ada *path* berarah (dan karena itu ada *path* berarah sederhana) dari R ke setiap titik lain V = R karena derajat masuk V adalah 1, ada titik V_1 = V dan sisi berarah dari V_1 ke V. Jika V_1 = R, pekerjaan telah selesai. Jika tidak, ada titik $V_2 = V_1$, dan sisi berarah dari V_2 ke V_1, karena derajat masuk V_1 adalah 1. Dan karena tidak ada sikel berarah, maka V_2 = V. Jika V_2 = R, maka pekerjaan selesai. Jika tidak demikian, maka proses ini dapat diulangi, dengan setiap *iterasi* (pengulangan) menghasilkan titik baru. Karena banyak titiknya hingga, akhirnya pasti R dicapai. Jadi, telah dibuat *path* berarah dari R ke V.

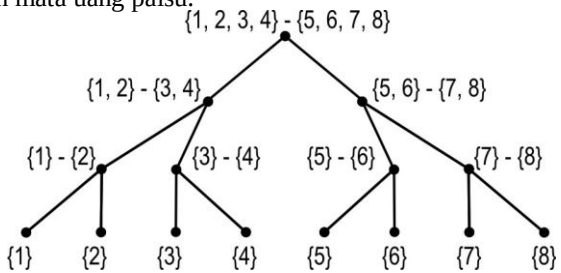
Tunggalnya *path* berarah dari R ke V diperoleh seperti pada bagian (a) dan (b).

Sekarang akan diperlihatkan contoh yang menggunakan pohon berakar untuk mendapatkan penyelesaian suatu masalah.

Contoh 9

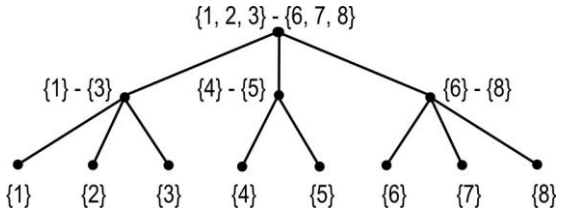
Andaikan ada tujuh mata uang yang identik. Mata uang yang kedelapan kelihatan sama, tetapi sebenarnya lebih berat. Dengan menggunakan neraca diupayakan menemukan mata uang yang berbeda itu dengan penimbangan yang sesedikit mungkin dilakukan. Mata uang itu diberi label 1, 2, 3, ..., 8. Catat bahwa bila mata uang itu diletakkan pada dua lengan neraca maka salah satu lengan itu turun atau kedua lengan itu seimbang (terjadi salah satu). Dapat di konstruksi sebuah pohon berakar seperti pada Gambar 4.11 yang memberikan pendekatan sistematis untuk melakukan penimbangan.

Label yang diberikan disamping setiap titik menunjukkan mata uang mana yang ditimbang pada setiap lengan timbangan. Contoh: {1,2} - {3,4} berarti bahwa mata uang 1 dan 2 ditimbang pada lengan kiri dan mata uang 3 serta 4 ditimbang di lengan kanan. Jika lengan kanan turun, maka dilanjutkan dengan anak di sebelah kanan untuk penimbangan berikutnya. Dilakukan hal yang sama jika lengan kiri turun. Contoh: dimulai dengan membandingkan mata uang 1, 2, 3, dan 4 di lengan kiri dan mata uang 5, 6, 7, dan 8 di lengan kanan. Jika lengan kiri turun, maka dibandingkan lagi mata uang 1 dan 2 terhadap mata uang 3 dan 4. Jika pada penimbangan ini lengan kanan turun, maka berikutnya dibandingkan mata uang 3 dan 4. Jika dalam penimbangan ini lengan kanan turun lagi, maka dicapai titik terminal yang menunjukkan bahwa titik 4 adalah mata uang yang palsu. Karena setiap titik terminal berada di akhir path berarah sederhana yang panjangnya 3 dari akar, terlihat bahwa skema ini membutuhkan adanya tiga penimbangan untuk mendapatkan mata uang palsu.



Gambar 4.11

Apakah ada pendekatan lain untuk dapat menemukan mata uang palsu dengan banyak penimbangan yang lebih sedikit? Karena dengan yang setimbang memiliki 3 hasil yang mungkin, dapat dibuat pohon berakar yang memiliki tiga anak dan bukan dua anak seperti yang dikerjakan di atas. Gambar 4.12. menunjukkan kemungkinan itu. Kita berjalan menuju ke anak yang di tengah bila dua lengan setimbang. Karena setiap titik terminal berada di ujung path berarah sederhana yang panjangnya dua dari akar, maka uang palsu dapat ditemukan dengan hanya melakukan dua kali penimbangan.



Gambar 4.12

Pohon pada Gambar 4.11 dan 4.12 disebut pohon keputusan, disebabkan karena cara pohon itu menstrukturkan proses pembuatan keputusan.

DEFINISI 5

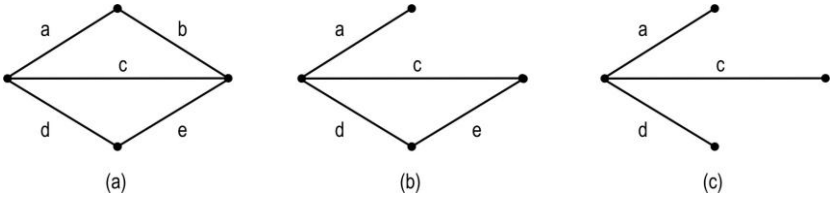
Pohon jumlah graph G adalah pohon (yang dibentuk dengan menggunakan sisi dan titik graph G) yang memuat semua titik graph G.

Jika Graph G merupakan pohon, maka pohon jumlah satu-satunya adalah G sendiri. Suatu graph dapat memiliki lebih dari satu pohon jumlah. Ada beberapa cara untuk memperoleh pohon jumlah suatu graph. Salah satunya dengan menghapuskan sebuah sisi dari setiap sikel. Metode ini digunakan untuk menentukan sistem fundamental sirkuit dalam jaringan kerja listrik. Proses ini diilustrasikan dalam contoh berikut.

Contoh 10

Graph pada Gambar 4.13 (a) bukan pohon karena graph itu memuat sikel a, b, c. Prosedur untuk memperoleh pohon adalah dengan menghapus sebuah sisi di setiap sikelnya. Dengan menghapus b dari sikel a, b, e, d akan diperoleh graph pada Gambar 4.13 (b) yang tetap bukan pohon, karena masih ada sikel c, e, d. Sehingga dihapus lagi sebuah sisi pada sikel ini, misal e.

Graph hasilnya terlihat pada Gambar 4.13 (c) dan sekarang merupakan pohon. Pohon ini adalah pohon jumlah untuk graph aslinya.



Gambar 4.13

Jika suatu graph terhubung memiliki n titik dan e sisi, dengan e $\geq$ n, maka harus dilakukan proses penghapus e - n + 1 kali dengan tujuan mendapatkan pohon. Karena dengan melakukan penghapusan ini, banyak sisi yang semula e berubah menjadi n - 1, yang merupakan banyak sisi dalam pohon dengan n titik.

Metode yang digambar di atas bukan satu-satunya cara untuk mendapatkan pohon jumlah. Ada banyak cara lainnya, dan beberapa di antaranya lebih mudah untuk diprogram pada komputer, karena tidak diperlukan adanya sikel. Salah satu metode ini adalah *Algoritma Pencari Pertama Lebar*.

Algoritma Pohon Jumlah Pencarian Pertama Lebar

Algoritma ini akan mendapatkan pohon jumlah, jika ada, untuk graph G yang bertitik n. Di dalam algoritma ini, L adalah himpunan titik berlabel dan T adalah himpunan sisi yang menghubungkan titik-titik di L.

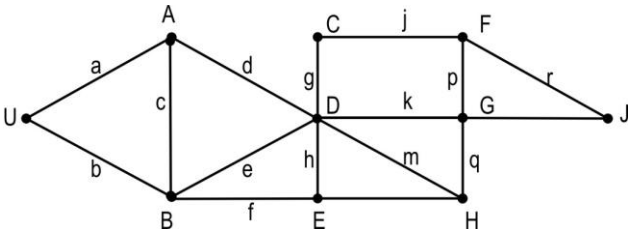
Langkah-langkahnya sebagai berikut.

1. (mulai pada sebuah titik). Ambil titik U dan berikan pada U label 0. Misalkan L = {U}, T = ϕ, dan k = 0.
2. (L memiliki n titik). Jika L memuat semua titik G, berhentilah; sisi-sisi di T dan titik-titik di L membentuk pohon jumlah untuk G.
3. (L memiliki titik kurang dari n). Jika L tidak memuat semua titik G, tentukan titik yang tidak berada di L yang berdekatan dengan titik di L yang label terbesarnya k. Jika tidak ada titik seperti itu, G tidak memiliki pohon jumlah. Bila tidak demikian, berikan pada titik yang baru itu label k + 1, dan letakkan titik itu di L untuk setiap titik baru berlabel k + 1,

letakkan di T satu sisi yang menghubungkan titik itu ke titik berlabel k. Jika ada lebih dari satu sisi seperti itu, pilihlah satu sisi sembarang. Kembalilah ke langkah 2.

Contoh 11

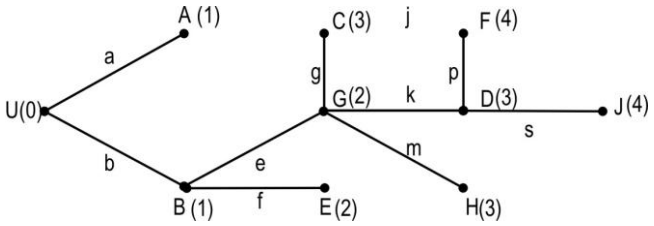
Akan diaplikasikan algoritma pohon jumlah pencarian pertama lebar untuk mendapatkan pohon jumlah bagi graph pada Gambar 4.14. Untuk mudahnya titik dan sisa graph itu diberi nama. Satu titik telah dipilih untuk menjadi awal U.



