Management Science

Della Herli Rahman (16/394080/PA/17171)
Vincent Michael Sutanto (16/398531/PA/17492)

Transportation Management

- Problem
- Method
- Implementation
- Result

Problem

Orders

- Origin
- Destination
- Qty
- Weight

Minimum Cost

Problem

Method

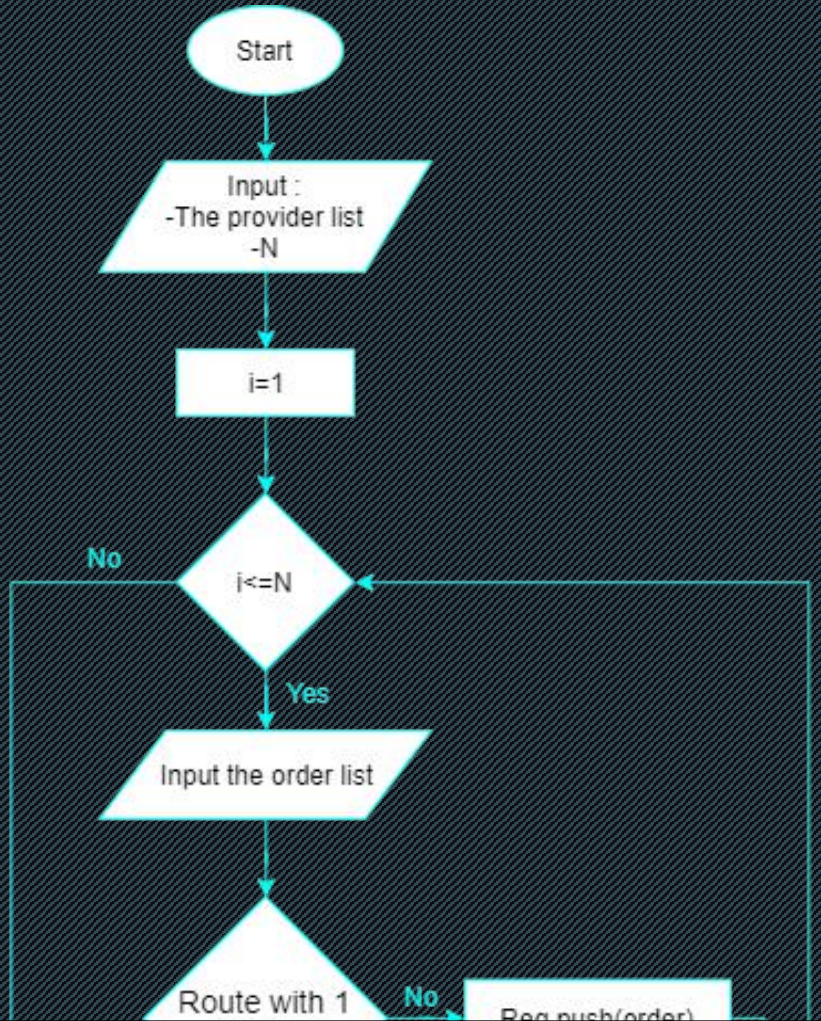
Implementation

Result

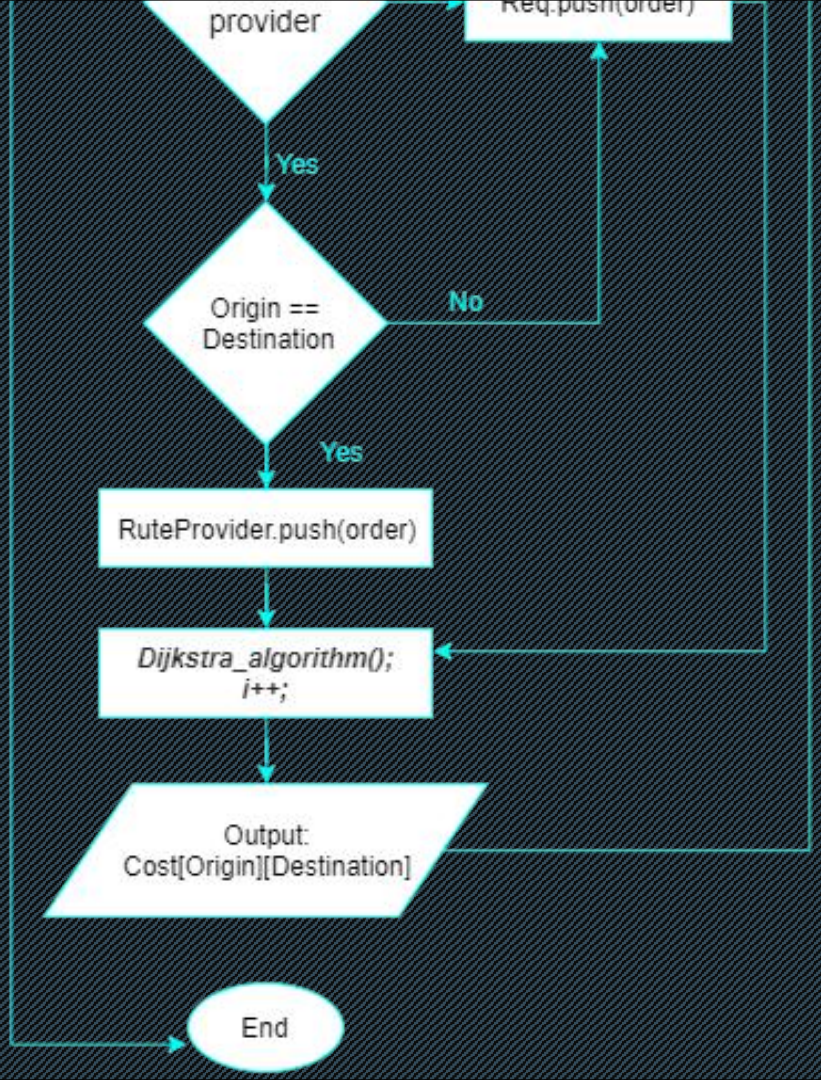
Method

N Shortest Path Tree

Method


Start
Input :
-The provider list
-N
i=1
No
i<=N
Yes
Input the order list
Route with 1
No

Method


provider
Req.push(order)
Yes
Origin ==
Destination
No
Yes
RuteProvider.push(order)
Dijkstra_algorithm();
i++;
Output:
Cost[Origin][Destination]
End

Method

Input

- Provider
- Order

Assign

- Data[15][15] , 7 providers + 8 cities
- Data[i][j]= cost city i using provider j

Preprocessing

- Filter Order
- Sorting Orders

Process /order

- Dijkstra's algorithm
- Dist[15]

Finding minimum cost by route

- Update capacity and request

Output /order

- cost * weight

Penjelasan Dijkstra's Algorithm

Dijkstra Algorithm digunakan untuk mengatasi permasalahan request yang harus dikirimkan dengan lebih dari 1 provider

