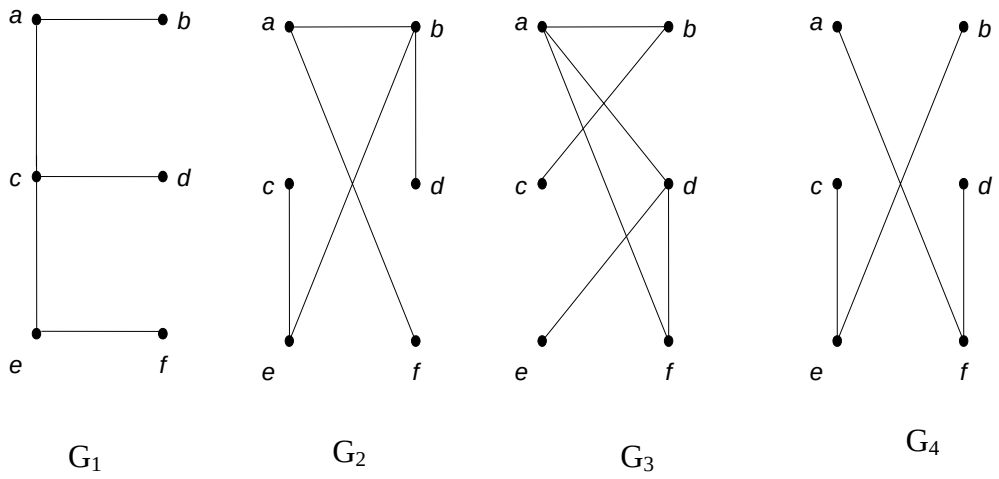
BAB V

P O H O N (T R E E)

Pohon (*tree*) merupakan salah satu bentuk khusus dari struktur suatu graf. Misalkan A merupakan sebuah himpunan berhingga simpul (*vertex*) pada suatu graf G yang terhubung. Untuk setiap pasangan simpul di A dapat ditentukan suatu lintasan yang menghubungkan pasangan simpul tersebut. Suatu graf terhubung yang setiap pasangan simpulnya hanya dapat dihubungkan oleh suatu lintasan tertentu, maka graf tersebut dinamakan pohon (*tree*). Dengan kata lain, pohon (*tree*) merupakan graf tak-berarah yang terhubung dan tidak memiliki sirkuit.

Contoh :



Gambar 6.1 G_1 dan G_2 adalah pohon, sedangkan G_3 dan G_4 bukan pohon

Hutan (*forest*) merupakan kumpulan pohon yang saling lepas. Dengan kata lain, hutan merupakan graf tidak terhubung yang tidak mengandung sirkuit. Setiap komponen di dalam graf terhubung tersebut adalah pohon. Pada gambar 6. 1 G_4 merupakan salah satu contoh hutan, yaitu hutan yang terdiri dari dua pohon.
Berikut adalah beberapa sifat pohon :

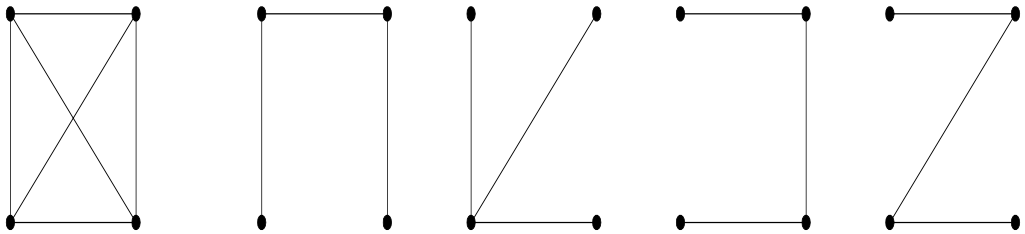
- Misalkan G merupakan suatu graf dengan *n* buah simpul dan tepat $n - 1$ buah sisi. Jika G tidak mempunyai sirkuit maka G merupakan pohon.
- Suatu pohon dengan *n* buah simpul mempunyai $n - 1$ buah sisi.
- Setiap pasang simpul di dalam suatu pohon terhubung dengan lintasan tunggal.
- Misalkan *G* adalah graf sederhana dengan jumlah simpul *n*, jika *G* tidak mengandung sirkuit maka penambahan satu sisi pada graf hanya akan membuat satu sirkuit.

