TUGAS BESAR ALGORITMA DAN PEMROGRAMAN
EXHAUSTIVE SEARCH

Nama Kelompok:

1. Anak Agung Indi Kusuma Putra (2205551079)
2. Muhammad Ibrahim (2205551083)

Program Studi Teknologi Informasi

Fakultas Teknik

Universitas Udayana

Tahun 2022

1. Penjelasan Algoritma Exhaustive Search

Algoritma *exhaustive search* merupakan algoritma yang menggunakan algoritma *brute force* untuk menyelesaikan permasalahannya. Algorima *exhaustive search* dilakukan dengan mencoba semua kemungkinan yang ada, sehingga dapat ditemukan solusi yang terbaik. Walaupun dapat menemukan solusi yang terbaik, algoritma ini memiliki kelemahan seperti kompleksitas waktu yang besar, sehingga tidak dapat menyelesaikan masalah dalam jumlah besar. Permasalahan yang dapat diselesaikan oleh algoritma ini adalah permasalahan kombinatorik, seperti permasalahan permutasi, kombinasi, dan himpunan bagian dari sebuah himpunan.
Langkah-langkahnya metode exhaustive search sebagai berikut:

1. Enumerasi (list) setiap solusi yang mungkin dengan cara yang sistematis.
2. Evaluasi setiap kemungkinan solusi satu per satu, beberapa kemungkinan solusi yang tidak layak dikeluarkan, dan simpan solusi terbaik yang ditemukan sampai sejauh ini (the best solusi found so far).
3. Bila pencarian berakhir, umumkan solusi terbaik.

2. Contoh masalah yang dapat di selesaikan dengan Exhaustive Search

Ada beberapa masalah yang dapat diselesaikan dengan algoritma exhaustive search. Diantara *seperti traveling salesman problem,* dan *assignment problem.*

2.1 Traveling Salesman Problem (TSP)

Traveling salesman problem (TSP) merupakan salah satu contoh masalah yang dapat diselesaikan dengan algoritma *exhaaustive search*. Singkatnya, TSP adalah masalah untuk mencari jalur yang paling efektif diantara semua jalur yang mungkin dilewati.

2.2 Assignment Problem

Assignment problem yaitu masalah dimana terdapat sejumlah n orang yang perlu di berikan pekerjaan yang sebanyak n pekerjaan. Dengan syarat satu orang mengambil satu pekerjaan, yang berarti setiap orang mendapat satu pekerjaan dan setiap pekerjaan dikerjakan oleh tepat satu orang.

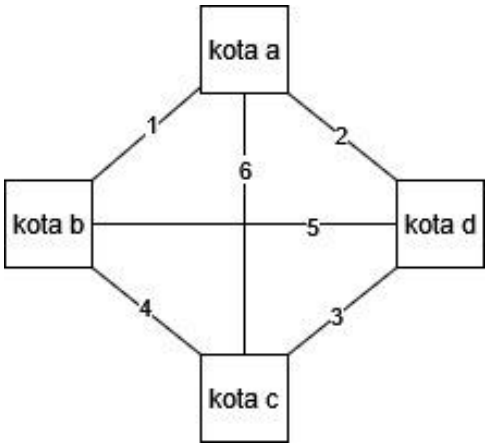
3. Solusi Manual

3.1 By Hand

3.1.1 Traveling Salesman Problem

Solusi algoritma untuk masalah TSP (*Travelling Salesman Problem*) menggunakan metode exhaustive yang menggunakan pendekatan bruteforce untuk mencari kemungkinan jalur terbaik dan terefektif yang bisa ditempuh. Sehingga akan memakan waktu yang cukup lama jika jumlah kota yang akan dikunjungi banyak. Yang pertama dilakukan adalah untuk menentukan banyak kota yang akan disinggahi, setelah itu menghitung permutasi dan jalur jalur yang mungkin untuk ditempuh, setelah itu baru menghitung jumlah jarak dari masing masing kemungkinan, setelah semua kemungkinan sudah ditemukan jaraknya maka akan dicari jalur dengan jarak terpendek yang akan menjadi jalur yang akan dilalui.