Gambar 4.14

Mula-mula U diberi label 0, dan himpunan L = {U} serta T = ϕ. Titik belum berlabel yang berdekatan dengan titik berlabel 0 adalah titik A dan B. Pada A dan B diberi label 1, dan L menjadi {U, A, B}. Lebih lagi, sisi a dan b ditambahkan pada T sehingga T = {a, b}. Karena L tidak memuat semua titik graph, dicari titik yang belum berlabel yang berdekatan dengan titik berlabel 1. titik-titik itu adalah D dan E, yang kemudian diberi label 2. Sekarang L = {U, A, B, D, E}. Walaupun ada tiga sisi antara titik berlabel 1 dan titik berlabel 2, akan dipilih hanya 2 dari titik itu, yang satu insiden dengan D dan yang lainnya insiden dengan E. Pilih e dan f sembarang, sehingga T= {a, b, e, f). Karena L tidak memuat semua titik graph itu, tentukan titik-titik yang belum berlabel yang berdekatan dengan titik berlabel 2. Titik-titik ini adalah C, G, dan H, serta pada titik tersebut diberi label 3 dan dimasukkan di L. Sehingga L menjadi {U, A, B, C, D, E, G, H}. Karena 3 titik ditambahkan pada L, maka perlu dipilih tiga sisi untuk ditambahkan pada T. Andaikan dipilih g, k, dan m, sehingga T = {a, b, e, f, g, k, m}. L tidak memuat semua titik graph, sehingga dilihat titik-titik yang belum berlabel yang berdekatan dengan titik yang berlabel 3. Titik itu adalah F dan J, yang kemudian diberi label 4. Sekarang L adalah {U, A, B, C, D, E, F, G, H, J}. Sisi p dan s juga ditambahkan pada T sehingga T = {a, b, e, f, g, k, m, p, s}. Karena memuat semua titik graph. T adalah pohon jumlah yang

diilustrasikan pada Gambar 4.15. Label titik-titiknya diletakkan dalam kurung.

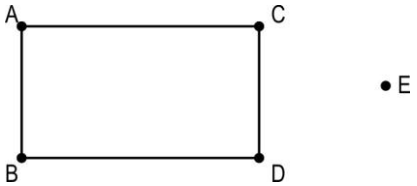


Gambar 4.15

Perlu dicatat bahwa ketika menggunakan algoritma ini, ada tempat-tempat yang sisinya dipilih sembarang. Pilihan berbeda menghasilkan pohon jumlah yang berbeda.

Contoh 12

Graph G yang diberikan pada Gambar 4.16 tidak memiliki pohon jumlah, karena tidak ada sisi-sisi yang menghubungkan semua titik G. Sehingga tidak ada path dari A ke E.



Gambar 4.16

Pada contoh-contoh di atas, terlihat bahwa keberadaan pohon jumlah berkaitan dengan keterhubungan graph itu. Teorema berikut menyatakan hubungan itu secara eksplisit.

Teorema 8

Graph G adalah terhubung jika dan hanya jika G memiliki pohon jumlah.

Bukti

Misalkan graph G memiliki pohon jumlah T. Karena T adalah graph terhubung yang memuat semua titik G, maka untuk setiap titik U dan V di G

ada *path* yang menggunakan sisi-sisi T. Tetapi karena sisi-sisi T adalah juga sisi-sisi G, maka ada *path* antara U dan V yang menggunakan sisi-sisi G. Karena itu G terhubung. Sebaliknya, andaikan G adalah terhubung. Dengan mengaplikasikan algoritma pohon jumlah pencarian pertama lebar pada G diperoleh himpunan L yang memiliki elemen titik-titik berlabel dan himpunan T yang memiliki sisi-sisi yang menghubungkan titik-titik di L. Lebih lagi, T adalah pohon. Karena G terhubung, maka setiap titik G berlabel. Jadi, L memuat semua titik G dan T merupakan pohon jumlah untuk G.

Algoritma lain untuk memperoleh pohon jumlah adalah *algoritma pencari pertama dalam*.

Algoritma Pohon Jumlah Pencarian Pertama Dalam

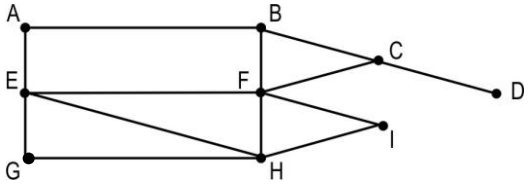
Algoritma ini akan memberikan label titik-titik dan memilih sisi-sisi graph. Pada algoritma ini, L adalah himpunan titik berlabel dan T adalah himpunan sisi yang dipilih.

Langkah-langkahnya adalah sebagai berikut.

1. (mulai berdekatan). Ambil titik U dan berikan pada U label 1. Misalkan L = {U}, T = 0, dan k=1.
2. (titik berdekatan). Jika titik tertentu di L berdekatan dengan titik tidak di L, lanjutkan ke langkah 3. Jika tidak demikian, berhenti.
3. (tentukan titik berikutnya). Andaikan V adalah titik dengan label terbesar yang berdekatan dengan titik W yang tidak di L. Tambahkan 1 pada k, beri W label k, masukkan pada V dan W di T, dan lanjutkan ke langkah 2.

Contoh 13

Langkah pertama dalam mengaplikasikan pencari pertama dalam pada graph. Gambar 4.17 adalah memilih titik awal secara sembarang, katakan A, maka A diberi label 1 dan letakkan di dalam L.

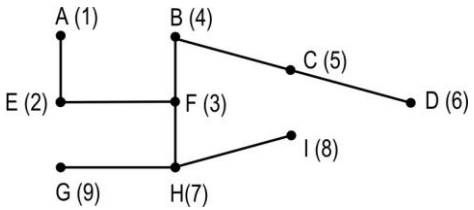


Gambar 4.17

Karena A ada di L dan titik berdekatan B serta E tidak di L, langkah 3 segera dijalankan. Sekarang dipilih titik tidak berlabel yang berdekatan dengan A, katakan E, dan beri E label 2. Titik E dimasukkan di L dan sisinya p ada A serta E ditempatkan di T. Karena ada titik di L dengan titik berdekatan tidak di L, kita berjalan maju ke langkah 3.

Kali ini, E adalah titik berlabel terbesar yang memiliki titik berdekatan yang tidak di L, pilih satu dari titik berdekatan yang belum berlabel ini, katakan F. Berikan pada F label 3, tempatkan F di L, dan masukkan sisi pada E dan F di T. Langkah 3 perlu diulangi lagi. Di sini F adalah titik berlabel terbesar yang memiliki titik berdekatan yang tidak di L, sehingga dipilih satu dari titik berdekatan ini, katakan B. Sekarang B diberi label 4, dan sisi pada B dan F ditempatkan di T. Langkah 3 diulangi dengan melihat pada titik yang berdekatan dengan B yang belum berlabel. Karena titik yang seperti itu hanya C, maka diberikan label 5 pada C dan menempatkan sisi B dan C di T. Demikian pula pada D ditempatkan di T.

Masih ada titik-titik berlabel yang memiliki titik berdekatan tanpa label. Kali ini bukan titik D yang berlabel terbesar 6 dan memiliki titik berdekatan semacam itu. Jadi, kita kembali ke titik berlabel 5 dan mendapatkan bahwa titik itu juga tidak memiliki titik berdekatan tanpa label. Pengujian titik berlabel 4 menunjukkan hal yang sama. Titik F yang berlabel 3 memiliki titik-titik berdekatan tidak berlabel dan dipilih satu, katakan H. Kemudian titik H diberi label 7 dan sisi FH ditempatkan di T. Langkah 3 sekarang diulangi, dengan G atau I, salah satu mendapatkan label 8. Andaikan dipilih I, maka sisi pada H dan I dimasukkan di T, karena I tidak memiliki titik berdekatan tanpa label, kita kembali ke H, yang berdekatan dengan titik yang tidak berlabel G. Jadi G diberi label 9, dan sisi pada G dan H ditempatkan di T. Gambar 4.18 mengilustrasikan pelabelan dan penempatan sisi di T itu secara lengkap.



Gambar 4.18

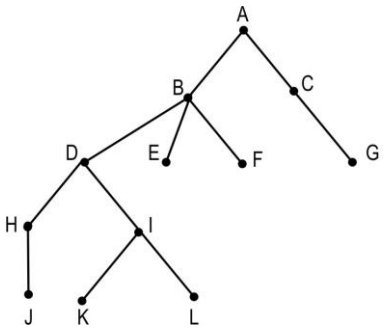
Terdapat perbedaan mendasar antara pencari pertama lebar dan pencari pertama dalam. Dengan pencari pertama lebar, dari setiap titik kita berjalan keluar ke semua titik berdekatannya, dan proses ini diulangi di setiap titik. Kita tidak perlu berjalan kembali untuk melanjutkan pencarian. Tetapi dalam pencari pertama dalam, kita berjalan keluar dari sebuah titik sejauh yang kita dapat, dan bila tidak dapat melanjutkan, kita kembali ke titik yang baru dipilih dan memilih pilihan titik berikutnya, kemudian kita berjalan lagi sejauh yang dimungkinkan.

Silakan istirahat sejenak!

