BAB II
LANDASAN TEORI

A. Optical Distribution Point (ODP)

Menurut Wahyudiono (2021), *ODP* merupakan sabuah perangkat pendukung layanan *fiber optik* yang berfungsi sebagai titik terminasi kabel *drop optic* atau tempat untuk membagi satu (*core optic*) ke beberapa pelanggan (terminal). ODP secara fisik berbentuk kotak yang berukuran lebih kecil dan berada dalam posisi diatas tiang outdoor maupun indoor. Terdapat beberapa komponen di dalam ODP yaitu terminal distribusi dan optical frame. Pembagian jalur optik di ODP menggunakan splitter berbanding 1:8. Jalur yang terdistribusi ke pelanggan dapat memuat 8 pelanggan dan bisa di maksimalkan dengan penggunaaan splitter 1:16.

ODP memiliki fungsi sebagai berikut :

1. Sebagai titik terminasi ujung kabel distribusi dan titik tambat awal atau titik pangkal kabel drop.
2. Sebagai titik distribusi dari kabel distribusi menjadi beberapa saluran kabel drop.
3. Sebagai tempat splitter.
4. Sebagai tempat penyambungan kabel serat optik.

ODP harus dilengkapi dengan ruang untuk splicing, ruang untuk splitter dan sistem pentanahan. Kapasitas ODP ada berbagai macam, yaitu ODP berkapasitas 8, 12, 16, 24, dan 48 port.

Kemudian jika ditinjau dari lokasi atau tempat pemasangannya, ODP dapat dibagi menjadi tiga tipe:

1. ODP tipe wall atau on pole. ODP jenis ini dapat dipasang di dinding atau dapat dipasang juga di atas tiang. ODP jeni ini digunakan untuk instalasi kabel aerial atau kabel udara.
2. ODP tipe pedestal. ODP jenis ini diinstalasi di atas permukaan tanah, dan ODP jenis ini juga digunakan untuk instalasi kabel drop bawah tanah dengan menggunakan pelindung pipa.

3. ODP tipe closure. ODP jenis ini sangan fleksibel dapat dipasang di bawah tanah dan juga dapat dipasang di atas di antara dua tiang.

Pada saat pemasangan ODP dengan tipe tertentu dapat atau menentukan penempatan ODP didasarkan pada efisiensi jaringan, kebutuhan layanan dan batas maksimum redaman yang diijinkan. ODP sendiri terdapat Passive splitter yang memiliki redaman yang cukup besar dan sangat berpengaruh terhadap kelayakan jaringan yang akan diinstalasi. Pada segmen distribusi splitter yang dipasang yaitu splitter 1:8 dan 1:6. Sedangkan PT. Telkom Indonesia lebih sering menggunakan passive plitter 1:8.

B. Teori Dasar Graf

Diberikan beberapa definisi dasar graf menurut Gary Chartrand(2000), yang meliputi definis graf, *adjancent, incident, u-v walk, circuit, cycle,* graf berbobot, dan graf terhubung.

Definisi 2.1. Graf *G* adalah himpunan tak kosong berhingga *V*(*G*) di sertai dengan relasi simetris dan tak refleksif *R* pada *V*(*G*). Dalam graf *G*, *V*(*G*) adalah suatau himpunan *vertex* dan *E*(*G*) disebut himpunan *edge*.

Banyaknya *vertex* pada *G* disebut *orde G*, di notasikan |*V*(*G*)|. Sedangkan banyaknya *edge* pada *G* disebut *size G*, dinotasikan |*E*(*G*)|. Sebuah graf harus memuat minimal satu *vertex* dan dimungkinkan tidak memiliki *edge*. Graf yang himpunan *edge*-nya adalah himpunan kosong dinamakan graf kosong atau null graf.

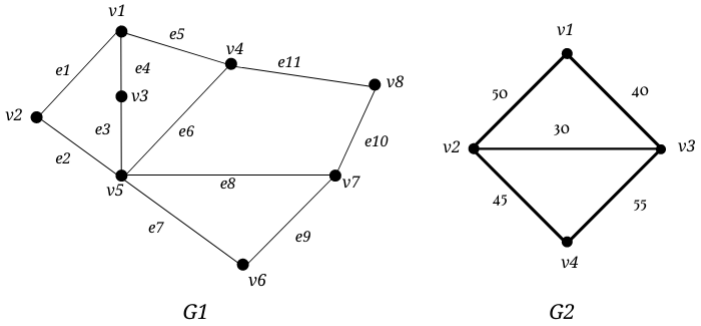
Definisi 2.2. *Vertex u* dan *v* di katakan *adjancent*, jika $uv \in E(G)$. Lebih lanjut *vertex u incident edge e* dan juga *vertex v incident* dengan *edge e*, jika $e = uv \in E(G)$, maka *u* dan *v* adalah kejadian dengan *e*.

