

PENYELESAIAN TRAVELLING SALESMAN PROBLEM DENGAN MENGGUNAKAN APLIKASI BAHASA PEMROGRAMAN PASCAL DAN C++

Debora Exaudi Sirait*

Universitas Sam Ratulangi Manado, Sulawesi Utara-Indonesia 95115

Switamy Agnitha Purba

Politeknik Teknologi Kimia Industri, Medan-Indonesia 20229

Lolyta Damora Simbolon

Universitas HKBP Nomensen Medan-Indonesia 20235

Abstrak. *Persoalan travelling salesman problem merupakan salah satu masalah optimasi klasik yang banyak diaplikasikan dalam kehidupan nyata, seperti pendistribusian barang, perencanaan rute transportasi, hingga pergerakan robot industri. Penelitian ini bertujuan mengimplementasikan algoritma penyelesaian travelling salesman problem menggunakan dua bahasa pemrograman berbeda, yaitu Pascal dan C++, serta membandingkan kinerja keduanya dari sisi efektivitas, efisiensi memori, dan kecepatan eksekusi. Metode yang digunakan meliputi pemodelan graf berbobot, perhitungan lintasan terpendek, dan penerapan algoritma Branch and Bound. Hasil pengujian menunjukkan bahwa kedua bahasa pemrograman menghasilkan rute optimal yang identik, yaitu 1 – 5 – 6 – 7 – 10 – 9 – 8 – 4 – 2 – 3 – 1 dengan total jarak minimum 196. Program Pascal memiliki keunggulan pada struktur sintaks yang sederhana dan penggunaan memori rendah (± 356 KB), sedangkan C++ unggul dalam kecepatan eksekusi (0,09466 detik) meskipun ukuran berkas output lebih besar ($\pm 1,26$ MiB). Perbedaan kinerja ini menunjukkan bahwa Pascal lebih sesuai untuk tujuan edukasi dan aplikasi berskala kecil, sementara C++ lebih unggul untuk aplikasi kompleks dan kebutuhan integrasi sistem modern. Dengan demikian, algoritma TSP dapat diimplementasikan secara efektif baik menggunakan Pascal maupun C++, dan pemilihan bahasa pemrograman sebaiknya disesuaikan dengan konteks kebutuhan.*

Kata Kunci: *travelling salesman problem, bahasa pemrograman, Pascal, C++*

Abstract. *The travelling salesman problem is one of the classic optimisation problems that is widely applied in real life, such as in the distribution of goods, transportation route planning, and even the movement of industrial robots. This study aims to implement algorithms for solving the travelling salesman problem using two different programming languages, namely Pascal and C++, and to compare their performance in terms of effectiveness, memory efficiency, and execution speed. The methods used include weighted graph modelling, shortest path calculation, and the application of the Branch and Bound algorithm. The test results show that both programming languages produce identical optimal routes, namely 1 – 5 – 6 – 7 – 10 – 9 – 8 – 4 – 2 – 3 – 1 with a minimum total distance of 196. The Pascal programme has the advantage of a simple syntax structure and low memory usage (± 356 KB), while C++ excels in execution speed (0.09466 seconds) despite having a larger output file size (± 1.26 MiB). This difference in performance indicates that Pascal is more suitable for educational purposes and small-scale applications, while C++ is superior for complex applications and modern system integration needs. Thus, the TSP algorithm can be implemented effectively using either Pascal or C++, and the choice of programming language should be tailored to the context of the requirements.*

Keywords: *travelling salesman problem, programming language, Pascal, C++*

Sitasi: Sirait, D.E., Purba, S.A., Simbolon, L.D. 2025. Penyelesaian Travelling Salesman Problem dengan Menggunakan Aplikasi Pemrograman Pascal dan C++. *MES (Journal of Mathematics Education and Science)*, 11(1): 65-74.