LATIHAN

Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

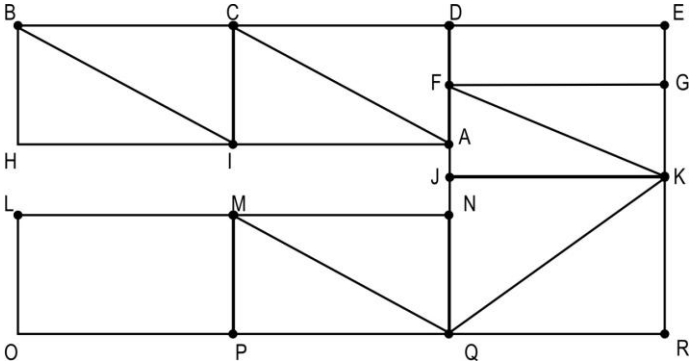
1) Gambar graph yang bukan pohon sedemikian hingga banyak titiknya lebih satu dari banyak sisinya!
2) Perhatikan pohon berakar berikut.



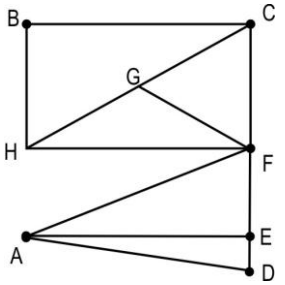
Tuliskan :

a. Akarnya
b. Titik internalnya
c. Titik terminalnya
d. Orang tua G
e. Anak B
f. Keturunan D
g. Nenek moyang H

3) Tentukan pohon jumlah dari graph berikut dengan menggunakan algoritma pohon jumlah pencarian pertama lebar (mulai dengan A dan gunakan urutan abjad bila ada pilihan untuk sebuah titik)!

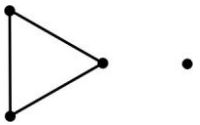


4) Tentukan pohon jumlah dari graph berikut dengan menggunakan algoritma pohon jumlah pencarian pertama dalam!



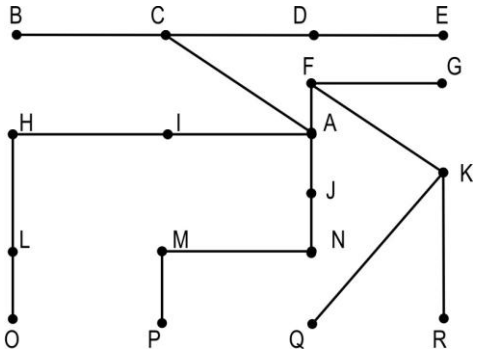
Periksa dan teliti kembali jawaban Anda, sekarang cocokkan jawaban dengan kunci jawaban berikut ini.

Petunjuk Jawaban Latihan

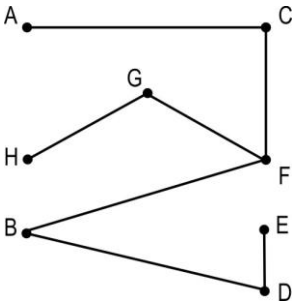
1) 

2) a. A
   b. A, B, C, D, H, I
   c. J, K, L, E, F, G
   d. C
   e. D, E, F
   f. H, I, J, K, L
   g. A, B, D

3)



4)



RANGKUMAN

Pohon ialah graph terhubung yang tidak memiliki sikel. Hutan adalah graph tanpa sikel.

Sebuah pohon pada graph G yang memuat semua titik G disebut pohon rentang dari G.

Pohon berakar adalah graph berarah (*digraph*) T yang memenuhi dua syarat:

(1) Bila arah sisi-sisi pada T diabaikan, hasil graph tidak berarahnya merupakan sebuah pohon, dan

(2) Ada titik tunggal R sedemikian hingga derajat masuk R adalah 0 dan derajat masuk sembarang titik lainnya adalah 1.

Pohon jumlah graph G adalah pohon (yang dibentuk dengan menggunakan sisi dan titik graph G) yang memuat semua titik graph G untuk menentukan pohon jumlah dari graph G dapat dilakukan dengan

menggunakan algoritma pohon jumlah pencarian pertama lebar atau algoritma pohon jumlah pencarian pertama dalam.

TES FORMATIF 1

Pilihlah satu jawaban yang paling tepat!

1) Sebuah pohon mempunyai dua titik berderajat 2, sebuah titik berderajat 3, dan tiga titik berderajat 4. Banyaknya titik berderajat satu adalah ....
   A. 7
   B. 8
   C. 9
   D. 10

2) Suatu kompetisi sepak bola melibatkan 16 kesebelasan. Apabila kompetisi tersebut dilakukan dengan cara sistem gugur (kesebelasan yang kalah tidak bertanding lagi), berapa banyak pertandingan untuk menentukan juara pertama?
   A. 12 kali
   B. 15 kali
   C. 16 kali
   D. 18 kali

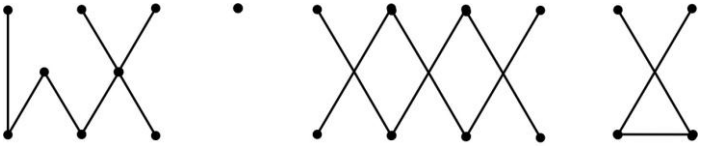
3) Bobot salah satu di antara delapan mata uang yang bentuknya identik lebih ringan daripada mata uang lainnya. Dengan menggunakan alat pengukur yang hanya dapat membedakan apakah dua benda sama beratnya, berapa kali minimum pengukuran bobot harus dilakukan untuk mengidentifikasi mata uang yang bobotnya teringan?
   A. 6 kali
   B. 5 kali
   C. 4 kali
   D. 3 kali

4) Banyaknya *path* (lintasan) sederhana yang panjangnya bukan nol pada pohon yang memiliki n titik dengan $n \geq 2$ adalah ....
   A. (n(n-1))/2
   B. (n(n-1))/3
   C. n(n-1)
   D. 2n(n-1)

5) Pernyataan berikut ini yang benar adalah ....
   A. jika T pohon, maka untuk setiap dua titik u dan v yang berbeda di T terdapat lebih dari satu lintasan (*path*) yang menghubungkan kedua titik tersebut.
   B. jika T pohon, maka $|V(T)| = |E(T)| - 1$
   C. pohon adalah suatu graph yang tidak memiliki sikel
   D. jika e sebuah sisi pada graph G maka $\tau(G) = \tau(G) = \tau(G\text{-}e) + \tau(G.e)$

6) Graph lengkap dengan 4 titik mempunyai ....
   A. 8 pohon rentang
   B. 16 pohon rentang
   C. 20 pohon rentang
   D. 24 pohon rentang

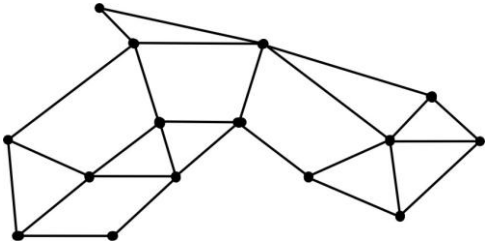
7) Perhatikan graph pada gambar di bawah ini.



   Yang *bukan* merupakan pohon adalah graph pada gambar ....
   A. (i)
   B. (ii)
   C. (iii)
   D. (iv)

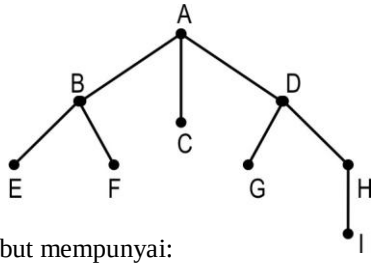
8) Banyaknya sisi pada pohon dengan 21 titik adalah ....
   A. 20 buah
   B. 21 buah
   C. 22 buah
   D. tidak tentu

9) Seorang petani ingin mengairi sawahnya yang tanamannya mulai tumbuh. (Berikut ini adalah peta sawah itu. Daerah yang dibatasi sisi menunjukkan sawah dan sisi graph itu menunjukkan pematang sawah yang memisahkan petak sawah yang satu dengan yang lain). Pengairan dilakukan dengan melubangi pematang dan membiarkan air dari luar menutupi semua sawah. Berapa banyak lubang (sesedikit mungkin) yang harus dibuatnya?



A. 18
B. 16
C. 14
D. 12

10) Perhatikan pohon berakar berikut ini



Pohon akar tersebut mempunyai:
(i) akarnya A
(ii) titik internalnya A, B, D, H
(iii) titik terminalnya E, F, C, G, I
Pertanyaan yang benar adalah ....
A. hanya (i) dan (ii) benar
B. hanya (i) dan (iii) benar
C. hanya (ii) dan (iii) benar
D. (i), (ii), dan (iii) benar

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 1 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 1.

Rumus:

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

Arti tingkat penguasaan yang Anda capai:
90 - 100% = baik sekali
80 - 89% = baik
70 - 79% = cukup
< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan Kegiatan Belajar 2. **Bagus!** Tetapi bila tingkat penguasaan Anda masih di bawah 80%, Anda harus mengulangi Kegiatan Belajar 1, terutama bagian yang belum Anda kuasai.

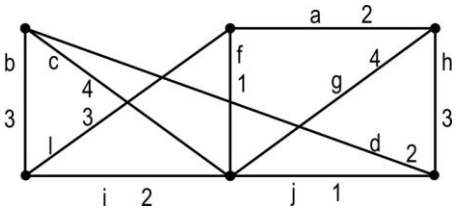
Kegiatan Belajar 2

Pohon Jumlah Minimal dan Pohon Biner

A. POHON JUMLAH MINIMAL

Ketika dibangun saluran yang menghubungkan tempat-tempat penyimpanan minyak, biaya pembangunan setiap saluran itu tidak sama besarnya. Karena kemiringan tanah, jarak, dan faktor-faktor lainnya maka biaya pembangunan salah satu saluran itu dapat lebih tinggi daripada saluran lainnya.

Masalah ini dapat digambarkan sebagai graph berbobot, dengan bobot setiap sisinya menunjukkan biaya pembangunan saluran itu (lihat Gambar 4.19). Masalahnya adalah bagaimana dapat membangun jaringan-jaringan pipa yang biayanya serendah mungkin. Dengan kata lain, ingin ditemukan pohon jumlah yang total biaya sisinya sekecil mungkin.