Definisi 2.3. Suatu $u - v$ *walk* dalam graf *G* adalah barisan bergantian antara *vertex* dan *edge G* yang dimulai dari *vertex u* dan berakhir di *vertex v*, dengan setiap edge menghubungkan *vertex* berurutan dalam barisan tersebut. Suatu $u - v$ *trail* pada graf *G* adalah $u - v$ *walk* dimana tidak ada pengulangan *edge* di dalamnya. Suatu *u-v path* pada graf *G* adalah *u-v walk* yang tidak mengulang *vertex* manapun.

Definisi 2.4. *Circuit* adalah *u-v trail* dimana $u = v$ yang memuat sedikitnya tiga edge. Suatu *circuit* dimana tidak ada pengulangan *vertex* di dalamnya (kecuali *vertex* awal dan akhir) disebut *cycle.*

Definisi 2.5. Graf *G* disebut terhubung jika setiap dua *vertex* dalam graf G dapat di tentukan suatu path yang menghubungkan dua *vertex* tersebut. jika tidak, maka graf *G* tidak terhubung.

Definis 2.6. Graf *G* dikatakan berbobot apabila setiap *edgenya* diberi sebuah nilai(bobot).



Gambar 2.1 . Graf G_1 dan Graf G_2

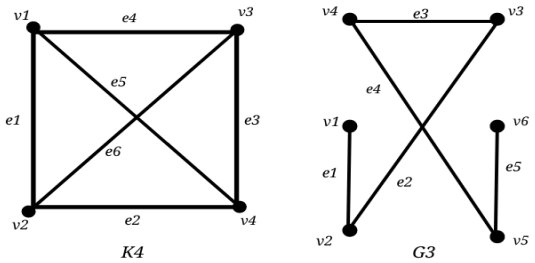
Gambar 2.1 graf G_1 merupakan contoh graf terhubung dengan $|V(G)| = 8$ dan $|E(G)| = 11$. Dari graf G_1 di dapatkan bahwa v_1 *adjancent* dengan v_2 dan e_1 *incident* dengan *vertex* v_1 dan v_2. Contoh $u - v\ walk$ adalah $v_1, e_1, v_2, e_2, v_5, e_7, v_6$. Contoh $u - v\ trail$ adalah $v_1, e_1, v_2, e_2, v_5, e_3, v_3, e_4, v_1$. Contoh $u - v\ path$ adalah $v_1, e_1, v_2, e_2, v_5, e_8, v_7, e_{10}, v_8$. Contoh *cycle* adalah $v_1, e_1, v_2, e_2, v_5, e_3, v_3, e_4, v_1$. Contoh *circuit* adalah $v_1, e_1, v_2, e_2, v_5, e_7, v_6, e_9, v_7, e_{10}, v_8, e_{11}, v_4, e_6, v_5, e_3, v_3, e_4, v_1$. Sedangkan graf G_2 adalah contoh graf berbobot.

C. Kelas – kelas graf

Berikut di jelaskan definisi graf lengkap dan graf pohon :

Definisi 2.6. Graf lengkap adalah graf sederhana yang setiap *vertex* mempunyai *edge* ke semua *vertex* lainnya. Graf lengkap dengan *n* buah *vertex* dilambangkan dengan K_n.

Definisi 2.7 Graf Pohon adalah graf terhubung sederhana yang tidak mengandung sirkuit.



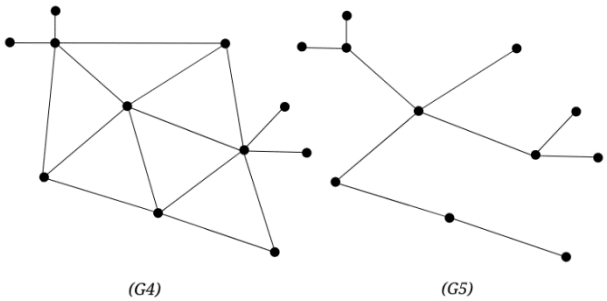
Gambar 2.2. Graf lengkap K_4, graf pohon G_3

Berdasarkan Gambar 2.2, graf K_4 merupakan contoh graf lengkap karena setiap *vertex* mempunyai *edge* ke semua *vertex* lainnya. Graf G_3 merupakan contoh graf pohon karena terhubung dan tidak mengandung *circuit*.