5.1 Pohon Merentang Minimum (*Minimum Spanning Tree*)

Spanning Tree dari suatu graf terhubung merupakan subgraf merentang yang berupa pohon. Pohon merentang diperoleh dengan cara menghilangkan sirkuit di dalam graf tersebut.

Contoh *spanning tree* dari suatu graf terhubung (**Munir, 2003**) :

Perhatikan graf dibawah ini :



G T_1 T_2 T_3 T_4

Terlihat bahwa T_1, T_2, T_3, T_4 merupakan s*panning tree* dari graf G. Perlu diperhatikan bahwa setiap graf terhubung berbobot paling sedikit mempunyai satu buah *spanning tree*. Pohon rentang yang memiliki bobot minimum dinamakan pohon merentang minimum (*minimum spanning tree*). Dalam kehidupan nyata, salah satu contoh aplikasi s*panning tree* adalah menentukan rangkaian jalan dengan jarak total seminimum mungkin yang menghubungkan semua kota sehingga setiap kota tetap terhubung satu sama lain.

Dalam menentukan suatu *minimum spanning tree* dari suatu graf terhubung, kita dapat menentukannya dengan mengunakan dua cara yaitu algoritma Prim dan algoritma Kruskal.

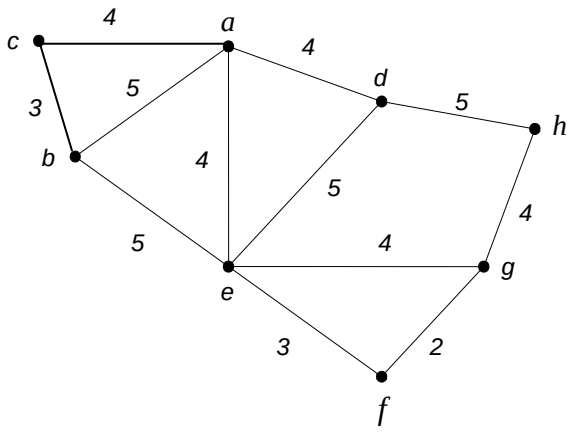
Algoritma Prim memiliki langkah-langkah sebagai berikut :

1. Pilih sisi dari graf G yang berbobot minimum, masukkan ke dalam T.
2. Pilih sisi (u, v) dalam G yang mempunyai bobot minimum dan bersisian dengan simpul di T, dengan syarat sisi tersebut tidak membentuk sirkuit di T. Masukkan (u, v) ke dalam T.
3. ulangi langkah 2 sebanyak $n - 2$ kali.

Jumlah langkah seluruhnya dalam algoritma Prim adalah sebanyak jumlah sisi di dalam *spanning tree* dengan n buah simpul, yaitu $(n - 1)$ buah.

Contoh :

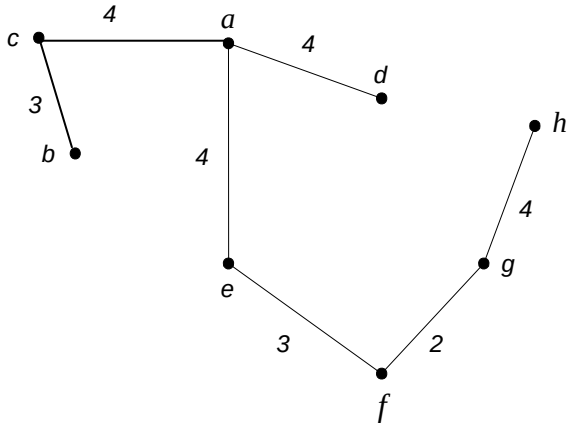
Tentukan *minimum spanning tree* dari graf dibawah ini :



Jawab :

- Pilih sisi *fg* sehingga kita mempunyai *T* ({*f, g*}, *fg*)
- Langkah selanjutnya dapat dipilih sisi *ef* karena sisi tersebut berbobot minimum yang bersisian dengan simpul *f* .
- Selanjutnya pilih sisi *ae* atau *gh* karena sisi tersebut berbobot minimum yang bersisian dengan simpul pada *T*, yaitu *e* dan *g*.

Jika proses ini dilanjutkan terus maka akan diperoleh *minimum spanning tree* seperti dibawah ini :

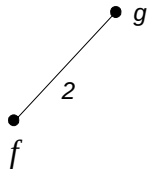


Terlihat bahwa *spanning tree* tersebut mempunyai total bobot 2 + 3 + 4 + 4 + 4 + 4 + 3 = 24.