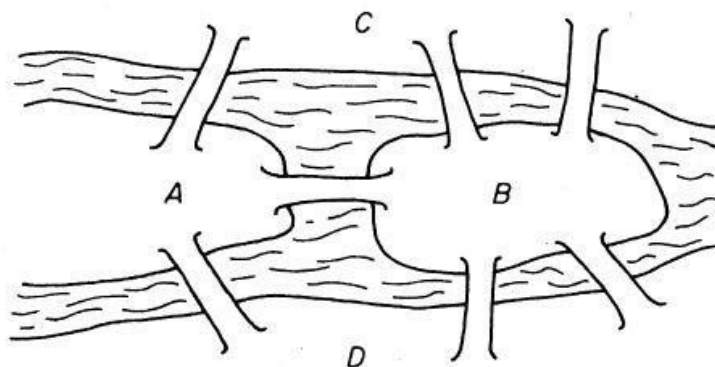
Submit: 10 September 2025	Revise: 24 September 2025	Accepted: 11 Oktober 2025	Publish: 14 Oktober 2025
-------------------------------------	-------------------------------------	-------------------------------------	------------------------------------

PENDAHULUAN

Banyak terapan Travelling Salesman Problem (TSP) yang muncul dalam kehidupan sehari-hari. Persoalan TSP merupakan salah satu masalah optimasi klasik yang banyak diaplikasikan dalam kehidupan nyata, seperti pendistribusian barang, perencanaan rute transportasi, hingga pergerakan industri robot. Beberapa contoh: misalkan sebuah mobil pos ditugaskan mengambil surat dari kotak pos yang tersebar pada n buah lokasi diberbagai sudut kota. Satu simpul menyatakan kantor pos tempat mobil pos mulai berangkat. Sisi (i, j) diberi bobot yang sama dengan jarak dari kotak pos i ke kotak pos j . rute yang dilalui mobil pos adalah sebuah perjalanan(tour) yang mengunjungi setiap kotak pos hanya satu kali dan kembali lagi ke kantor pos asal. Kita harus menentukan rute perjalanan yang mempunyai total jarak terpendek.

Dalam industri robot, misalkan kita ingin menggunakan lengan robot untuk mengencangkan mur pada beberapa buah peralatan mesin dalam sebuah jalur perakitan. Lengan robot mulai berada dari posisi awalnya, lalu berturut-turut pindah ke mur-mur berikutnya dan kembali lagi keposisi awalnya. Siklus yang dibentuk jelaslah perjalanan mengunjungi simpul-simpul sebuah graf. Tiap simpul menyatakan mur, sisi menyatakan perpindahan diantara dua simpul. Perjalanan dengan biaya minimum berarti meminimumkan waktu yang dibutuhkan robot untuk menyelesaikan tugasnya, yaitu total waktu perpindahan dari sebuah mur ke mur lainnya.

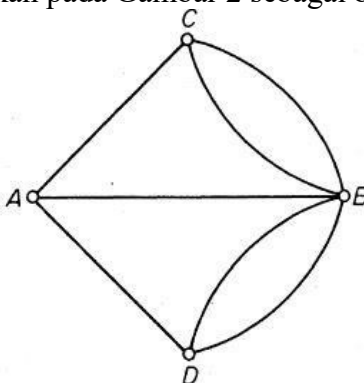
Hal ini tidak jauh berbeda dengan pengaplikasian teori graf dalam permasalahan yang relevan. Teori graf sebenarnya merupakan topik lama namun memiliki banyak aplikasi modern seperti kimia molekul dan sistem jaringan. Graf pertama kali dikenal pada saat seorang matematikawan Swiss, bernama Leonhard Euler (1736), berhasil mengungkap misteri teka-teki jembatan Koningsberg mengalir sebuah sungai bernama sungai Pregel yang membelah kota. Bagian aliran sungai tersebut ada yang mengapit suatu pulau dan kemudian aliran sungai bercabang menjadi 2 bagian. Dengan demikian aliran sungai Pregel membelah kota Koningsberg dan memisahkan daratannya menjadi 4 bagian (misal daratan A, B, C dan D). Untuk menghubungkan keempat daratan tersebut dibuat 7 jembatan seperti terlihat pada Gambar 1 berikut.