D. Spanning Tree (pohon merentang)

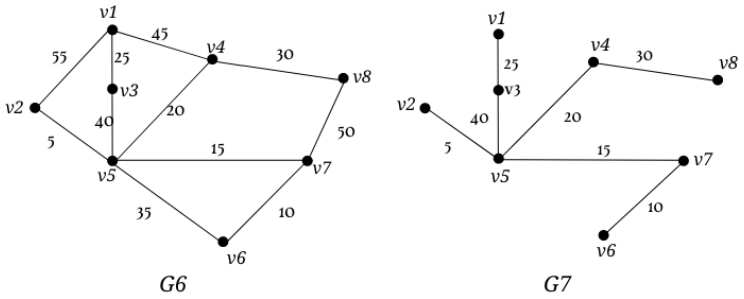
Pada subbab ini di jelaskan terkait *spening tree, minimum spening teree, maksimum spenning tree, akar, dan anak :*

Definisi 2.8. Graf *G(V,E)* dapat di ubah menjadi pohon $T = (V_1, E_1)$ dengan cara memutuskan *circuit – circuit* yang ada. Disebut pohon merentang karena semua *vertex* pada pohon *T* sama dengan semua *vertex* pada graf *G*, dan *edge-edge* pada pohon *T* ⊆ *edge-edge* pada graf *G*. dengan kata lain, $V_1 = V$ dan $E_1 \subseteq E$.



Gambar 2. 3 graf G_4 dan graf G_5

Definisi 2.9. Jika graf *G* adalah graf berbobot, maka pohon merentang T dari G disebut sebagai pohon merentang minimum (*minimum spanning tree*) apabila memiliki bobot minimum dalam *edgenya*.

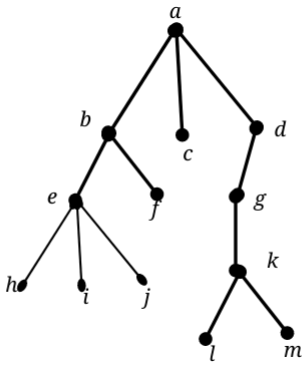


Gambar 2. 4 graf G_6 dan graf, G_7

Berdasarkan Gambar 2.4 graf G_7 merupakan graf minimum spanning tree dari graf G_6. Sedangkan graf G_6 merupakan graf maksimum spenning tree.

Definisi 2.10. Jika graf *G* adalah graf berbobot, maka pohon merentang T dari G| disebut sebagai pohon merentang maksimum (maksimum *spanning tree*) apabila memiliki bobot maksimal dalam *edgenya.*

Definisi 2.11. graf *G* dikatatakan pohon berakar, jika *veretxnya* diperlakukan sebagi akar dan *edge-edgenya* diberi arah menjauh. Sedangkan *vertex y* dikatakan anak *vertex x* jika ada edge dari *vertex x* ke *vertex y*.



Gambar 2. 5 pohon berakar

Berdasarkan gambar 2.5 *vertex a* merupakan akar, sedangkan *vertex e* anak dari *vertex b* dan *vertex a*, *vertex k* anak dari *vertex g* dan *vertex d*.

E. Algoritma *Branch & Bound*

Diberikan definisi terkait Algoritma *Branch and Bound*:

Definisi 2.13. Algoritma *Branch and Bound* merupakan algoritma yang membagi permasalahan menjadi sub masalah lebih kecil yang mengarah ke solusi dengan pencabangan (*branching*) dan melakukan pembatasan (*bounding*) untuk mencapai solusi optimal. Pencabangan (*branching*) yaitu proses membentuk permasalahan ke dalam bentuk struktur pohon pencarian (*search tree*). Proses pencabangan dilakukan untuk membangun semua cabang pohon yang menuju solusi, sedangkan proses pembatasan dilakukan dengan menghitung estimasi nilai (*cost*), *vertex* dengan memperhatikan *edge*. Algoritma *Branch and Bound* pertama kali dikemukakan oleh A. H. Land dan A. G. Doig pada tahun 1960.