Gambar 4.19

Di dalam graph berbobot, bobot sebuah pohon adalah jumlah bobot sisi-sisi pohon itu. *Pohon jumlah minimal* di dalam sebuah graph berbobot adalah pohon jumlah yang bobotnya sekecil mungkin. Dengan kata lain, pohon jumlah minimal adalah pohon jumlah sedemikian hingga tidak ada pohon jumlah lain yang memiliki bobot kurang darinya.

Contoh 1

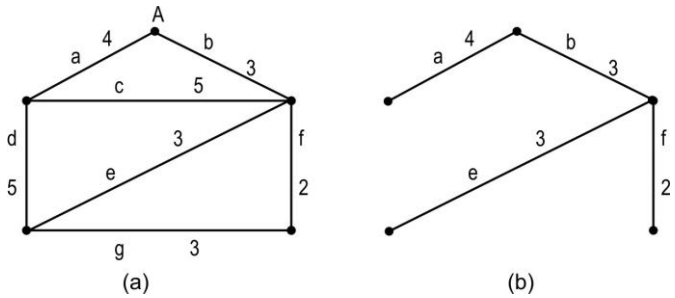
Graph berbobot pada Gambar 4.19, sisi-sisi b, c, e, g, dan h membentuk pohon jumlah dengan bobot 3 + 4 + 3 + 4 + 3 = 17. Sisi a, b, c, d, dan e membentuk pohon jumlah yang lain dengan bobot 2 + 3 + 4 + 2 + 3 = 14. Sisi-sisi a, d, f, i, dan j membentuk pohon jumlah yang lain lagi, dengan

bobot 8. Karena pohon jumlah ini menggunakan kelima sisi dengan bobot terkecil, maka tidak ada pohon jumlah lain dengan bobot yang lebih kecil. Jadi, sisi a, d, f, i, dan j membentuk pohon jumlah minimal untuk graph berbobot itu.

Pada contoh 1 di atas, dapat ditemukan pohon jumlah minimal dengan cara mencoba-coba. Untuk graph berbobot dengan titik dan sisi yang banyak sekali, pendekatan ini tidak praktis. Salah satu pendekatan sistematis yang dapat dilakukan adalah dengan menemukan semua pohon jumlah graph berbobot terhubung, dan menghitung bobotnya, serta kemudian memilih pohon jumlah dengan bobot terkecil. Pemeriksaan semua kemungkinan itu sangat memakan waktu, meskipun dikerjakan dengan komputer. Biasanya pohon jumlah minimal ini dicoba disusun dengan membangun pohon jumlah menggunakan sisi berbobot terkecil. Pendekatan ini diilustrasikan pada Contoh 2.

Contoh 2

Untuk graph berbobot pada Gambar 4.20 (a), kita mulai pada titik A dan memilih sisi yang berbobot terkecil pada A, yaitu b. Untuk melanjutkan penyusunan pohon, dilihat sisi a, c, e, dan f yang menyinggung sisi b. Dipilih satu sisi dengan bobot terkecil, yaitu f. Sisi berikutnya yang perlu dilihat adalah a, c, e, dan g yang menyinggung sisi yang telah dipilih. Ada dua sisi dengan bobot terkecil, yaitu e dan g, kemudian dipilih satu secara sembarang, katakan e. Sisi berikut yang diperhatikan adalah a, c, dan d (sisi g tidak diperhatikan lagi, karena apabila diambil sisi g akan membentuk sikel dengan e dan f). Sisi dengan bobot terkecil adalah a, dan dengan demikian a ditambahkan ke pohon jumlah. Keempat sisi a, b, e, f ini membentuk pohon jumlah, yang juga merupakan pohon jumlah minimal (lihat Gambar 4.20 b).



Gambar 4.20

Metode yang digunakan pada contoh 2 akan selalu menghasilkan pohon jumlah minimal. Metode ini ditemukan oleh Prim. Algoritma Prim menyusun pohon dengan memilih sembarang titik dan kemudian suatu sisi dengan bobot terkecil pada titik itu. Berikutnya, pohon itu dikembangkan dengan memilih suatu sisi berbobot terkecil. Pohon itu masih dikembangkan lagi dengan memilih suatu sisi berbobot terkecil yang membentuk pohon dengan dua sisi terpilih sebelumnya. Proses ini dilanjutkan sampai diperoleh sebuah pohon jumlah, yang juga merupakan pohon jumlah minimal. Prosedur ini dapat diformalisasikan sebagai berikut.

1. Algoritma Prim untuk Pohon Jumlah Minimal

Dengan algoritma ini akan ditemukan pohon jumlah minimal (jika ada) untuk sebuah graph berbobot G dengan n titik. Dalam algoritma ini S adalah himpunan titik dan T adalah himpunan sisi.

Langkah 1 (mulai). Pilih titik U, dan misalkan S = {U} serta T = { }.

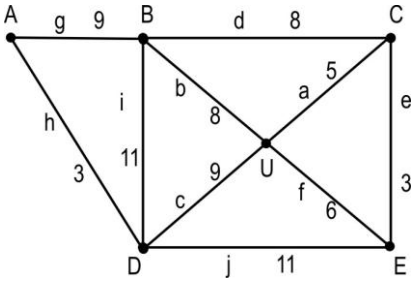
Langkah 2 (pemeriksaan untuk menyelesaikan). Jika S memuat semua titik G, kemudian berhenti; sisi di T dan titik di S membentuk pohon jumlah minimal untuk G.

Langkah 3 (pilih sisi berikutnya). Jika S tidak memuat semua titik G, tentukan sisi-sisi yang memiliki satu titik di S dan titik lainnya tidak di S. Jika tidak ada sisi seperti itu, G tidak terhubung dan tidak memiliki pohon jumlah minimal. Jika tidak demikian, pilih satu sisi seperti itu yang berbobot terkecil (rangkaian dapat diputuskan secara sembarang), dan tempatkan di T serta titiknya di S (salah satunya sudah di S).
Kembalilah ke langkah 2.

Pada langkah 3 algoritma Prim pemilihan suatu sisi dengan satu titik di S dan titik lain tidak di S menjamin tidak adanya sikel yang terbentuk dengan pengumpulan sisi-sisi di T, jadi pada akhir setiap iterasi langkah 3, sisi di T dan titik di S membentuk pohon.

Contoh 3

Algoritma Prim akan diaplikasikan pada graph berbobot G pada Gambar 4.21. Mulai di titik U, sehingga S = {U} dan T = { }.

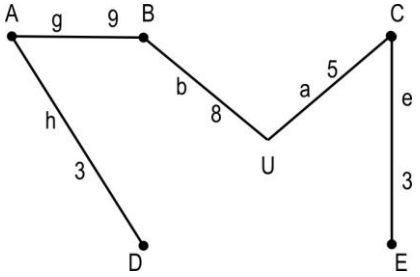


Gambar 4.21

Karena S tidak memuat semua titik G, diperhatikan sisi pada U, yaitu a, b, c, dan f, dan ambil satu yang berbobot terkecil. Sisi itu adalah a, sehingga a ditempatkan di T dan titiknya ditambahkan di S. Jadi S = {U, C} dan T = {a}. Terlihat S tidak memuat semua titik G, sehingga di lihat sisi-sisi yang memiliki satu titik di S dan satu titik tidak di S. Sisi-sisi ini adalah b, c, d, e, dan f.

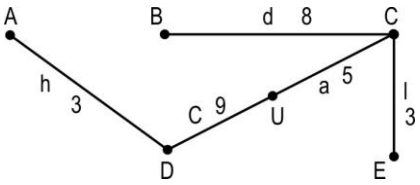
Di antara sisi-sisi ini, e memiliki bobot terkecil, sehingga sisi itu ditambahkan di himpunan T dan titik E ditambahkan pada himpunan S. Sekarang S = {U, C, E} dan T = {a, e}. Karena S belum memuat semua titik di G, diperhatikan lagi sisi-sisi dengan satu titik di S dan titik lain tidak di S. Sisi-sisi ini adalah b, c, d, dan j. Perhatikan bahwa sisi f tidak diperhatikan, karena dua titiknya ada di S. Dari sisi b, c, d, dan j, ada dua yang memiliki bobot terkecil, yaitu b dan d.

Diambil salah satu secara sembarang, misalkan b dan tambahkan pada T serta B pada S. Jadi, S = {U, C, E, B} dan T = {a, e, b}. Pada saat ini sisi-sisi dengan satu titik di S dan yang lain tidak di S adalah c, g, i, dan j. Dari semua titik ini, sisi c dan g memiliki bobot terkecil. Sehingga dipilih g secara sembarang, membuat S = {U, C, E, B, A} dan T = {a, e, b, g}. Sekarang perhatikan sisi-sisi c, h, i, dan j. Sisi yang memiliki bobot terkecil adalah h, sehingga dipilih sisi h ini. Maka S = {U, C, E, B, A, D} dan T = {a, e, b, g, h}. karena S memuat semua titik dari G, maka pohon jumlah minimum dari graph tersebut terlihat pada Gambar 4.22 dengan bobot 28.



Gambar 4.22

Pada dua tempat berbeda pada contoh di atas, dapat dipilih lebih dari satu titik dengan bobot terkecil. Algoritma itu menunjukkan bahwa setiap titik dengan bobot terkecil dapat saja dipilih. Oleh karena itu dapat disusun pohon jumlah minimal yang berbeda, tergantung pada pilihan yang dilakukan. Misal: jika pada contoh 3 dipilih sisi d dan bukan b, diikuti dengan c dan h, maka pohon jumlah minimalnya adalah seperti pada Gambar 4.23. Jadi, pohon jumlah minimalnya tidak tunggal.