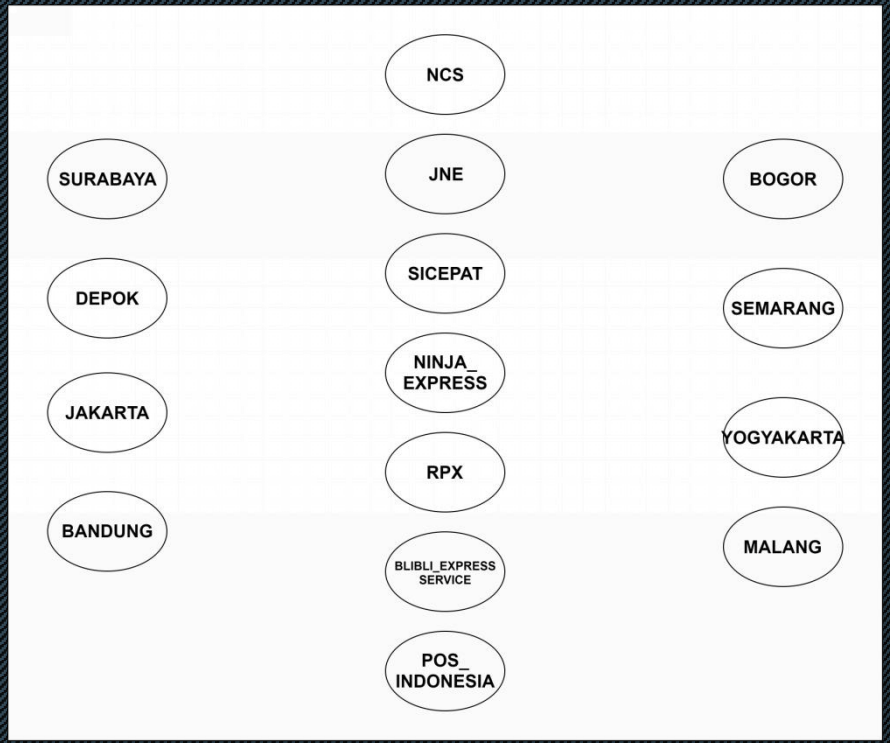
Penjelasan Dijkstra's Algorithm

Provider	Max Capacity / Day	Max Request / Day	Price / Kg	Range
NCS	65	150	11000	Surabaya, Semarang
JNE	630	3500	12000	Jakarta, Bandung, Yogyakarta, Semarang
SICEPAT	100	600	13000	Depok, Surabaya, Bogor, Yogyakarta
NINJA_EXPRESS	250	1350	13000	Bogor, Jakarta, Bandung
RPX	100	150	14500	Bandung, Surabaya, Malang
BLIBLI_EXPRESS_SERVICE	375	800	15000	Semarang, Jakarta, Bandung
POS_INDONESIA	320	1470	15000	Jakarta, Depok, Semarang, Yogyakarta

Diubah menjadi Graf

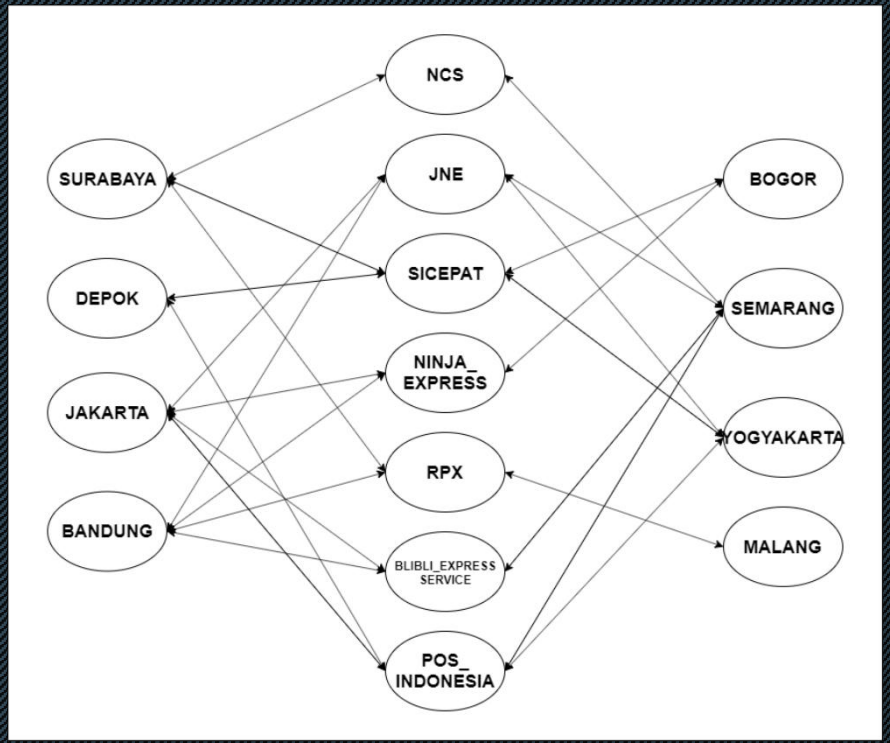
- Vertex : nama provider dan kota yang dilayani
- Edge : hubungan provider dengan kota yang dilayaninya