Secara umum algoritma *Branch and Bound* dalam melakukan pencarian solusi menggunakan teknik *Least Cost Search* atau pencarian nilai terkecil, teknik ini akan menghitung nilai (*cost*) setiap *vertex*. *vertex* yang memiliki nilai paling kecil di katakan memiliki kemungkinan paling besar menuju solusi. Setiap *vertex* aktif mempunyai sebuah nilai yang menyatakan nilai batas (*bound*). Sebuah *vertex* aktif merupakan *vertex* yang mempunyai nilai batas terkecil (karena teknik pencarian solusi *Least Cost Search*).

Misal $c(i)$ adalah nilai estimasi lintasan minimum dari *vertex* i ke *vertex* tujuan, jadi $c(i)$ menyatakan batas (*bound*) nilai pencarian solusi dari *vertex* i. Sehingga dapat dirumuskan fungsi heuristic untuk menghitung nilai estimasi sebagai berikut:

$$c(i) = f(i) + g(i)$$

Dengan :

$c(i)$: Nilai untuk simpul i.

$f(i)$: Nilai lintasan dari simpul akar ke simpul .

$g(i)$: Nilai untuk mencapai simpul tujuan dari simpul i.

Untuk permasalahan yang lebih komplek, sistem di gambarkan dengan matriks *A* berukuran *n* × *n*. Untuk dipilih *vertex* mana yang akan dieksplorasi. *Vertex* yang dieksplorasi adalah *vertex* dengan nilai batas

terkecil. Nilai batas di dapatkan dari reduksi baris dan kolom matriks *A* yang merepresentasikan graf. Reduksi dilakukan dengan mengurangi nilai c_{ij} pada baris atau kolom dengan nilai c_{ij} terkecil pada baris atau kolom tersebut, sedemikian sehingga didapatkan matriks tereduksi *A(t)* dengan sebuah nilai nol pada setiap baris dan kolom. Selanjutnya total nilai pereduksi menjadi nilai batas simpul akar. Untuk setiap simpul anak yang di bangkitkan dengan mengunjungi *A(i, j),* dilakukan reduksi matriks *A(t)* untuk mendapatkan matriks tereduksi *A* (*j*, simpul awal) diubah menjadi ∞.

Nilai batas *vertex* anak di hitung menggunakan rumus :

$$\hat{c}(S) = \hat{c}(R) + A(i,j) + r$$

Dengan :

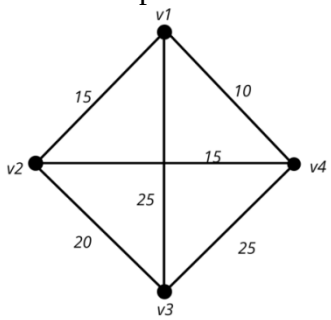
$\hat{c}(S)$: nilai perjalanan minimum yang melalui *vertex S*, dimana *S* adalah anak dari *vertex R*.

$\hat{c}(R)$: Nilai perjalanan minimum yang melalui *vertex R*, dimana *R* adalah orang tua dari *vertex S*.

$A(i,j)$: bobot *edge (i, j)* pada matriks tereduksi.

r : jumlah semua pereduksi pada proses reduksi matriks untuk *vertex S*.

Contoh; Menentukan rute terpendek dari semua *vertex* dengan v_1 sebagai titik awal?....



Gambar 2. 6 Graf dengan 4 *vertex*

Tabel 2. 1 Matrik rute graf

vertex	v_1	v_2	v_3	v_4
v_1	0	15	25	10
v_2	15	0	20	15
v_3	25	20	0	25
v_4	10	15	25	0

Langkah 1

1. Berangkat dari v_1 ke v_1, v_2 ke v_2 dan seterusnya adalah titik diijinkan sehingga diberi tanda ∞ untuk sel dalam matriks.
2. Untuk permasalahan minimalisasi, cari nilai jarak terkecil untuk setiap baris. Kemudian nilai jarak terkecil digunakan untuk mengurangi semua nilai jarak yang ada pada baris yang sama.
3. Memastikan semua baris dan kolom memiliki nilai nol. Apabila masih ada kolom yang memiliki nilai nol, maka cari nilai jarak terkecil untuk setiap kolom. Kemudian nilai jarak terkecil digunakan untuk mengurangi semua nilai jarak yang ada pada kolom yang sama. Hasil matriks yang dikurangkan elemen terkecil disetiap baris dan setiap kolom disebut matriks (C_{ij}). Semua angka yang digunakan untuk mengurangi tiap baris dan setiap kolom tersebut kemudian dijumlahkan, hasil daripenjumlahan inilah yang kemudian dijadikan sebagai *r(root)* atau bobot dari *vertex* awal atau akar. Hal ini juga berati bahwa solusi pada persoalan ini paling tidak memiliki bobot minimum sebesar *r(root).*