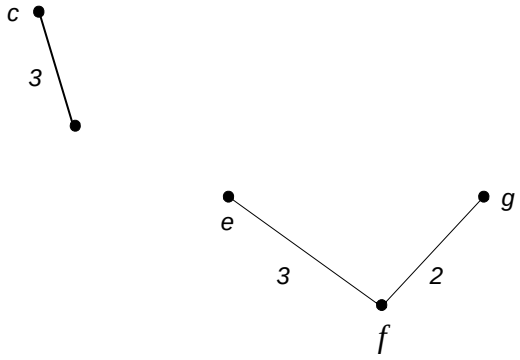
Langkah-langkah dalam algoritma Kruskal agak berbeda dengan algoritma Prim. Pada algoritma Kruskal, semua sisi dengan bobot yang minimal dimasukan kedalam *T* secara berurutan.

Langkah-langkah dalam menentukan *minimum spanning tree* dengan algoritma Kruskal adalah sebagai berikut :

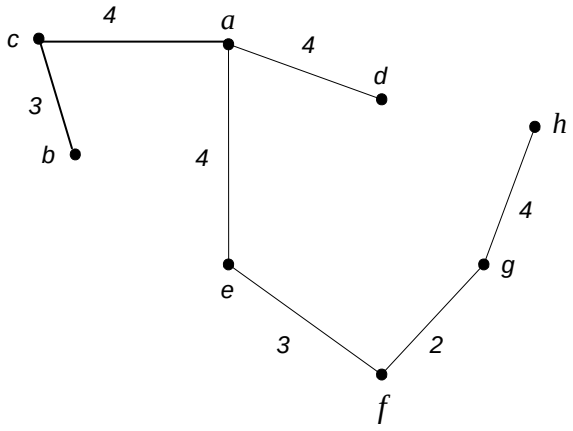
Langkah I : T berbentuk seperti pohon berikut



Langkah II : memasukan sisi-sisi yang berbobot 3 kedalam sehingga T berbentuk



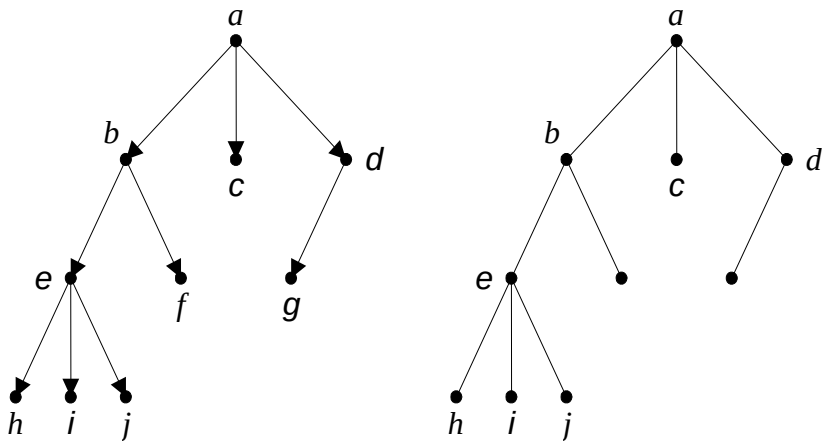
Langkah II : memasukan sisi-sisi yang berbobot 4 kedalam sehingga akhirnya diperoleh *minimum spanning tree* berikut :



5.2 Pohon Berakar

Pada suatu pohon, yang sisi-sisinya diberi arah sehingga menyerupai graf berarah, maka simpul yang terhubung dengan semua simpul pada pohon tersebut dinamakan **akar**. Suatu pohon yang satu buah simpulnya diperlakukan sebagai akar maka pohon tersebut dinamakan pohon berakar (*rooted tree*). Simpul yang berlaku sebagai akar mempunyai derajat masuk sama dengan nol. Sementara itu, simpul yang lain pada pohon itu memiliki derajat masuk sama dengan satu. Pada suatu pohon berakar, Simpul yang memiliki derajat keluar sama dengan nol dinamakan **daun**.