Misal kita ambil contoh Tsp problem dengan kasus 4 kota, lalu kita inputkan jarak dari masing" kota dan tentukan kota awal, kita cari mutasinya menggunakan rumus permutasi siklik sehingga didapat 6 kemungkinan jalur, setelah itu kita menggunakan pendekatan brute-force untuk mencari kemungkinan jalur dan menghitung jarak total dari jalur tersebut, setelah didapat semua kemungkinan maka kita cari jalur dengan jarak terpendek dan menjadikannya jalur terefektif untuk dilalui.



jarak antar kota

a > b = 1

a > c = 6

a > d = 2

b > c = 4

b > d = 5

c > d = 3

jalur yang mungkin = 3! = 6 jalur

Jalur yang mungkin serta Jarak totalnya

a > b > c > d > a

1 + 4 + 3 + 2 = 10

a > b > d > c > a

1 + 5 + 3 + 6 = 15

a > c > b > d > a

6 + 4 + 5 + 2= 17

a > c > d > b > a

6 + 3 + 5 + 1 = 15

a > d > c > b > a

2 + 3 + 4 + 1 = 10

a > d > b > c > a

2 + 5 + 4 + 6 = 17

Jalur paling efektif :

a > b > c > d > a

a > d > c > b > a

Dengan Jarak Total = 10

3.1.2 Assignment Cost

Solusi manual untuk masalah *Assignment Cost Problem* Membuat tabel yang berisi beban yang diperlukan seorang pekerja melakukan suatu pekerjaan. Dimana tabel tersebut dibuat dengan matriks dan sebesar n×n. Setelah diberi nilai berupa biaya atau beban pada seluruh indeks matriks tersebut, maka akan dijumlahkan indeks tersebut dengan syarat, tiap tiap indeks tidak ada indeks yang memiliki koordinat yang sama dengan indeks lain dalam satu penjumlahan. Setelah seluruh penjumlahan yang mungkin dilakukan selesai, maka akan dicari hasil yang paling kecil.

Misal kita mencari biaya terkecil dari 3 pekerja dan 3 pekerjaan. Dimana pekerja 1 memiliki biaya sebagai berikut, 6 untuk pekerjaan 1, 7 untuk pekerjaan 2, dan 4 untuk pekerjaan 3. Pekerja 2 memiliki biaya sebagai berikut, 8 untuk

pekerjaan 1, 9 untuk pekerjaan 2, dan 3 untuk pekerjaan 3. Pekerja 3 memiliki biaya sebagai berikut, 4 untuk pekerjaan 1, 5 untuk pekerjaan 2, dan 1 untuk pekerjaan 3. Setelah mengikuti algoritma *exhaustive search* untuk masalah ditemukan biaya terkecil adalah 14. Dimana 14 tersebut didapat dari 2 cara cara pertama yaitu pekerjaan 1 dikerjakan pekerja 1, pekerjaan 2 dikerjakan pekerja 3, dan pekerjaan 3 dikerjakan pekerja 2. Cara kedua yaitu pekerjaan 1 dikerjakan pekerja 3, pekerjaan 2 dikerjakan pekerja 1, dan pekerjaan 3 dikerjakan pekerja 2.

Pekerja	pekerjaan		
	1	2	3
1	6	7	4
2	8	9	3
3	4	5	1

Kombinasi yang mungkin terjadi 3!=6

Pekerjaan 1	Pekerjaan 2	Pekerjaan 3	
pekerja 1	Pekerja 2	pekerja 3	biaya : 16
pekerja 1	pekerja 3	pekerja 2	biaya : 14
pekerja 2	pekerja 1	pekerja 3	biaya : 16
pekerja 2	pekerja 3	pekerja 1	biaya : 17
pekerja 3	pekerja 2	pekerja 1	biaya : 17
pekerja 3	pekerja 1	pekerja 2	biaya : 14