Gambar 1. Jembatan Koningsberg

Masalah yang berkaitan dengan jembatan koningsberg tersebut adalah apakah seseorang dapat melakukan perjalanan berangkat dari suatu daratan kembali ke daratan tersebut melewati semua jembatan yang ada masing-masing tepat satu kali? Masalah tersebut sangat menarik penduduk setempat untuk mengetahui jawabannya. Sayangnya, penyelesaian permasalahan tersebut hanya dilakukan coba-coba, yaitu dengan mencoba langsung melewati jembatan-jembatan yang ada. Walaupun tidak seorang pun yang berhasil menempuh perjalanan yang demikian, namun masyarakat di tempat itu tetap belum yakin bahwa memang tidak mungkin melakukan perjalanan yang demikian.

Misteri teka-teki jembatan Koningsberg tersebut tetap tidak terpecahkan sampai munculnya pembuktian secara matematis dari Euler. Langkah pembuktian yang dilakukan Euler menggunakan bantuan sebuah graf sebagai model, yaitu dengan melambangkan setiap daratan yang dipisahkan oleh sungai Pregel dengan sebuah titik, sedangkan setiap jembatan dilambangkan dengan sebuah rusuk. Dengan demikian, terdapat 4 titik dan 7 rusuk, sehingga model tersebut dapat digambarkan pada Gambar 2 sebagai berikut:



Gambar 2. Graf yang Merepresentasikan Jembatan Koningsberg

Setelah melewati proses analisis yang cermat, akhirnya Euler sampai pada kesimpulan bahwa tidak terdapat perjalanan dari satu daratan kembali ke daratan tersebut melewati semua jembatan masing-masing tepat satu kali. Kesimpulan tersebut dituangkan dalam papernya yang berjudul: *solution problematis ad geometrian situs pertition (the solution of a problem relating to the geometry of positions)*.

Kedua permasalahan di atas sangat erat kaitannya dalam menentukan lintasan terpendek. Munir (2010) menyatakan bahwa persoalan mencari lintasan terpendek didalam graf merupakan salah satu persoalan optimasi. Graf yang digunakan dalam pencarian lintasan terpendek adalah graf berbobot (*weighted graph*), yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot. Bobot pada sisi graf dapat menyatakan jarak antar kota, waktu pengiriman pesan, ongkos, dan sebagainya. Contoh terapan pencarian lintasan terpendek misalnya: **Pertama**, Misalkan simpul pada graf dapat berupa kota, sedangkan sisi menyatakan jalan yang menghubungkan dua buah kota. Bobot pada sisi graf dapat menyatakan jarak antara dua buah kota atau rata-rata waktu tempuh antara dua buah kota. Apabila terdapat lebih dari satu lintasan dari kota A ke B, maka persoalan lintasan terpendek adalah menentukan jarak terpendek atau waktu tersingkat dari kota A ke B. **Kedua**, misalkan simpul pada graf merupakan terminal komputer atau simpul komunikasi dalam suatu jaringan, sedangkan sisi menyatakan saluran komunikasi yang menghubungkan dua buah terminal. Bobot pada graf dapat menyatakan biaya pemakaian saluran komunikasi antara dua buah terminal, jarak antara dua buah terminal, dan waktu pengiriman pesan. Persoalan lintasan terpendeknya adalah menentukan jalur komunikasi terpendek antara dua buah terminal komputer. Lintasan terpendek akan menghemat waktu pengiriman pesan dan biaya komunikasi.

METODE

Penelitian ini bertujuan mengimplementasikan algoritma penyelesaian TSP menggunakan dua bahasa pemrograman berbeda, yaitu Pascal dan C++, serta membandingkan kinerja keduanya dari sisi efektivitas, efisiensi memori, dan kecepatan eksekusi. Metode yang digunakan meliputi pemodelan graf berbobot, perhitungan lintasan terpendek, dan penerapan algoritma Branch and Bound.