Contoh : Pohon Berakar (**Munir, 2003**)



Pohon berakar

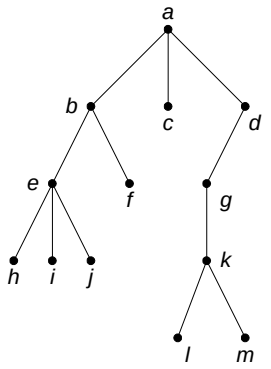
Pohon berakar setelah
tanda panah pada sisi dibuang

Pada pohon berakar diatas :

- *a* merupakan akar
- *c, d, f, g, h, i,* dan *j* merupakan daun

Terminologi pada Pohon Berakar

Perhatikan pohon berakar berikut ini :



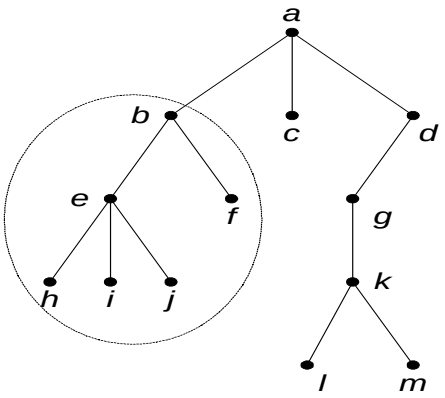
a. Anak (*child* atau *children*) dan Orangtua (*parent*)

b, *c*, dan *d* adalah anak-anak simpul *a*,
a adalah orangtua dari anak-anak itu

b. Lintasan (*path*)

Lintasan dari *a* ke *h* adalah *a*, *b*, *e*, *h*. dengan pnjang lintasannya adalah 3.
f adalah saudara kandung *e*, tetapi, *g* bukan saudara kandung *e*, karena orangtua mereka berbeda.

c. *Subtree*



c. Derajat (*degree*)

Derajat sebuah simpul adalah jumlah anak pada simpul tersebut.

Contoh :

Simpul yang berderajat 0 adalah simpul *c, f, h, I, j, l,* dan *m*.
Simpul yang berderajat 1 adalah simpul *d* dan g.
Simpul yang berderajat 2 adalah simpul *b* dan *k*.
Simpul yang berderajat 3 adalah simpul *a* dan *e*.

Jadi, derajat yang dimaksudkan di sini adalah derajat-keluar.
Derajat maksimum dari semua simpul merupakan derajat pohon itu sendiri. Pohon di atas berderajat 3

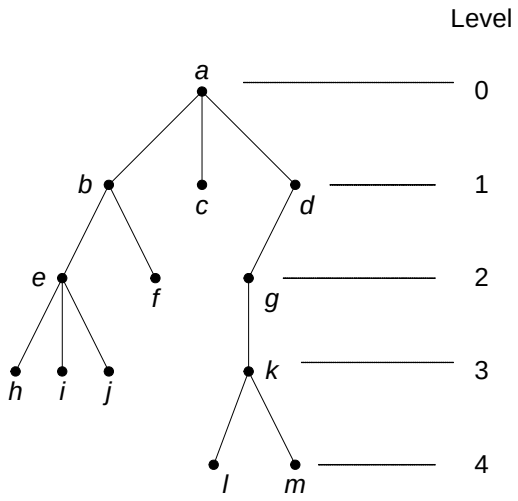
d. Daun (*leaf*)

Simpul yang berderajat nol (atau tidak mempunyai anak) disebut daun. Simpul *h, i, j, f, c, l,* dan *m* adalah daun.

e. Simpul Dalam (*internal nodes*)

Simpul yang mempunyai anak disebut simpul dalam. Simpul *b, d, e, g,* dan *k* adalah simpul dalam.

f. Aras (*level*) atau Tingkat



g. Tinggi (*height*) atau Kedalaman (*depth*)

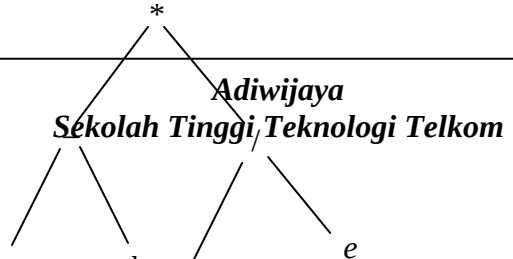
Aras maksimum dari suatu pohon disebut tinggi atau kedalaman pohon tersebut. Pohon di atas mempunyai tinggi 4.