Biaya minimum untuk menyelesaikan seluruh pekerjaan : 14

3.2 Pseudocode

3.2.1 Traveling Salesman Problem

```
TRAVELING SALESMAN PROBLEM

PROCEDURE swap() : void
PROCEDURE assign() : void
PROCEDURE jarak() : void
PROCEDURE menghitungJarak() : void
FUNCTION mutasi() : int
PROCEDURE jalurFinal() : void
PROCEDURE namaKelompok() : void

str[],matriks[100][100],              nama[26]              =
"abcdefghijklnopqrstuvwxyz" : char
kota,  hasil[1000000],x=1,  hasilMutasi,  ascii[],  kecil=1,
finalRute=1 : int

STRUCTTUER terpendek : struct
    jalur[1000] : char

terpendek rute[1000] : struct

PROCEDURE swap(char *x, char *y)
DEKLARASI :
    temp : char
DESKRIPSI :
BEGIN
    temp = *x
    *x = *y
    *y = temp
END

FUNCTION permute(char *a, int l, int r, char kotaAwal) : char
DESKRIPSI:
BEGIN
    IF(l == r  && a[0] == kotaAwal ) THEN
```

```
            a[kota] = a[0]
            x++
            CALL menghitungJarak(a)
        ELSE
             FOR (int i = l i <= r i++)
                   CALL swap((a + l), (a + i))
                   CALL permute(a, l + 1, r, kotaAwal)
                   CALL swap((a + l), (a + i))
             ENDFOR
    END

    FUNCTION main() : int
    DEKLARASI :
        hasilMutasi = 1, n : int
        kotaAwal : char
    DESKRIPSI :
    BEGIN
        CALL namaKelompok()
        WRITE "Masukan jumlah kota :"
        READ kota
        CALL assign()
        n = CALL strlen(str)
        CALL jarak()
        hasilMutasi = CALL mutasi(hasilMutasi)
        CALL getchar()
        WRITE "Masukan kota awal :"
        READ kotaAwal
        CALL system("cls")
        WRITE "List jalur :"
        CALL permute(str,0,n-1,kotaAwal)
        CALL jalurFinal()
        return 0
    END

    PROCEDURE assign(){
    DESKRIPSI :
    BEGIN
```

```
    WRITE "List kota :"
    FOR( int i = 0 i < kota i++)
        str[i] = nama[i]
        WRITE "str[i]"
    ENDFOR
END

PROCEDURE jarak() : void
DESKRIPSI :
BEGIN
    FOR ( int i = 1 i <= kota i++)
        FOR (int j = 1 j <= kota j++)
            IF(i == j) THEN
                matriks[i][j] = 0
            ELSE IF( j < i) THEN
                matriks[i][j] = matriks [j][i]
            ELSE
                WRITE "Masukan jarak nama[i-1] ke nama[j-1] :
"
                READ matriks[i][j]
        ENDFOR
    ENDFOR
END

PROCEDURE menghitungJarak(char *a) : void
DEKLARASI :
    totalSementara : int
DESKRIPSI :
BEGIN
    WRITE "Jalur ke-%d : \n", x-1
    FOR(int i = 0 i <= kota  i++)
        ascii[i] = (int)a[i] - 96
        IF(i == kota) THEN
            WRITE "a[i]"
        ELSE IF(i == 0) THEN
            WRITE "a[i] > "
        ELSE
```

```
            WRITE "a[i] > "
        ENDFOR
        WRITE "\n"
        WRITE "Total jarak ="
        FOR (int i = 1 i <= kota  i++)
            IF (i == kota) THEN
                WRITE "matriks[ascii[i-1]][ascii[i]]"
            ELSE
                WRITE "matriks[ascii[i-1]][ascii[i]])"
            totalSementara += matriks[ascii[i-1]][ascii[i]]
        ENDFOR
        hasil[x-1] = totalSementara
        WRITE " = hasil[x-1]"
        IF(kecil > hasil[x-1]) THEN
            finalRute = 1
            kecil = hasil[x-1]
            FOR(int i=0i <= kota i++)
                rute[finalRute].jalur[i] = str[i]
            ENDFOR
        ELSE IF(kecil == hasil[x-1] || kecil == 1) THEN
            IF(kecil == hasil[x-1]) THEN
                finalRute++
            FOR(int i=0i <= kota i++)
                kecil = hasil[x-1]
                rute[finalRute].jalur[i] = str[i]
            ENDFOR
    END

    FUNCTION mutasi(int hasilMutasi) : int
    DESKRIPSI :
    BEGIN
        FOR ( int i = kota-1 i > 0 i--)
            hasilMutasi *= i
        ENDFOR
        WRITE "Total jalur yang mungkin = hasilMutasi jalur"
        return hasilMutasi
    END
```

```
PROCEDURE jalurFinal() : void
DESKRIPSI :
BEGIN
    WRITE "Jalur yang paling efektIF ada %d jalur\n\t\tYaitu
: \n",finalRute
    FOR(int i = 1 i <= finalRute i++)
        WRITE "i"
        FOR(int j = 0 j <= kota j++ )
            IF(j==kota) THEN
                WRITE "rute[i].jalur[j]"
            ELSE
                WRITE "rute[i].jalur[j]"
        ENDFOR
    ENDFOR
    WRITE "Dengan jarak = kecil"
    CALL system("pause")
END
```