Reduksi matriks

$$A = \begin{bmatrix} \infty & 15 & 25 & 10 \\ 15 & \infty & 20 & 15 \\ 25 & 20 & \infty & 25 \\ 10 & 15 & 25 & \infty \end{bmatrix} \begin{matrix} -10 \\ -15 \\ -20 \\ -10 \end{matrix} => \begin{bmatrix} \infty & 5 & 15 & 0 \\ 0 & \infty & 5 & 0 \\ 5 & 0 & \infty & 5 \\ 0 & 5 & 15 & \infty \end{bmatrix} \begin{matrix} C_3 - 5 \\ => \end{matrix} \begin{bmatrix} \infty & 5 & 10 & 0 \\ 0 & \infty & 0 & 0 \\ 5 & 0 & \infty & 5 \\ 0 & 5 & 10 & \infty \end{bmatrix}$$

$$r = 10 + 15 + 20 + 10 + 5 = 60$$

Langkah 2

Misalkan *A* adalah matriks tereduksi untuk *vertex R*. *Vertex S* adalah anak *vertex R* sedemikian sehingga dalam perjalanan *edge*(*S*,*R*) pada ruang pohon pencarian berhubungan dengan *edge* (*i*,*j*). Jika *S* dapat dihitung dengan langkah sebagai berikut:

1. Ubah semua nilai *I* dan kolom *j* menjadi ∞, untuk mencegah agar tidak ada lintasan yang keluar dari *vertex i* atau masuk pada *vertex j*.
2. Ubah *A(i,j)* menjadi ∞, ini untuk mencegah penggunaan *edge* (*j*,*i*), dan

3. Reduksi kembali semua baris dan kolom pada matriks *A* kecuali untuk elemen ∞.

Perhatikan perhitungan *vertex* – *vertex* berikut ini:

1. Lintasan $v_1 - v_2 = 5$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 0 & 0 \\ 5 & \infty & \infty & 5 \\ 0 & \infty & 10 & \infty \end{bmatrix} B_3 - 5 \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 0 & 0 \\ 0 & \infty & \infty & 0 \\ 0 & \infty & 10 & \infty \end{bmatrix}$$

Terdapat reduksi pada baris ke 3 sebesar 5, jadi ***r*** = 5. Maka nilai batas *vertex* pada *search tree* adalah

$$\hat{C}(S) = \hat{C}(R) + Aij + r = \ 60 + 5 + 5 = 70.$$

2. Lintasan $v_1 - v_3 = 10$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 0 \\ 5 & 0 & \infty & 5 \\ 0 & 5 & \infty & \infty \end{bmatrix}$$

Tidak ada *reduksi* yang dilakukan, jadi *r* = 0. Jadi nilai batas *vertex* pada *search tree* adalah

$$\hat{C}(S) = \hat{C}(R) + Aij + r = \ 60 + 10 + 0 = 70.$$

3. Lintasan $v_1 - v_4 = 0$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 0 & \infty \\ 5 & 0 & \infty & \infty \\ 0 & 5 & 10 & \infty \end{bmatrix}$$

Tidak ada *reduksi* yang dilakukan, jadi *r* = 0. Jadi nilai batas *vertex* pada *search tree* adalah

$$\hat{C}(S) = \hat{C}(R) + Aij + r = \ 60 + 0 + 0 = 60.$$

Lintasan *search tree* terpendek berada pada lintasa $v_1 - v_4$

4. Lintasan $v_1 - v_4 - v_2 (v_4 - v_2 = 5)$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 0 & \infty \\ 5 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix} B_3 - 5 \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & 0 & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Terdapat *reduksi* pada baris ke 3 sebesar 5, jadi *r* = 5. Maka nilai batas *vertex* pada *search tree* adalah