Pohon berakar yang urutan anak-anaknya penting (diperhatiakn) maka pohion yang demikian dinamakan **pohon terurut** (*ordered tree*). Sedangka, pohon berakar yang setiap simpul cabangnya mempunyai paling banyak *n* buah anak disebut pohon *n-ary*. Jika $n = 2$, pohonnnya disebut pohon biner (*binary tree*).

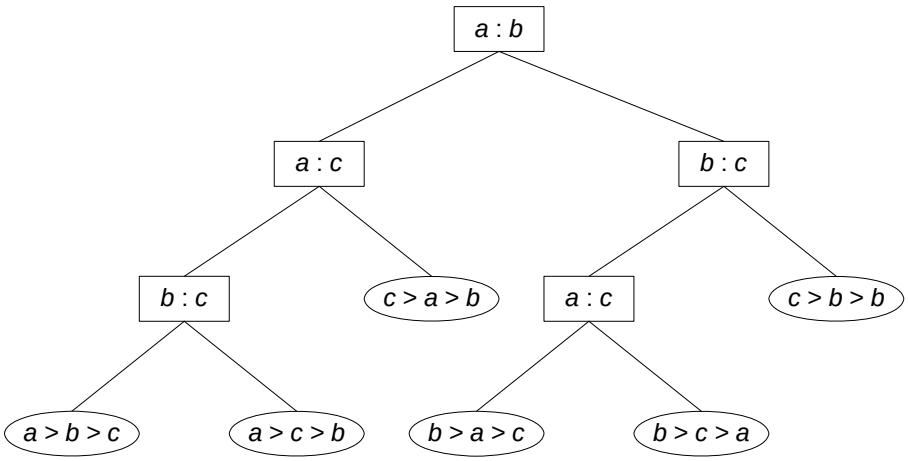
Contoh :

Berikut adalah beberapa contoh pohon biner :

1. Pohon Ekspresi
   Ekspresi aritmetika $(a - b)*((c + d) / e)$ dapat dinyatakan dalam suatu pohon biner, dimana peubah sebagai daun dan operator aritmetika sebagai simpul dalam dan akar.



2. Pohon keputusan (**Munir, 2004**)



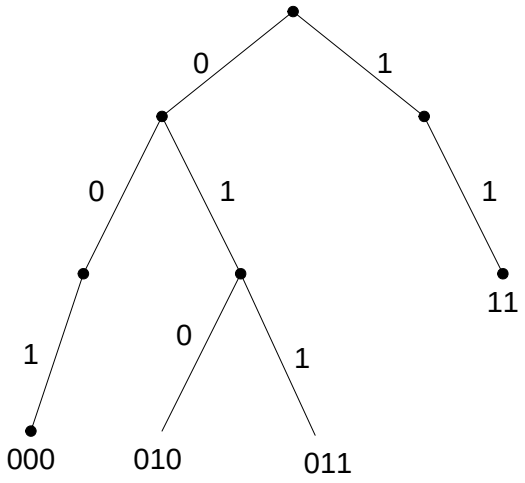
Pohon keputusan untuk mengurutkan 3 buah elemen

3. Kode awalan (*prefix code*)
   Kode awalan merupakan himpunan kode (salah satunya adalah kode biner) sedemikian sehingga tidak ada anggota himpunan yang merupakan awalan dari
   kode yang lain.

   Contoh :

   a. { 001, 010, 011, 11,} merupakan kode awalan
   b. {001, 010, 01, 111} bukan merupakan kode awalan, karena 01 merupakan awalan dari 010.

   Kode awalan (a) dapat dinyatakan dalam pohon biner, yaitu :



4. Kode Hufman

Dalam komunikasi data, seringkali ditemukan data berukuran besar sehingga waktu pengiriman data tersebut menjadi lama. Hal ini menyebabkan pentingnya kompresi data dengan tujuan memperkecil ukuran data tersebut. Kode Hufman merupakan salah satu metode pengkodean dalam hal kompresi data.
Perhatikan tabel kode ASCII berikut ini :

Simbol	Kode ASCII
A	01000001
B	01000010
C	01000011
D	01000100

Jadi rangkaian bit untuk string '*ADABCCA*' , dapat direpresentasikan dalam bentuk :

01000001010001000100000101000010010000011010000001101000001

atau

$7 \times 8 = 56$ bit (7 *byte*).