Bahasa Pemrograman Pascal

Sebelum matriks digunakan untuk menyimpan data, terlebih dahulu matriks harus dideklarasikan. Mendeklarasikan matriks artinya menentukan nama matriks, tipe data dan elemen-elemennya, dan ukurannya. Pendeklarasian matriks di dalam teks algoritma di dalam bagian DEKLARASI. Kata kunci yang dipakai untuk matriks tetap array.

Ada beberapa cara mendeklarasikan matriks:

Sebagai nama peubah. Contoh:

DEKLARASI

M: array[1..5,1..4] of integer

Sebagai tipe bentukan. Contoh:

DEKLARASI

Type Mat: array[1..5,1..4] of integer

Mendefinisikan ukuran maksimum matriks sebagai sebuah konstanta. Contoh:

DEKLARASI

const NbarisMaks=20 (jumlah baris maksimum)

const NkolomMaks=20 (jumlah kolom maksimum)

M: array[1..NbarisMaks,1..NkolomMaks] of integer

Contoh Algoritma Matriks:

Procedure BacaMatriks1(var M: MatriksInt; Nbar, Nkol: integer);

{mengisi elemen matriks M[1..NbarisMaks,1..NkolomMaks] dari papan ketik}

Var

i : integer; {indeks baris}

j : integer; {indeks kolom}

begin

for i := 1 to Nbar do

begin

write ('Ketikkan elemen M[' , i , ', ', j , ']:');

readln (M[i,j]);

end;

end;

{seluruh elemen matriks M sudah berisi nilai yang dibaca dari papan ketik}

Contoh pemanggilan prosedur di atas:

Writeln('Memasukkan data ukuran matriks:');

Write('Berapa jumlah baris (1-20)?'); readln(p);

Write('Berapa jumlah kolom (1-20)?'); readln(q);

BacaMatriks(M, p, q);

Bahasa Pemrograman C++

Matriks dalam C++ diimplementasikan sebagai array dua dimensi. Deklarasi umum:

tipe_data nama_array[baris][kolom];

Contoh:

int M[3][4]; // Matriks 3 baris $\times$ 4 kolom bertipe int.

Elemen matriks diakses dengan indeks [i][j], dimulai dari 0.

Contoh Program Input dan Tampilkan Matriks

```
#include <iostream>
using namespace std;
int main() {
    int baris, kolom;
    cout << "Masukkan jumlah baris: ";
    cin >> baris;
    cout << "Masukkan jumlah kolom: ";
    cin >> kolom;
    int matriks[50][50];
    cout << "\nMasukkan elemen matriks:\n";
    for (int i = 0; i < baris; i++) {
        for (int j = 0; j < kolom; j++) {
            cout << "Elemen [" << i << "][" << j << "]: ";
            cin >> matriks[i][j];
        }
    }
    cout << "\nMatriks yang dimasukkan:\n";
    for (int i = 0; i < baris; i++) {
        for (int j = 0; j < kolom; j++) {
            cout << matriks[i][j] << "t";
        }
        cout << endl;
    }

    return 0;
}
```

HASIL DAN PEMBAHASAN

Travelling Salesman Problem dengan Bahasa Pemrograman Pascal

Pemrograman yang dapat digunakan dalam menyelesaikan *Travelling Salesman Problem* pada aplikasi Pascal adalah sebagai berikut.