$$\hat{C}(S) = \hat{C}(R) + Aij + r = 60 + 5 + 5 = 70.$$

5. Lintasan $v_1 - v_4 - v_3 (v_4 - v_3 = 10)$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ 5 & 0 & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Tidak ada *reduksi* yang dilakukan, jadi r = 0. Jadi nilai batas *vertex* pada *search tree* adalah

$$\hat{C}(S) = \hat{C}(R) + Aij + r = 60 + 10 + 0 = 70.$$

6. Lintasan $v_1 - v_4 - v_2 - v_3 (v_2 - v_3 = 0)$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Tidak ada *reduksi* yang dilakukan, jadi r = 0. Jadi nilai batas *vertex* pada *search tree* adalah.

$$\hat{C}(S) = \hat{C}(R) + Aij + r = 70 + 0 + 0 = 70$$

7. Lintasan $v_1 - v_4 - v_3 - v_2 (v_3 - v_2 = 0)$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Tidak ada *reduksi* yang dilakukan, jadi r = 0. Jadi nilai batas *vertex* pada *search tree* adalah.

$$\hat{C}(S) = \hat{C}(R) + Aij + r = 70 + 0 + 0 = 70$$

8. Lintasan $v_1 - v_4 - v_3 - v_2 - v_1 (v_2 - v_1 = 0)$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Tidak ada *reduksi* yang dilakukan, jadi r = 0. Jadi nilai batas *vertex* pada *search tree* adalah.

$$\hat{C}(S) = \hat{C}(R) + Aij + r = 70 + 0 + 0 = 70$$

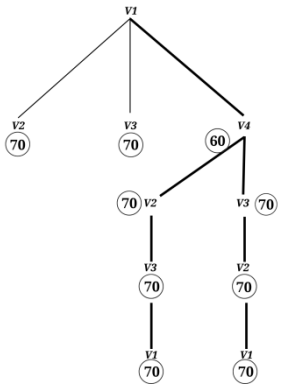
9. Lintasan $v_1 - v_4 - v_2 - v_3 - v_1 (v_3 - v_1 = 0)$

$$\begin{bmatrix} \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty \end{bmatrix}$$

Tidak ada *reduksi* yang dilakukan, jadi r = 0. Jadi nilai batas *vertex* pada *search tree* adalah.

$$\hat{C}(S) = \hat{C}(R) + Aij + r = 70 + 0 + 0 = 70$$

Karena tidak ada lagi *vertex* di *search tree*, maka solusi persoalan di atas adalah $v_1 - v_4 - v_2 - v_3 - v_1$ atau $v_1 - v_4 - v_3 - v_2 - v_1$ dengan bobot minimal 70



Gambar 2. 7 Search Tree Matriks A

F. Penelitian Yang Relevan

Dalam penelitian yang berjudul "Penentukan Rute Terpendek Pemeliharaan *Optical Distribution Point* (*ODP*) Alogaritma *Brach And Bound* ", ada beberapa penelitian yang relevan sebagai referensi dari penulis. Berikut merupakan persamaan dan perbedaan penelitian ini dengan penelitian yang lainnya :

Tabel 2. 2 penelitian yang relevan

No	Penulis	Judul	Persamaan	Perbedaan
1.	St. Khadija, Syarli dan Akhmad Qashlim	Sisitem Informasi Jalur Terpendek Menggunakan Algoritma Dijkstra (Jurnal JPCS; Vol. 2(1), 2020)	Menentukan Jalur Terpendek	Metode yang digunakan Algoritma *Dijkstra*. Sedangkan Dalam Penelitian Yang Dilakukan Oleh Penulis Adalah Menggunakan Algoritma *Branch and Bound*.
2.	Tita Wulansari dan Tiany Martha Aditya	Penentuan Rute Optimal Distribusi Paving Block (Jurnal Teknik Industri; Vol. 6(2), 2020)	Menentukan Rute Terpendek Dengan Menggunakan Metode *Branch And Bound.*	Penentuan Rute Optimal Distribusi *Paving Blok.* Sedangkan Dalam Penelitian Yang Dilakukan Oleh Penulis Menentukan Rute Terpendek Pemeliharaan *ODP*.