Tabel Tabel kekerapan dan kode Huffman untuk *string 'ABACCDA'*

Simbol	Kekerapan	Peluang	Kode Huffman
A	3	3/7	0
B	1	1/7	110
C	2	2/7	10
D	1	1/7	111

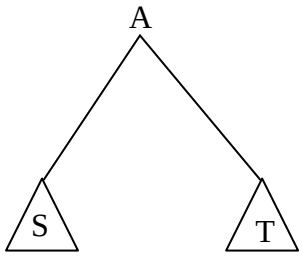
Sehingga rangkaian bit untuk string *'ADABCCA'*:

0111110010100

atau 13 bit.

5.3 Penelusuran Pohon Biner

Misalkan, berikut ini adalah pohon biner dimana A merupakan akar pohon biner tersebut. Sementara itu, S dan T merupakan upapohon (*subtree*) dari pohon biner.



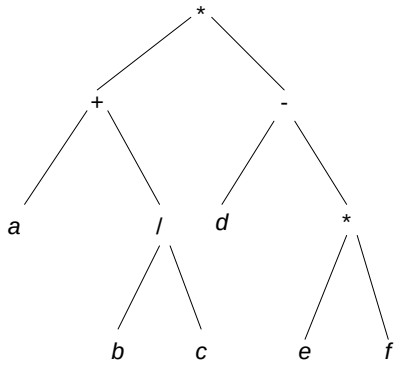
Ada tiga jenis penelusuran pohon biner diatas, antara lain :

1. *Preorder* : *A, S, T*
   - kunjungi *A*
   - kunjungi *S* secara *preorder*
   - kunjungi *T* secara *preorder*
2. *Inorder* : *S , A, T*
   - kunjungi *S* secara *inorder*
   - kunjungi *A*
   - kunjungi *T* secara *inorder*

3. *Postorder* : *S, T , A*
   - kunjungi *S* secara *postorder*
   - kunjungi *T* secara *postorder*
   - kunjungi *A*

Contoh :

Tentukan hasil penelusuran *preorder, inorder,* dan *postorder* dar pohon di bawah ini :

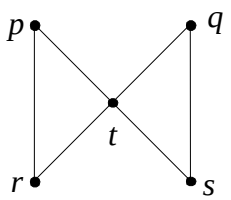


Jawab :

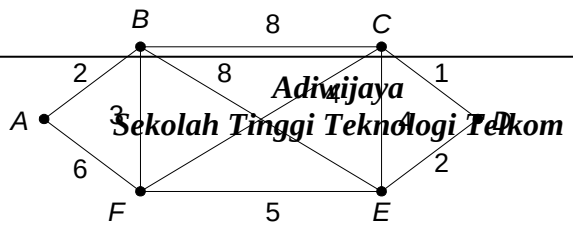
preorder	: * + *a* / *b* *c* - *d* * *e f*	(*prefix*)
inorder	: *a* + *b* / *c* * *d* - *e* * *f*	(*infix*)
postorder	: *a b c* / + *d e f* * - *	(*postfix*)

Latihan :

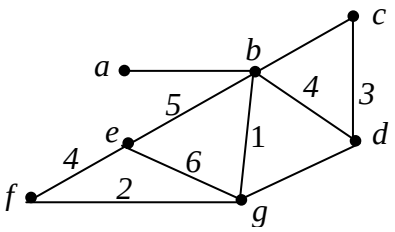
1. Tentukan semua spanning tree dari graf berikut :



2. Diketahui suatu graf seperti dibawah ini :
   a. graf G1



b. graf G2



Tentukan *minimum spanning tree* dengan menginakan :
a. Algoritma Prim
b. Algoritma Kruskal

3. Buat sketsa graf biner (pohon ekspresi) yang merepresentasikan ekpresi :
   a. p / (q – r)*(s + t)
   b. (p + q) / r – (s + t * u)

4. Tentukan hasil penelusuran dari pohon ekspresi pada soal no. 3 dalam bentuk *preorder, inorder,* dan *postorder*.

5. Pada graf dibawah ini, himpunan simpul mendefinisikan himpunan desa pada suatu kecamatan. Dalam rangka pembuatan jalan antar desa dibuatlah anggaran pembiayaan seperti tertulis sebagai bobot (dalam satuan juta rupiah) setiap sisi. Tentukan biaya minimum yang harus disiapkan dalam pembangunan jalan antar desa tersebut sehingga setiap desa pada kecamatan tersebut terhubung (ingat definisi terhubung pada suatu graf).