Function secondMin(adj : Node; var i : integer) : integer;

Var first, second, j : integer;

Begin

for j:= 0 To N-1 do

begin

if(i<>j) then

begin

if(adj[i,j]<=first) then

begin

second := first;

first := adj[i,j];

end

else if((adj[i,j]<=second) And (adj[i,j]<>first)) then

begin

second := adj[i,j];

end;

```
        end;
    end;
End;

Procedure showCost;
Const
    v : Vertex = (14,6,26,40,15,14,17,36,14,14);
Var
    i : ShortInt;
    gt: Integer;
Begin
    i :=1;
    gt:=0;
    write('Total Jarak','= ':4);
    Repeat
        gt := gt + v[i];
        if(i<N) then
            write(v[i],' + ')
        else
            write(v[i]);
            i := i + 1;
        Until(i>N);
        write(' = ',gt:2);
        writeLn;
    End;

Procedure TSPRec(adj : Node; current_bound : real; current_weight : integer;
    level : integer; current_path : array of integer);
Var
    current_res, i,j : integer;
    temp            : real;
    visited1        : array[0..10] of boolean;
Begin
    if(level=N-1) then
        begin
            if(adj[current_path[level-1],current_path[0]]<>0) then
                begin
                    current_res := current_weight + adj[current_path[level-1],current_path[0]];
                    if(current_res<final_res) then
                        begin
                            copyToFinal(current_path);
                            final_res := current_res;
                        end;
                    end;
                end;
            end;

        for i := 0 To N-1 do
            begin
                if((adj[current_path[level-1],i]<>0) And (visited=false)) then
```

```
begin
    temp := current_bound;
    current_weight := current_weight + adj[current_path[level-1],i];
    if(level=1) then
        current_bound := ((firstMin(adj, current_path[level-1])+
                           firstMin(adj, i))/2)
    else
        current_bound := ((secondMin(adj, current_path[level-1])+
                           secondMin(adj, i))/2);
    begin
        if(current_bound + current_weight < current_res) then
            begin
                current_path[level] := i;
                visited1[i] := true;
                TSPRec(adj, current_bound, current_weight, level+1, current_path);
            end;
            current_weight := adj[current_path[level-1],i];
            current_bound := temp;
            {memset(visited, false, sizeof(visited));}
            for j:=0 To level-1 do visited1[current_path[j]] := true;
        end;
    end;
end;

End;

Procedure TSP(adj : Node);
var
    current_path : array[0..N-1] of integer;
    visited      : array[0..N-1] of boolean;
    current_bound : real;
    i             : integer;
Begin
    current_bound :=0;
    {
        memset(current_path, -1, sizeof(current_path));
        memset(visited, 0, sizeof(current_path));
    }
    for i:=0 To N-1 do current_bound := current_bound + (firstMin(adj, i)+
                                                         secondMin(adj,i));
    if(current_bound=1) then
        current_bound := current_bound/2 + 1
    else
        current_bound := current_bound/2;
    visited[0] :=false;
    current_path[0] :=0;
    TSPRec(adj, current_bound, 0, 1, current_path);
End;
```



```
{Main Program}  
BEGIN  
    TSP(adj);  
    Header;  
    shortestPath;  
    writeLn;  
    showCost;  
    Footer;  
    ReadKey;  
END.
```

Kelebihan dari bahasa atau aplikasi pemrograman Pascal ini adalah bahwa struktur sintaks sederhana dan mudah dipahami, cocok untuk pembelajaran algoritma dasar; Penggunaan memori relatif rendah, yaitu sekitar 356 KB; Stabil dan mudah dalam proses *debugging* serta pelacakan logika algoritma; serta cocok untuk aplikasi kecil dan tujuan edukatif. Adapun kekurangan dari bahasa atau aplikasi pemrograman C++ adalah bahwa kecepatan eksekusi relatif lebih lambat dibandingkan C++; Dukungan terhadap pengembangan modern dan integrasi sistem terbatas; Kurang fleksibel untuk proyek skala besar atau berbasis objek; serta Lingkungan pengembangan sudah jarang digunakan di dunia industri.