Gambar 4.23

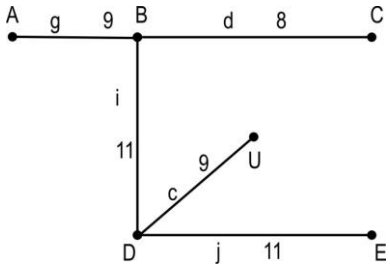
Perhatikan kembali Gambar 4.21. Andaikan sekarang bobot sisi-sisi itu menunjukkan besarnya keuntungan yang dihasilkan bila minyak dipompakan melalui saluran itu. Masalahnya sekarang adalah mendapatkan pohon jumlah saluran yang menghasilkan keuntungan terbesar. Jadi, sekarang diinginkan untuk mendapatkan pohon jumlah yang total bobot sisinya bukan paling kecil, melainkan paling besar.

Pohon jumlah maksimal di dalam graph berbobot adalah pohon jumlah yang bobotnya sebesar mungkin. Dengan kata lain, bobot pohon jumlah itu paling besar, tidak ada pohon jumlah lain yang bobotnya lebih besar. Untungnya, cara mendapatkan pohon jumlah maksimal amat mirip dengan cara menemukan pohon jumlah minimal. Yang diperlukan hanyalah menggantikan kata “terkecil” dengan kata “terbesar” dalam algoritma Prim.

Contoh 4

Untuk graph berbobot pada Gambar 4.21, akan disusun pohon jumlah maksimal. Mulai dengan mengambil titik U. Maka S = {U} dan T = { }. Sekarang dilihat sisi-sisi pada U dan ambil satu bobotnya yang terbesar yaitu sehingga S = {U, D} dan T {c}. Karena S tidak memuat semua titik graph itu, diperhatikan sisi-sisi lain dengan satu titik di S dan yang lain tidak di S. Sisi-sisi ini adalah a, b, f, h, i, dan j. Di antaranya ada dua dengan bobot terbesar, i dan j. Dipilih satu secara sembarang katakan i. Sehingga sekarang T = {c, i} dan S = {U, B, D}.

Proses ini diulang lagi dengan memilih sisi j yang berbobot terbesar dan memilih satu titik di S dan yang lain tidak di S. Sekarang T = {c, i, j} dan S = {U, B, D, E}. Dilihat lagi sisi-sisi dengan satu titik di S dan yang lain tidak di S. Di antaranya, g adalah sisi yang berbobot terbesar, sehingga T = {c, g, i, j} dan S = {U, A, B, D, E}. Pilih sisi d yang berbobot terbesar dan memilih satu titik di S dan yang lain tidak di S, sehingga T = {c, d, g, i, j} dan S = {U, A, B, C, D, E} yang merupakan himpunan semua titik. Oleh karena itu T adalah pohon jumlah maksimal dengan bobot 48 seperti terlihat pada Gambar 4.24.



Gambar 4.24

Algoritma lain yang dapat digunakan untuk mendapatkan pohon jumlah minimal ditemukan oleh Kruskal.

2. Algoritma Kruskal untuk pohon jumlah minimal

Algoritma ini akan mendapatkan pohon jumlah minimal, jika ada, untuk graph berbobot G yang memiliki n titik, dengan $n \geq 2$. Dalam algoritma ini, S adalah himpunan titik dan T adalah himpunan sisi.

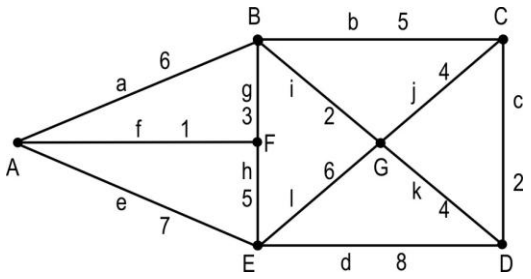
Langkah 1 (mulai). Jika tidak ada sisi, G tidak terhubung, dan karena itu tidak memiliki pohon jumlah minimal. Jika tidak demikian, ambil sebuah sisi dengan bobot terkecil (rangkaian dapat diputuskan secara sembarang). Tempatkan sisi itu di T dan titiknya di S.

Langkah 2 (pemeriksaan untuk penyelesaian). Jika T memuat n - 1 sisi, maka berhentilah; sisi-sisi di T dan titik-titik di S membentuk pohon jumlah minimal. Jika tidak demikian lanjutkan ke langkah 3.

Langkah 3 (ambil sisi berikutnya). Tentukan sisi-sisi berbobot terkecil yang tidak membentuk sikel dengan sembarang sisi yang ada di T. Jika tidak ada sisi seperti itu, G tidak terhubung dan tidak memiliki pohon jumlah minimal. Jika tidak demikian, pilih satu sisi sejenis itu (rangkaian dapat diputus secara sembarang), dan tempatkan sisi itu di T dan titiknya di S. Kembalilah ke langkah 2.

Contoh 5

Gunakan algoritma Kruskal untuk menentukan pohon rentang dengan bobot minimum dari graph berikut.

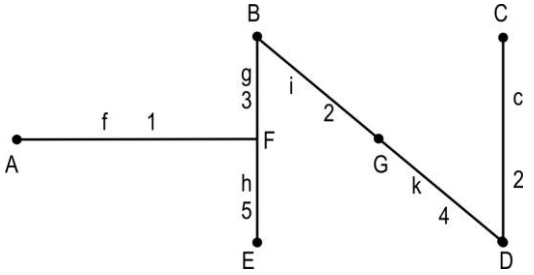


Gambar 4.25

Jawab

Ambil sebuah sisi dengan bobot terkecil, sisi itu adalah f, sehingga T = { } dan S = {A, F}. Tentukan sisi berbobot terkecil yang tidak membentuk sikel dengan sembarang sisi yang ada di T, sisi itu adalah g, sehingga T = {f, g} dan S = {A, B, F}. Pilih sisi berbobot terkecil yang tidak membentuk sikel

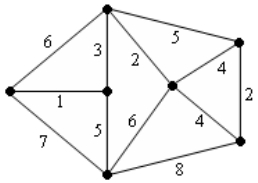
dengan sembarang sisi yang ada di T, sisi itu adalah i, sehingga T = {f, g, i} dan S = {A, B, F, G}. Sisi dengan bobot terkecil yang tidak membentuk sikel dengan sembarang sisi yang ada di T adalah j dan k, misalkan secara sembarang diambil k. Jadi T = {f, g, i, k} dan S = {A, B, D, F, G}. Ambil sisi bobot terkecil yang tidak membentuk sikel dengan sembarang sisi di T, yaitu c. Jadi T = {c, f, g, i, k} dan S = {A, B, C, D, F, G}. Pilih sisi h, sehingga T = {c, f, g, h, i, k} dan S = {A, B, D, E, F, G}. Karena S memuat semua titik dari graph tersebut, maka pohon jumlah minimum dari graph tersebut terlihat pada Gambar 4.26 dengan bobot 17.



Gambar 4.26

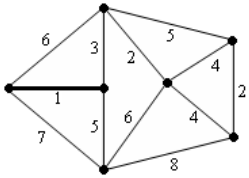
Contoh 6

Gunakan algoritma Kruskal untuk menentukan pohon rentang dengan bobot minimum dari graf G berikut.



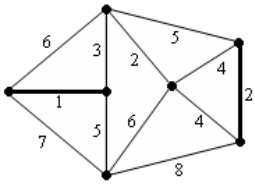
Gambar 4.27

Pertama-tama kita tentukan sisi dengan bobot terkecil. Sisi dengan bobot 1 terpilih sebagai sisi pertama yang kita pilih. Untuk itu kita hitamkan sisinya. Langkah ini memberikan hasil sebagai berikut.



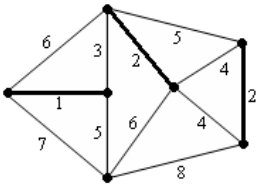
Gambar 4.28

Bobot terkecil berikutnya setelah 1 dan 2, namun terdapat 2 sisi yang mempunyai bobot 2. Misalkan kita pilih sisi yang terlihat vertikal. Pilihan ini memberikan gambar berikut.



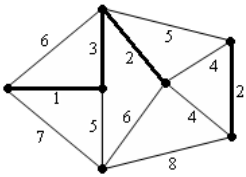
Gambar 4.29

Sisi dengan bobot 2 lainnya sekarang kita masukkan sebagai bagian dari pohon rentang yang sedang dikonstruksi. Ini menghasilkan gambar seperti di bawah ini.



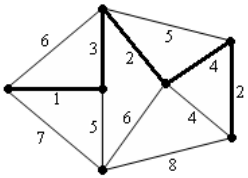
Gambar 4.30

Sisi berikutnya yang kita hitamkan adalah sisi dengan bobot 3



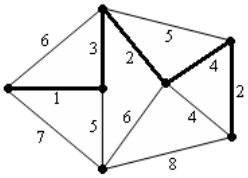
Gambar 4.31

Terdapat 2 sisi dengan bobot 4. Kita pilih sebuah sisi dengan bobot 4, sehingga kita peroleh gambar berikut.



Gambar 4.32

Sekarang hanya tinggal sebuah titik lagi yang belum terambil. Dari 4 sisi yang insiden dengan titik ini, yang mempunyai bobot terkecil adalah sisi dengan bobot 5. Kita pilih sisi ini sehingga pohon optimalnya terlihat pada gambar di bawah ini dengan sisi yang dicetak lebih hitam.



Gambar 4.33

Pohon optimal ini total bobotnya adalah 1 + 3 + 2 + 4 + 2 + 5 = 17.