No	Penulis	Judul	Persamaan	Perbedaan
3.	Made Irma Dwiputranti dan Vinny Meidiana Putri	Penerapan *Branch and Bound* Untuk Alternatif Pemilihan Rute Terpendek Dalam Pengiriman Dokumen (Jurnal Logistik Bisnis; Vol. 10(02), 2020)	Menentukan Rute Terpendek Dengan Menggunakan Metode *Branch And Bound.*	Penentuan Jalur Terpendek Yang Digunakan Untuk Meminimalkan Waktu Distribusi Dengan Cara Mencari Jarak Dan Rute Terdekat. Sedangkan Dalam Penelitian Yang Dilakukan Oleh Penulis Menentukan Rute Terpendek Pemeliharaan *ODP*.
4.	Gratia Melina Sari, Rainisa Heryanto dan Santoso	Penentuan Rute Distribusi Menggunakan Model Integer Linear Programming Dengan Metode Branch And Bound (Jurnal Teknik Sistem dan Industri; Vol.01(01), 2020)	Menentukan Rute Dengan Menggunakan Metode *Branch And Bound.*	Menentukan Rute Distribusi Menggunakan Model *Integer Linier*. Sedangkan Dalam Penelitian Yang Dilakukan Oleh Penulis Menggunakan Model Manual.
5.	Windi Rayina Rosa, Suhartono dan Helmie Arif Wibawa	Penentuan Jalur Terpendek Pada Pelayanan Agen Travel (Jurnal Masyarakat Informatika; Vol.4(2), 2018)	Menentukan Rute Dengan Menggunakan Metode *Branch And Bound*	Menentukan Rute Terpendek Pada Pelayanan Agen Travel. Sedangkan Dalam Penelitian Yang Dilakukan Oleh Penulis Menentukan Rute Terpendek Pemeliharaan *ODP*.
6.	Chandra Desparaja dan R. Faris Gumelar	Pendistribusian Produk Kartu Seluler Untuk Alternatif Rute Terpendek (Jurnal Teknik; Vol.19(01), 2020)	Menentukan Rute Dengan Menggunakan Metode *Branch And Bound.*	Menentukan Rute Terpendek Pendistribusian Produk Kartu Seluler. Sedangkan Dalam Penelitian Yang Dilakukan Oleh Penulis Menentukan Rute Terpendek Pemeliharaan *ODP*.
7.	Abd Aziz Mursy, Hibban Kholiq dan Diah Ayu Saptyaning tyas	Menentukan Rute Terpendek Pendistribusian Bahan Bangunan (Journal *Eigen Mathematics*; Vol. 2(1) 2019)	Menentukan Rute Dengan Menggunakan Metode *Branch And Bound.*	Menentukan Rute Terpendek Pendistribusian Bahan Bangunan. Sedangkan Dalam Penelitian Yang Dilakukan Oleh Penulis Menentukan Rute Terpendek Pemeliharaan *Odp.*

No	Penulis	Judul	Persamaan	Perbedaan
8.	Muhammad Faizal, Sofia Dwiagnes dan Nitta Fitria Anggraeni	Pencarian Rute Terpendek Pada Distribusi Raw Material (Jurnal Teknik; Vol. 19(01), 2020)	Menentukan Rute Terpendek.	Menentukan Rute Terpendek Menggunakan Metode Djikstra, Sedangkan Dalam Penelitian Yang Dilakukan Oleh Penulis Menentukan Rute Terpendek Menggunakan Metode Algoritma *Branch And Bound.*
9.	Daniel B. Pailin dan Johan M. Tupan	Optimasi Rute Distribusi Produk Nestle (Seminar Nasional "Archipelago Engineering" (ALE) 2018)	Menentukan Rute Dengan Menggunakan Metode *Branch And Bound.*	Menentukan Rute Terpendek Menggunakan Metode *Branch And Bound* Dan *Two-Way Exchange Improvement Heuristic.* Sedangkan Dalam Penelitian Yang Dilakukan Oleh Penulis Menentukan Rute Terpendek Menggunakan Metode Algoritma *Branch And Bound.*
10.	Febri Triyanto, Hari Adianto dan Susy Susanty	Usulan Rancangan Rute Distribusi Gas LPG (Jurnal Teknik Industri; Vol. 03(03) 2015)	Menentukan Rute Dengan Menggunakan Metode *Branch And Bound.*	Membahas pemecahan permasalah Vehicle Routing Problem Penelitian Yang Dilakukan Oleh Penulis Menentukan Rute Terpendek Pemeliharaan *Odp.*