Penjelasan Dijkstra's Algorithm



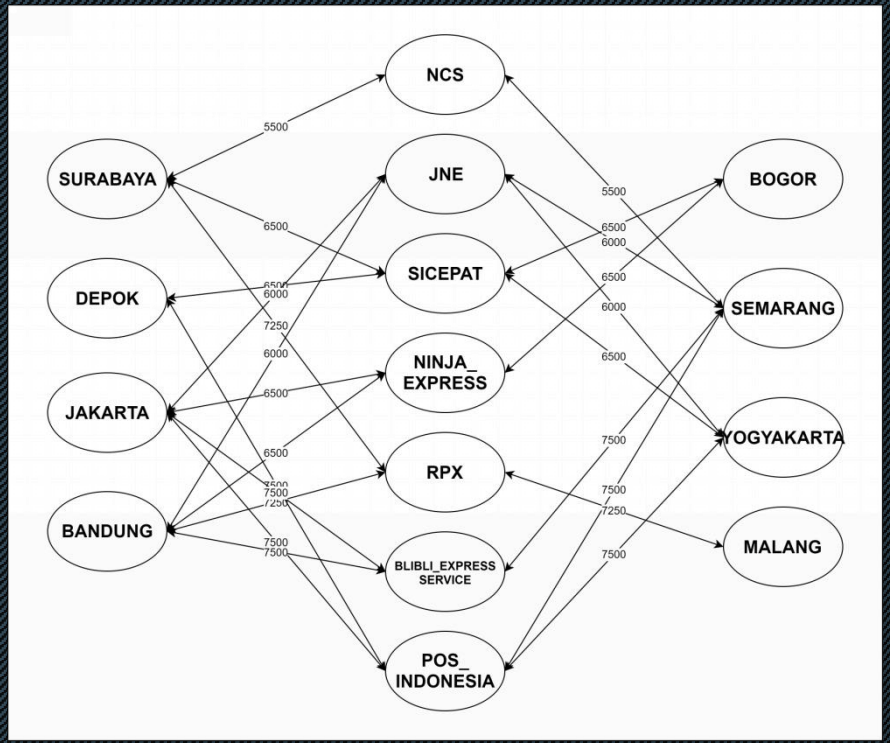
Provider	Max Capacity / Day
NCS	65
JNE	630
SICEPAT	100
NINJA_EXPRESS	250
RPX	100
BLIBLI_EXPRESS_SERVICE	375
POS_INDONESIA	320

Penjelasan Dijkstra's Algorithm



Provider	Max Capacity / Day
NCS	65
JNE	630
SICEPAT	100
NINJA_EXPRESS	250
RPX	100
BLIBLI_EXPRESS_SERVICE	375
POS_INDONESIA	320

Penjelasan Dijkstra's Algorithm



Provider	Max Capacity / Day
NCS	65
JNE	630
SICEPAT	100
NINJA_EXPRESS	250
RPX	100
BLIBLI_EXPRESS_SERVICE	375
POS_INDONESIA	320

Vertex

- Cost
- Route

Edge

- Cost
- Capacity
- Request

Bandung Jakarta 3 65.03

Source	Cost	Route
NCS	-	
JNE	6000	Bandung
SICEPAT	-	
NINJA EXPRESS	6500	Bandung
RPX	7250	Bandung
BLIBLI EXPRESS	7500	Bandung
POS INDONESIA	-	
Jakarta		
Bandung	0	-
Yogyakarta		
Semarang		
Bogor		
Depok		
Surabaya		
Malang		

Source	Cost	Previous Route
NCS	-	
JNE	6000	Bandung
SICEPAT	-	
NINJA EXPRESS	6500	Bandung
RPX	7250	Bandung
BLIBLI EXPRESS	7500	Bandung
POS INDONESIA	-	
Jakarta	12000	Bandung, JNE
Bandung	0	-
Yogyakarta	12000	Bandung, JNE
Semarang	12000	Bandung, JNE
Bogor	13000	Bandung, Ninja Express
Depok		
Surabaya	14500	Bandung, RPX
Malang	14500	Bandung, RPX

Source	Cost	Previous Route
NCS	17500	Bandung, JNE, Semarang
JNE	6000	Bandung
SICEPAT	18500	Bandung, JNE, Yogyakarta
NINJA EXPRESS	6500	Bandung
RPX	7250	Bandung
BLIBLI EXPRESS	7500	Bandung
POS INDONESIA	19500	Bandung, JNE, Semarang
Jakarta	12000	Bandung, JNE
Bandung	0	-
Yogyakarta	12000	Bandung, JNE
Semarang	12000	Bandung, JNE
Bogor	13000	Bandung, Ninja Express
Depok	25000	Bandung, JNE, Yogyakarta, Sicepat
Surabaya	14500	Bandung, RPX
Malang	14500	Bandung, RPX

Origin : Bandung
Destination : Jakarta
Qty : 3
Weight : 65.03
Cost = Dist["Jakarta"] * weight
= 12000 * 65.03
= 780.360
Route = Bandung JNE Jakarta

How if destination is Depok?