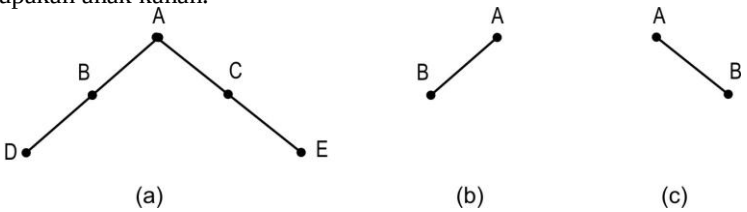
B. POHON BINER

Pohon biner adalah pohon berakar yang setiap titiknya memiliki paling banyak dua anak dan setiap anak ditunjuk sebagai *anak kiri* atau *anak kanan*. Jadi, pada pohon biner setiap titik mungkin memiliki 0, 1, atau 2 anak. Anak kiri digambarkan di sebelah kiri dan di bawah orang tuanya, serta anak kanan di sebelah kanan di bawah orang tuanya.

Contoh 7

Untuk pohon biner pada Gambar 4.34 (a), A adalah akarnya. Titik A memiliki dua anak, anak kiri B dan anak kanan C. Titik B memiliki satu anak, anak kiri D. Demikian pula titik C memiliki anak kanan E, tetapi tidak

memiliki anak kiri. Pada pohon biner Gambar 4.34 (b), titik A memiliki anak kiri B. Berbeda dengan pohon biner pada Gambar 4.34 (c), di sini B merupakan anak kanan.

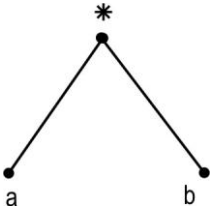


Gambar 4.34

Pohon biner digunakan dalam ilmu komputer untuk mengolah data.

Contoh 8

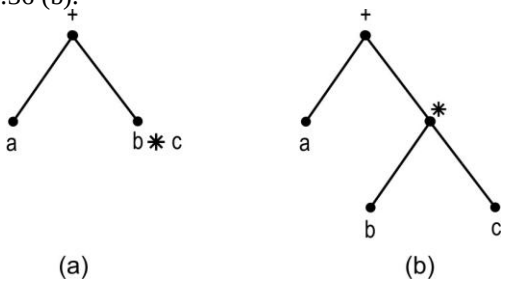
Pernyataan a * b (dengan * menunjukkan perkalian) disajikan dengan pohon biner pada Gambar 4.35.



Gambar 4.35

Contoh 9

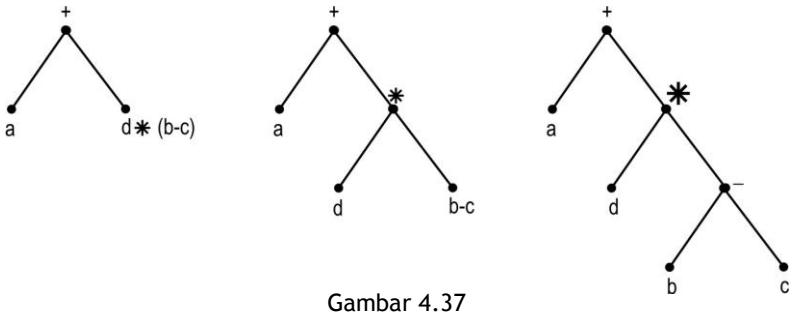
Pernyataan a + b * c berarti a + (b * c). Operasi terakhir yang dijalankan adalah penjumlahan. Mula-mula pernyataan ini disajikan dengan pohon biner pada Gambar 4.36 (a). Proses ini diulangi dan menghasilkan pohon biner pada Gambar 4.36 (b).



Gambar 4.36

Contoh 10

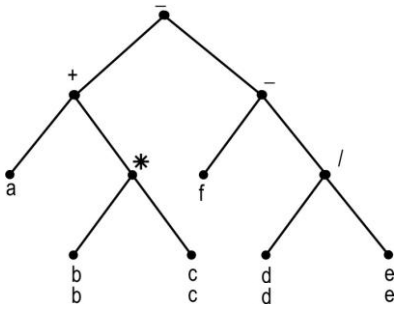
Pohon pernyataan untuk a + d * (b - c) diciptakan dengan barisan pohon biner pada Gambar 4.37.



Gambar 4.37

Contoh 11

Pernyataan (a + b * c) - (f - d/e) disajikan dengan pohon pernyataan pada Gambar 4.38.



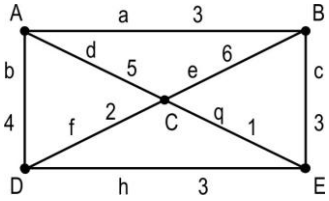
Gambar 4.38

Silakan Anda istirahat sejenak, kemudian kerjakanlah soal-soal latihan berikut.

LATIHAN

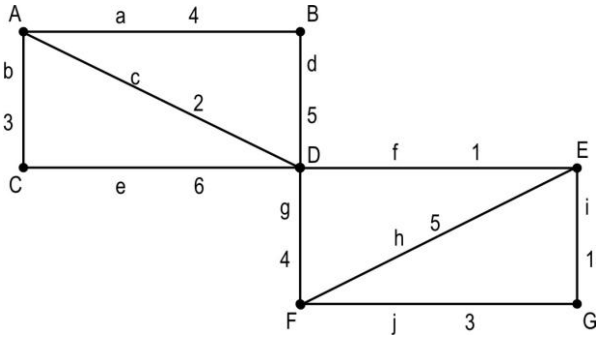
Untuk memperdalam pemahaman Anda mengenai materi di atas, kerjakanlah latihan berikut!

1) Tentukan pohon jumlah minimal dan bobotnya dengan menggunakan algoritma Prim, mulai dari titik A pada graph berikut!



2) Tentukan pohon jumlah minimal dan bobotnya dengan menggunakan algoritma Prim, mulai dari titik C pada graph soal nomor 1!

3) Tentukan pohon jumlah minimal dan bobotnya dengan menggunakan algoritma Kruskal pada graph berikut!



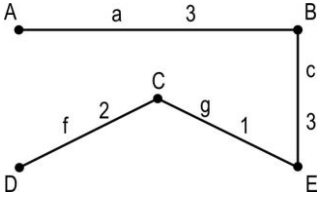
4) Konstruksikan pohon pernyataan untuk operasi ((a - b)/c)* (d + e/f)!

Periksa dan teliti kembali jawaban Anda, sekarang cocokkan jawabannya dengan kunci jawaban berikut ini.

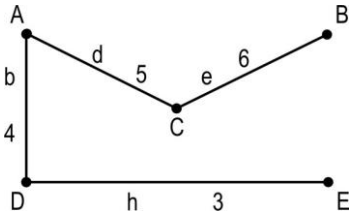
Petunjuk Jawaban Latihan

1) Mulai dari titik A, sehingga T = { } dan S = {A}. Perhatikan sisi a, b, d, ambil bobot terkecil yaitu a, sehingga T = {a} dan S = {A, B}. Perhatikan sisi-sisi yang memiliki satu titik di S dan satu titik tidak di S, yaitu b, c, d, e. Pilih sisi c, karena bobot terkecil, sehingga T = {a, c} dan S = {A, B, F}. Ambil sisi yang memiliki satu titik di S dan satu titik tidak di S dengan bobot terkecil yaitu g, sehingga T = {a, c, g} dan S = {A, B, C, E}. Pilih sisi f, didapat T = {a, c, g, f} dan S = {A, B, C, D, E}. Karena S memuat semua titik dari graph tersebut, maka pohon

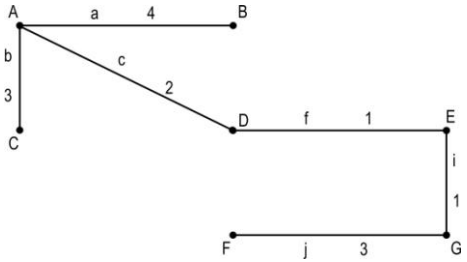
jumlah minimal dari graph tersebut adalah seperti berikut dengan bobot 9.



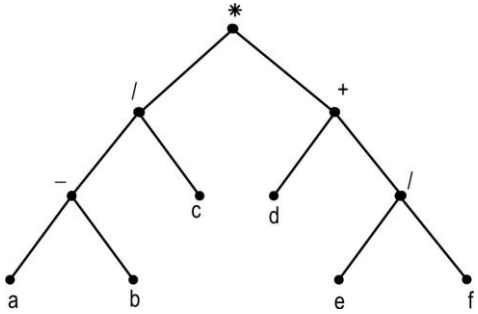
2) Mulai dari titik C, sehingga T = { } dan S = {C}. Perhatikan sisi d, e, f, g. Ambil bobot terbesar, yaitu e, sehingga T = {e} dan S = {B, C}. Perhatikan sisi-sisi yang memiliki satu titik di S dan satu titik tidak di S. Pilih sisi d, karena bobotnya terbesar sehingga T = {d, e} dan S = {A, B, C}. Kemudian pilih sisi b sehingga T = {b, d, e} dan S = {A, B, C, D}. Perhatikan sisi c, g, h. Karena bobot terbesar yaitu c dan h, pilih secara sembarang misalnya sisi h, sehingga T = {b, d, e, h} dan S = {A, B, C, D, E} yang merupakan pohon jumlah maksimal dengan bobot 18, seperti pada gambar berikut.



3) Pilih sebuah sisi dengan bobot terkecil, sisi itu adalah f dan i. Misalkan diambil secara sembarang f, sehingga T = {f} dan S = {D, E}. Tentukan sisi berbobot terkecil yang tidak membentuk sikel dengan sembarang sisi yang ada di T, sisi itu adalah i, sehingga T = {f, i} dan S = {D, E, G}. Pilih sisi berbobot terkecil yaitu c, sehingga T = {c, f, i} dan S = {A, D, E, G}. Pilih sisi dengan bobot terkecil yaitu b atau j, misalkan diambil b. Jadi T = {b, e, f, g, i} dan S = A, C, D, E, G}. Ambil sisi bobot terkecil yang tidak membentuk sikel dengan sembarang sisi di T, yaitu J. Sehingga T = {b, c, f, i, j} dan S = {A, C, D, E, F, G}. Pilih sisi a, sehingga T = {a, b, c, f, i, j} dan S = {A, B, C, D, E, F, G} yang merupakan pohon jumlah minimal dengan bobot 14 seperti terlihat pada gambar berikut.