Travelling Salesman Problem dengan Bahasa Pemrograman C++

Pemrograman yang dapat digunakan dalam menyelesaikan *Travelling Salesman Problem* pada aplikasi C++ adalah sebagai berikut.

```
// Program C ++ untuk memecahkan Masalah Travelling Salesman  
// Menggunakan Algoritma Branch and Bound.  
// Hitung batas awal  
    for (int i=0; i<N; i++)  
        curr_bound += (firstMin(adj, i) +  
                        secondMin(adj, i));  
// Membulatkan batas bawah menjadi bilangan bulat  
curr_bound = (curr_bound&1)? curr_bound/2 + 1 :  
                                                    curr_bound/2;  
// mulai dari simpul 1 jadi simpul pertama  
// dalam curr_path [] adalah 0  
visited[0] = true;  
curr_path[0] = 0;  
// Panggil ke TSPRec untuk curr_weight sama dengan 0 dan level 1  
TSPRec(adj, curr_bound, 0, 1, curr_path);  
}  
int main()  
{  
    // Matriks untuk grafik yang diberikan  
    int adj[N][N] =  
        { { 0, 33, 14, 24, 14, 26, 50, 0, 0, 0},  
          {33, 0, 14, 36, 0, 0, 0, 43, 22, 50},  
          {14, 14, 0, 29, 19, 0, 0, 31, 0, 0},  
          {24, 36, 29, 0, 24, 57, 0, 17, 0, 0},  
          {14, 0, 19, 24, 0, 6, 0, 70, 0, 0},  
          {26, 0, 0, 57, 6, 0, 26, 48, 42, 45},
```



```
{50, 0, 0, 0, 0, 26, 0, 38, 70, 40},  
{ 0, 43, 31, 17, 70, 48, 38, 0, 14, 31},  
{ 0, 22, 0, 0, 0, 42, 70, 14, 0, 15},  
{ 0, 50, 0, 0, 0, 45, 40, 31, 15, 0}  
  
};  
TSP(adj);  
cout << "===== PROGRAM OUTPUT  
=====\\n\\n" ;  
cout<< " Rute Optimal Pendistribusian Barang Simalungun Atas\\n\\n";  
printf( " Minimum cost\\t\\t : %d\\n\\n", final_res);  
printf( " Taken\\t\\t : ");  
for (int i=0; i<=N; i++)  
    printf("%d ", final_path[i]);  
return 0;  
}
```

Kelebihan dari aplikasi C++ adalah bahwa Kecepatan eksekusi sangat tinggi, yaitu sekitar 0,09466 detik dalam penelitian; Mendukung paradigma pemrograman berorientasi objek dan fitur modern; Dapat diintegrasikan dengan berbagai sistem dan teknologi terkini; serta Lebih cocok untuk pengembangan aplikasi kompleks atau skala besar. Adapun kekurangannya adalah bahwa Ukuran berkas output lebih besar, yaitu $\pm 1,26$ MiB; Sintaks lebih kompleks, membutuhkan pemahaman mendalam tentang konsep pemrograman; Penggunaan memori lebih tinggi dibandingkan Pascal; serta Proses *debugging* bisa lebih rumit pada proyek besar.

KESIMPULAN

Hasil penelitian menunjukkan bahwa algoritma *Travelling Salesman Problem* dapat diimplementasikan secara efektif menggunakan Pascal maupun C++ dengan hasil perhitungan rute yang identik, yaitu 1 – 5 – 6 – 7 – 10 – 9 – 8 – 4 – 2 – 3 – 1 dengan total jarak atau minimum cost 196. Kedua program berhasil dikompilasi tanpa error, membuktikan konsistensi dan akurasi algoritma. Pascal menonjol dalam kesederhanaan serta penggunaan memori yang rendah (sekitar 356 KB), sedangkan C++ menghasilkan berkas output lebih besar ($\pm 1,26$ MiB) namun mampu mengeksekusi program dengan sangat cepat (0,09466 detik). Perbedaan utama terletak pada lingkungan pengembangan dan efisiensi sumber daya: Pascal sesuai untuk pembelajaran dan aplikasi ringan, sedangkan C++ unggul dalam kecepatan eksekusi dan integrasi dengan sistem modern. Pemilihan bahasa pemrograman dapat disesuaikan dengan tujuan, baik untuk edukasi dan efisiensi memori maupun pengembangan aplikasi skala besar dan kebutuhan teknologi yang lebih luas. Algoritma *Travelling Salesman Problem* dapat diimplementasikan secara efektif baik dengan Pascal maupun C++. Pascal menawarkan keunggulan pada kesederhanaan dan pemakaian memori, sedangkan C++ memberikan keunggulan dalam kecepatan eksekusi dan dukungan pengembangan modern. Pemilihan bahasa pemrograman dapat disesuaikan dengan tujuan: edukasi dan efisiensi memori (Pascal) atau pengembangan aplikasi skala besar dan integrasi teknologi (C++).