Cost = Dist["Depok"] * weight
= 25000 * 65.03
= 162.575
Route = Bandung JNE Yogyakarta
Yogyakarta Sicepat Depok

Problem

Method

Implementation

Result

Implementation : C++

ManagementScience.cpp

```cpp
#include<bits/stdc++.h>
using namespace std;

struct Express{
    string nama;
    float max_capacity;
    int max_req;
    double cost;
    map<string, int> city;
};

struct Order{
    string asal;
    string tujuan;
    int qty;
    double berat, biaya;
};

struct Rute{
    vector<int> alur;
    int cost;
};

string S[9]= {"Jakarta", "Bandung","Yogyakarta","Semarang","Bogor","Depok","Surabaya","Malang"};
string P[8] = {"NCS", "JNE", "SICEPAT", "NINJA EXPRESS", "RPX", "BLIBLI EXPRESS SERVICE", "POS INDONESIA" };
Express data[15][15];
Express Provider[10];
Order Requ;
```

Implementation : https://github.com/dellahera/Transportation-Management/

```
Rute Orderan(Order X){
    for(long long j=0; j<8; j++){
        if(S[j]==X.asal) awal = j+7;
        if(S[j]==X.tujuan)  akhir= j+7;}

    Order temp;
    temp= X;
    Rute dist[15];
    bool flag[15];
    //Rute dengan kota yang sama
    if(awal == akhir){
        dist[awal].alur.push_back(awal);
        for(int i=0; i<7; i++){
            if( Provider[i].city[temp.asal]==1 && temp.qty <= Provider[i].max_capacity && Provider[i].max_req>0){
                dist[awal].alur.push_back(i);
                dist[awal].cost = data[i][awal].cost* 2;
                break;
            }
        }
    }
    else { //Rute yang melewati provider berbeda
        for (int i = 0; i < 15; i++)
        dist[i].cost = INT_MAX, flag[i] = false;

        dist[awal].cost = 0;

        for (int i = 0; i < 14; i++) {
            int u = minDist(dist, flag);
            flag[u] = true;
            for (int j = 0; j < 15; j++) {
                if (!flag[j] && data[u][j].cost>0 && dist[u].cost != INT_MAX
                    && dist[u].cost + data[u][j].cost < dist[j].cost && temp.qty <= data[u][j].max_capacity && data[u][j].max_req >0) {
                        dist[j].cost = dist[u].cost + data[u][j].cost;
                        dist[j].alur.push_back(u);
                        dist[j].alur.insert(dist[j].alur.begin(), dist[u].alur.begin(), dist[u].alur.end());
                }
            }
        }
    }
    return dist[akhir];
}
```