4)



RANGKUMAN

Di dalam graph berbobot, bobot sebuah pohon adalah jumlah bobot sisi-sisi pohon itu. Pohon jumlah minimal di dalam sebuah graph adalah pohon jumlah yang bobotnya paling kecil. Pohon jumlah maksimal di dalam graph berbobot adalah pohon jumlah yang bobotnya paling besar.

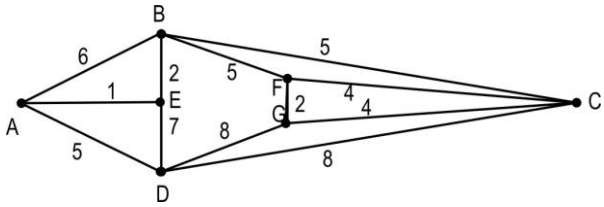
Untuk menentukan pohon jumlah minimal (jika ada) untuk sebuah graph berbobot G dapat menggunakan algoritma Prim atau algoritma Kruskal. Dalam algoritma Prim, langkah awal adalah memilih salah satu titik dalam graph G secara sembarang, sedangkan dalam algoritma Kruskal dimulai dengan memilih sisi dengan bobot paling kecil.

Pohon biner adalah pohon berakar yang setiap titiknya memiliki paling banyak dua anak dan setiap anak ditunjuk sebagai anak kiri atau anak kanan. Jadi, pada pohon biner setiap titik mungkin memiliki 0,1, atau 2 anak.

TES FORMATIF 2

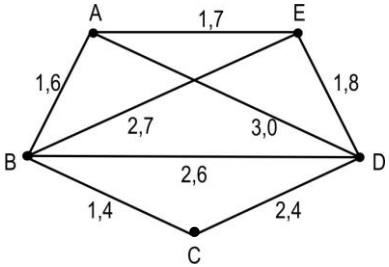
Pilihlah satu jawaban yang paling tepat!

1) Dengan menggunakan algoritma Kruskal, pohon rentang dengan bobot minimum dari graph berikut adalah ....



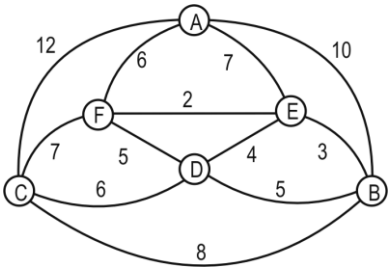
A. 18
B. 19
C. 20
D. 21

2) Lima komputer harus dipasang di sebuah kantor dan satu sama lain saling terhubung (dapat secara langsung atau dengan melalui komputer lain, salah satu). Panjang kabel yang diperlukan untuk menghubungkan unit-unit komputer yang berdekatan diberikan dalam meter. Berapa panjang minimum kabel yang diperlukan?



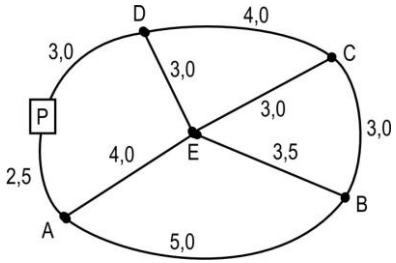
A. 7,3 meter
B. 7,1 meter
C. 6,5 meter
D. 6,1 meter

3) Di sebuah kota terdapat enam tempat yaitu A, B, C, D, E, dan F. Yang perlu dipasang kabel telepon agar terhubung satu dengan yang lain. Berapa panjang minimum kabel yang diperlukan? (jarak dalam kilometer).



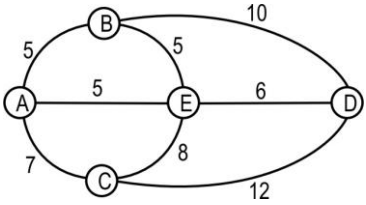
A. 21 kilometer
B. 22 kilometer
C. 23 kilometer
D. 24 kilometer

4) Kantor pusat suatu perusahaan akan membangun jaringan pos elektronik (menggunakan satelit) dengan anak-anak perusahaannya yang terbesar di tempat-tempat seperti pada gambar berikut. Biaya membangun hubungan itu tergantung pada jaraknya dan dinyatakan dalam jutaan rupiah. Berapa biaya maksimum untuk membangun jaringan itu?



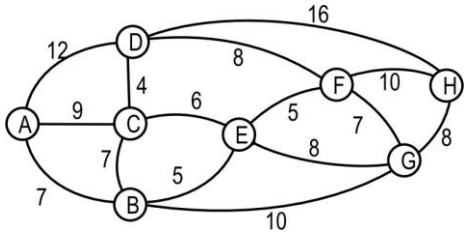
A. 14,5 juta rupiah
B. 16 juta rupiah
C. 17,5 juta rupiah
D. 19 juta rupiah

5) Gambar berikut memberikan informasi tentang panjang kabel yang diperlukan untuk menghubungkan lima lampu taman (panjang dalam meter). Berapa panjang minimum kabel yang diperlukan?



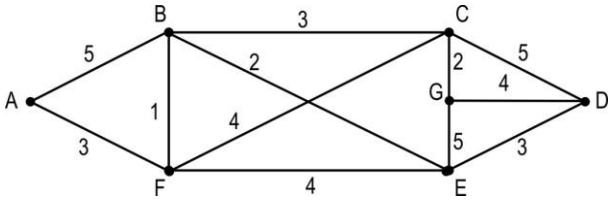
A. 37 meter
B. 24 meter
C. 23 meter
D. 21 meter

6) Tempat-tempat wisata di suatu Taman Nasional saling terhubung dengan jalan-jalan yang jaraknya (dalam kilometer) diperlihatkan pada gambar berikut. Setiap tempat wisata perlu mendapatkan persediaan air. Pipa-pipa air itu dipasang sepanjang sisi jalan-jalan yang telah dibangun. Berapa panjang minimum pipa yang diperlukan?



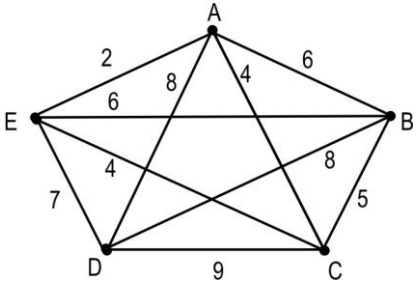
A. 40 kilometer
B. 42 kilometer
C. 44 kilometer
D. 46 kilometer

7) Dengan menggunakan algoritma Prim, pohon rentang dengan bobot maksimum dari graph berikut adalah ....



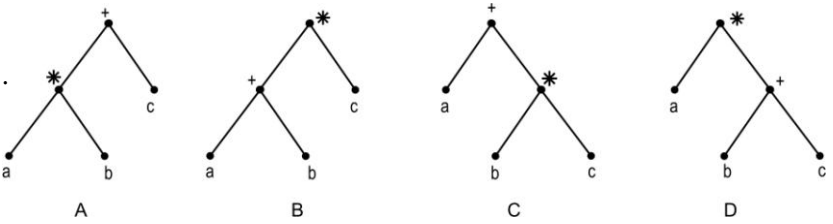
A. 14
B. 20
C. 26
D. 29

8) Pohon rentang minimum dari graph berikut mempunyai bobot ....

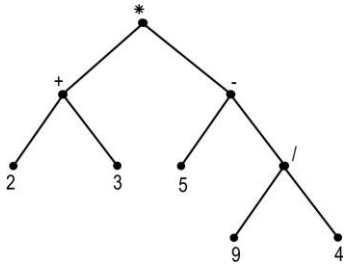


A. 22
B. 20
C. 18
D. 16

9) Konstruksi pohon pernyataan dari operasi a * b + c adalah ....



10) Diagram pohon berikut menunjukkan pohon pernyataan dari operasi ….



A. 2 + 3 * 5 - 9/4
B. 2 + (3 * 5) - (9/4)
C. (2 + 3 * 5) - (9/4)
D. (2 + 3) * (5 - 9/4)

Cocokkanlah jawaban Anda dengan Kunci Jawaban Tes Formatif 2 yang terdapat di bagian akhir modul ini. Hitunglah jawaban yang benar. Kemudian, gunakan rumus berikut untuk mengetahui tingkat penguasaan Anda terhadap materi Kegiatan Belajar 2.

Rumus:

$$\text{Tingkat penguasaan} = \frac{\text{Jumlah Jawaban yang Benar}}{\text{Jumlah Soal}} \times 100\%$$

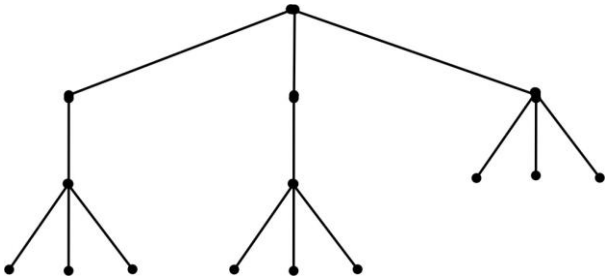
Arti tingkat penguasaan yang Anda capai:
90 - 100% = baik sekali
80 - 89% = baik
70 - 79% = cukup
< 70% = kurang

Apabila mencapai tingkat penguasaan 80% atau lebih, Anda dapat meneruskan dengan modul berikutnya. **Bagus!** Tetapi bila tingkat penguasaan Anda masih di bawah 80%, Anda harus mengulangi Kegiatan Belajar 2, terutama bagian yang belum Anda kuasai.