DAFTAR PUSTAKA

A.S. Rosa. (2018). Struktur Data Terapan Dalam Berbagai Bahasa Pemrograman: Pascal, C, C++, dan Java. Penerbit Modula: Bandung.

- Albertus, W., Atje, S., Juli, R. (2016). Pengembangan Aplikasi *Travelling Salesman Problem* dengan Optimisasi *Robust* (Studi Kasus pada Pekan Paralimpiade Nasional XV 2016): Jurnal Informatika, Departemen Ilmu Komputer Fakultas MIPA Universitas Padjadjaran Jatinangor, Indonesia.
- Azwar, N. (2019). *Aplikasi Program Dinamik pada Traveling Salesman Problem (TSP)*: Program studi Matematika: Universitas Sumatera Utara.
- Dewi, Eka Poespita., Pradjanigsih, Agustina., Hasan, Muhammad. (2012). Optimasi Rute *Multiple Travelling Salesman Problem* melalui Pemrograman *Integer* Dengan Metode *Branch and Bound*. Jember: Universitas Jember.
- Hartono, Jogyanto (2000). *Konsep Dasar Pemrograman Bahasa C*. Yogyakarta: Penerbit Andi Yogyakarta.
- Kelvin, A. 2015. *Aplikasi Program Dinamis dalam Pemecahan TSP*, Teknik Informatika Institut Teknologi Bandung: Makalah IF2211 Strategi Algoritma, Semester II Tahun 2015/2016.
- Laksana, Janice. 2012. Penerapan Algoritma *Branch and Bound* untuk Penentuan Jalur Wisata. Bandung: Institut Teknologi Bandung.
- Munir, Rinaldi. 2010. Matematika Diskrit. Bandung: Informatika
- Nugraha, Pasca. 2010. Penerapan Algoritma *Branch and Bound* Dalam Menentukan Rute Terpendek Untuk Perjalanan Antarkota di Jawa Barat. Bandung: Institut Teknologi Bandung.
- Purwanto, Eko Budi. 2008. Perancangan & Analisis Algoritma. Yogyakarta: Graha Ilmu.
- Rohman, Saiful. 2020. Optimisasi *Travelling Salesman Problem* dengan Algoritma Genetika pada Kasus Pendistribusian Barang PT.POS Indonesia di Kota Bandar Lampung: Fakultas MIPA. Bandar Lampung
- Sirait, D. E. (2021). *Dasar pemrograman C++*. Perkumpulan Rumah Cemerlang Indonesia.
- Sirait, D. E. (2021). *Pemrograman Pascal*. Perkumpulan Rumah Cemerlang Indonesia.
- Sirait, D. E., & Simarmata, J. E. (2020). Penyelesaian masalah travelling salesman problem dengan menggunakan bahasa pemrograman Pascal. *MES: Journal of Mathematics Education and Science*, 6(1), 10–15.
- Yushi, N. (2011). Program Aplikasi *Travelling Salesman* Pada Pendistribusian Agen Minuman Ringan Menggunakan Turbo Pascal 7.1, Program Studi Teknik Informatika – Fakultas Teknik, Matematika dan IPA, Universitas Indraprasta PGRI: Jurnal Ilmiah Faktor Exacta Vol. 4 (1).