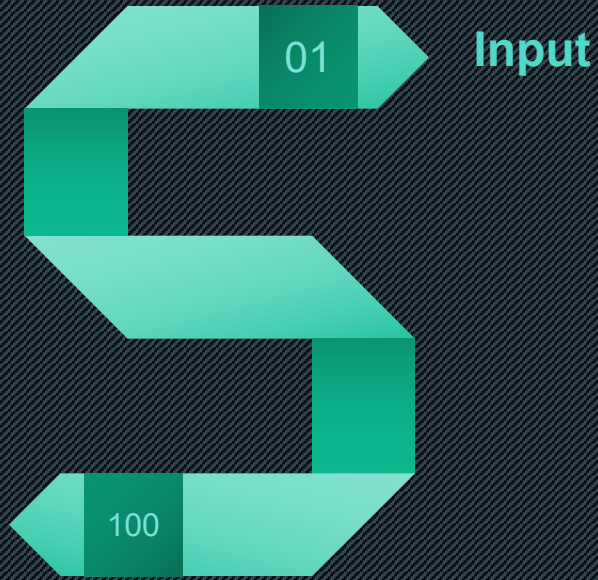
Problem

Method

Implementation

Result

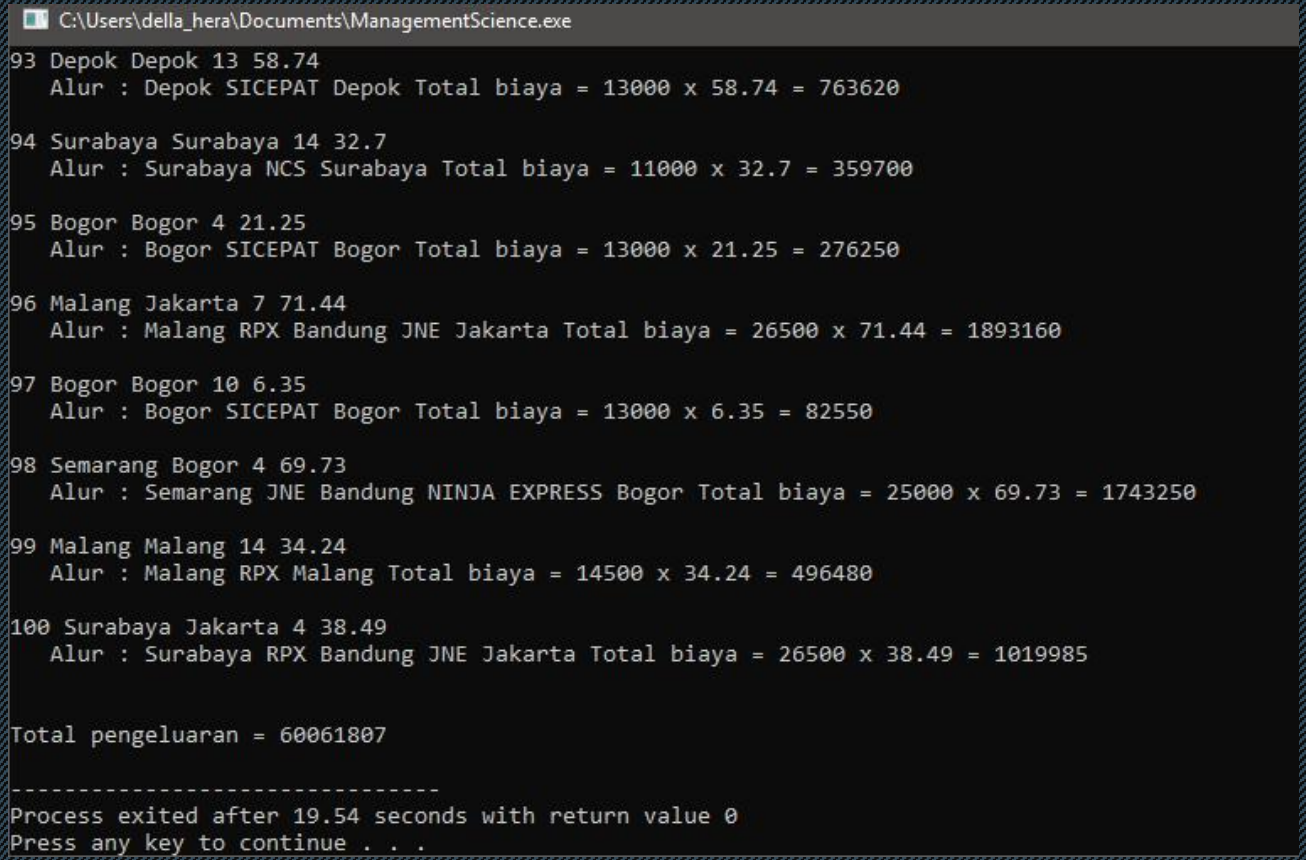
Result



Total Cost:

Rp. 60.061.807, 00

Result


C:\Users\della_hera\Documents\ManagementScience.exe
93 Depok Depok 13 58.74
Alur : Depok SICEPAT Depok Total biaya = 13000 x 58.74 = 763620
94 Surabaya Surabaya 14 32.7
Alur : Surabaya NCS Surabaya Total biaya = 11000 x 32.7 = 359700
95 Bogor Bogor 4 21.25
Alur : Bogor SICEPAT Bogor Total biaya = 13000 x 21.25 = 276250
96 Malang Jakarta 7 71.44
Alur : Malang RPX Bandung JNE Jakarta Total biaya = 26500 x 71.44 = 1893160
97 Bogor Bogor 10 6.35
Alur : Bogor SICEPAT Bogor Total biaya = 13000 x 6.35 = 82550
98 Semarang Bogor 4 69.73
Alur : Semarang JNE Bandung NINJA EXPRESS Bogor Total biaya = 25000 x 69.73 = 1743250
99 Malang Malang 14 34.24
Alur : Malang RPX Malang Total biaya = 14500 x 34.24 = 496480
100 Surabaya Jakarta 4 38.49
Alur : Surabaya RPX Bandung JNE Jakarta Total biaya = 26500 x 38.49 = 1019985
Total pengeluaran = 60061807
Process exited after 19.54 seconds with return value 0
Press any key to continue . . .

Thank you