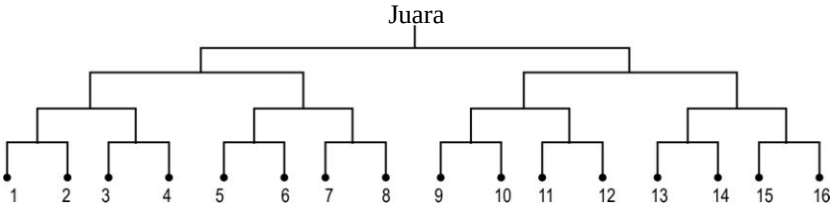
Kunci Jawaban Tes Formatif

Tes Formatif 1

1) C Banyaknya titik berderajat satu adalah 9, seperti terlihat pada gambar pohon berikut.

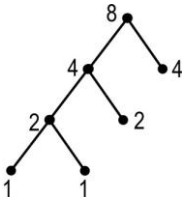


2) B Bagan pertandingannya adalah sebagai berikut.

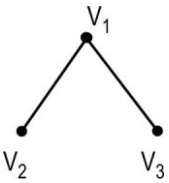


banyaknya pertandingan untuk menentukan juara pertama ada 15 kali pertandingan.

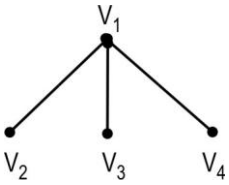
3) D Diagram pohonnya adalah sebagai berikut.



4) A n = 2, ada 1 path: lintasannya V_1 - V_2
n = 3, ada 3 path: lintasannya V_2 - V_1, V_3 - V_1, dan V_2 - V_1 - V_3



n = 4, ada 6 path: lintasannya V_1 - V_2, V_1 - V_3, V_1 - V_4, V_2 - V_1 - V_3,V_2 - V_1 - V_4, dan V_3 - V_1 - V_4



Untuk n titik dengan n $\geq$ 2 ada (n(n-1))/2 path.

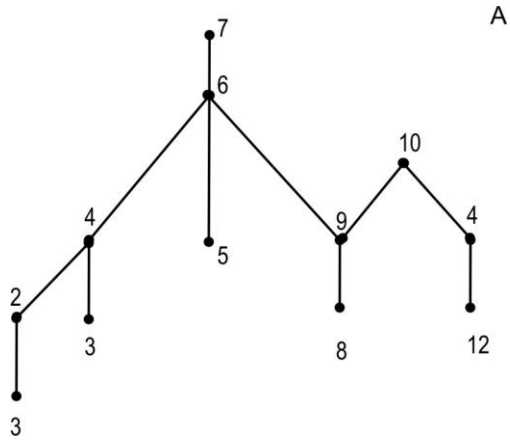
5) D A salah, karena hanya ada satu lintasan yang menghubungkan dua titik sembarang di T.
B salah, karena |V(T)| = |E(T)| + 1
C salah, karena pohon adalah graph terhubung yang tidak memiliki sikel.
D benar (teorema).

6) B $\tau(K_n) = n^{n-2}$
$\tau(K_4) = 4^{4-2} = 16$

7) C Graph pada gambar (iii) bukan merupakan pohon, karena tidak terhubung.

8) A |V(T)| = |E(T)| + 1
21 = |E(T)| + 1
|E(T)| = 20

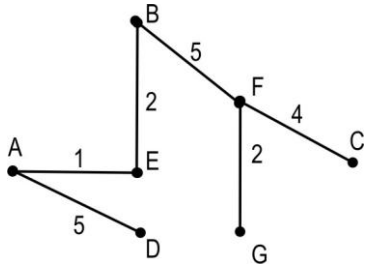
9) D



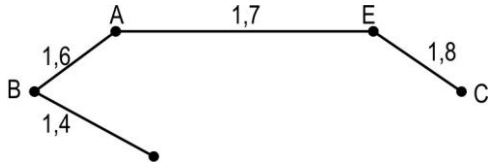
10) D Semuanya (i), (ii), dan (iii) benar.

Tes Formatif 2

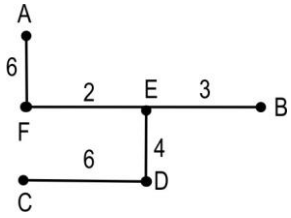
1) B



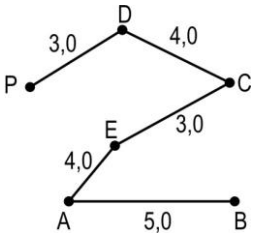
2) C



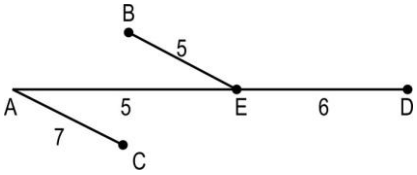
3) A



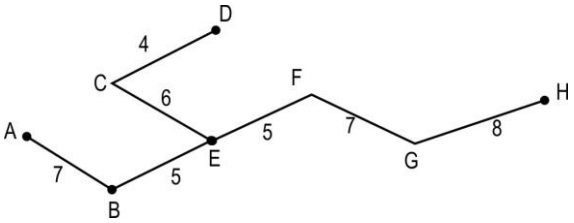
4) D


D
3,0
4,0
P
C
E
3,0
4,0
A
5,0
B

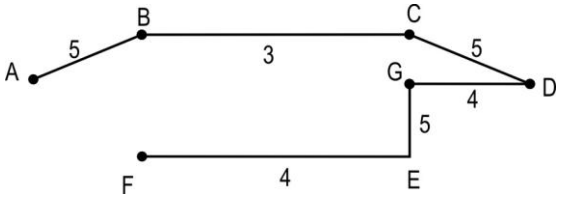
5) C


B
5
A
5
E
6
D
7
C

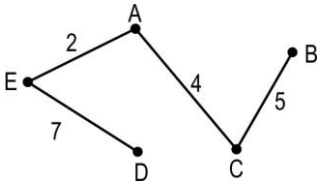
6) B


D
4
C
6
F
H
A
5
7
8
E
7
5
G
B

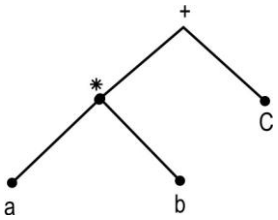
7) C


B
C
5
5
A
3
G
D
4
5
F
4
E

8) C


A
2
B
E
4
5
7
D
C

9) A



10) D (2 + 3) * (5 - 9/4)

Daftar Pustaka

Budayasa, I. K. (1997). *Matematika Diskrit I*. Surabaya: University Press IKIP SUrabaya.

Deo, N. (1994). *Graph Theory with Application to Engineering and Computer Science*. New Delhi: Prentice-Hall International, Inc.

Harary, F. (1969). *Graph Theory*. California: Addison Wesley Pbulishing Company.

Mayeda, W. (1992). *Graph Theory*. New York: John Eiley and Sons, Inc.

Suryadi, D. (1995/1996). *Materi Pokok Matematika Diskrit*. (Modul 5 dan Modul 6). Jakarta: Universitas Terbuka.

Suryanto. (1986). *Materi Pokok Pengantar Teori Graph*. Jakarta: Universitas Terbuka.

Sutarno, H., Priatna, N., dan Nurjanah (2004). *Matematika Diskrit*. Malang: IKIP Malang Press.

Tirta-Seputro, T. M. H. (1992). *Matematika Diskrit*. (terjemahan). Surabaya: University Press IKIP Surabaya.

Wilson, R, J. (1985). *Introduction to Graph Theory*. New York: John Wiley and Sons, Inc.

Glosarium

Ajasen
Dua buah titik disebut *ajasen*, jika kedua titik tersebut merupakan titik-titik ujung dari sisi yang sama.

Algoritma
Algoritma merupakan prosedur atau aturan. Contoh: algoritma Kruskal dan algoritma Huffman.

Bobot
Bobot suatu Graph G yaitu ukuran sisi-sisi dari graph G. Dapat juga merupakan bobot (jumlah) keseluruhan dari sisi-sisi graph G.

Derajat titik
Derajat titik V yaitu jumlah atau banyaknya sisi yang insiden dengan titik V.

Hutan
Hutan adalah graph tanpa sikel. Hutan merupakan gabungan dari beberapa pohon.

Insiden
Jika sebuah titik V merupakan titik ujung dari sisi E maka V dan E saling berinsidensi atau titik V *insiden* dengan sisi E.

Lintasan (path)
Suatu lintasan u-v (u-v path) dalam graph G adalah lintasan yang tidak mengulangi sebarang *sisi* (rusuk) dan tidak mengulangi sebarang *titik* (simpul).

Pohon
Pohon ialah graph terhubung yang tidak memiliki sikel.

Pohon berakar
Pohon berakar adalah graph berarah (*digraph*) T yang memenuhi syarat: (1) bila arah sisi diabaikan maka graphnya merupakan pohon, dan (2) ada titik tunggal R sehingga derajat masuknya 0 dan derajat masuk titik lainnya 1.

Pohon biner

Pohon biner adalah pohon berakar yang setiap titiknya memiliki paling banyak dua anak, yaitu anak kiri dan anak kanan.

Pohon jumlah

Pohon jumlah graph G adalah pohon yang dibentuk dengan menggunakan sisi dan titik graph G yang memuat semua titik di G.

Pohon jumlah minimal

Pohon jumlah minimal graph berbobot G adalah pohon jumlah yang bobotnya paling kecil.

Pohon rentang

Pohon rentang graph G adalah pohon pada graph G yang memuat semua titik di G.

Sikel

Sikel adalah suatu sirkuit yang tidak mengulang sebarang titik internalnya (titik dalamnya).

Sirkuit

Lintasan u-v, dengan u = v (titik awal sama dengan titik akhir) disebut sirkuit.