3.2.2 Assignment Cost Problem

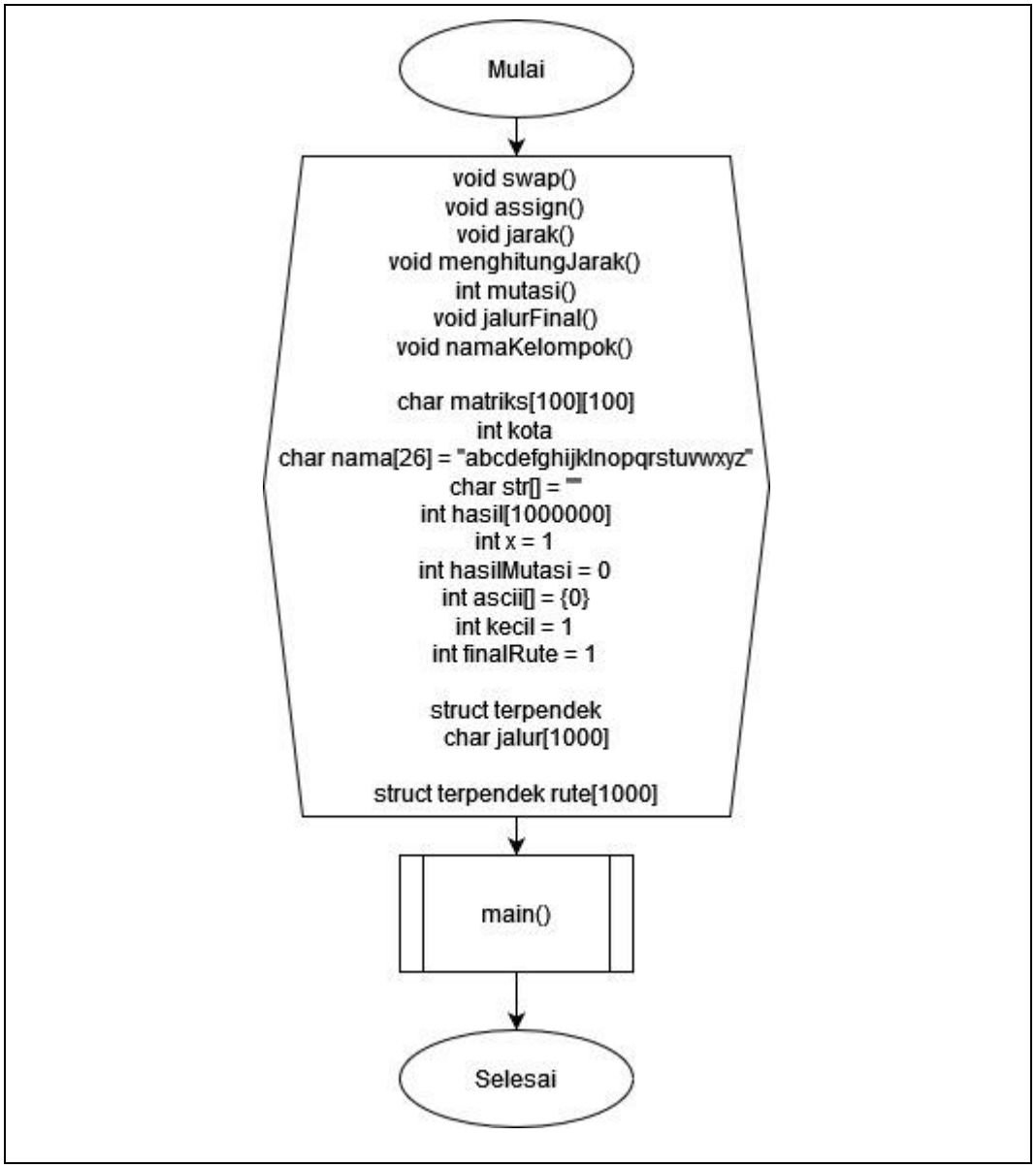
```
ASSIGN COST

DEKLARASI
int min=99999

FUNCTION tukar()
FUNCTION permutasi()
PROCEDURE awal()

FUNCTION main()
DEKLARASI :
l = 0, i, j, panjang, a[panjang], int arr[panjang][panjang], m,
n : INTEGER
l = 0
m = n =panjang
BEGIN
CALL awal()
WRITE("Masukkan jumlah pekerja dan pekerjaan : ")
READ("%d", &panjang)

FOR(i = 0 i < panjang i++)
a[i] = i
END FOR
FOR(i = 0 i < panjang i++)
```

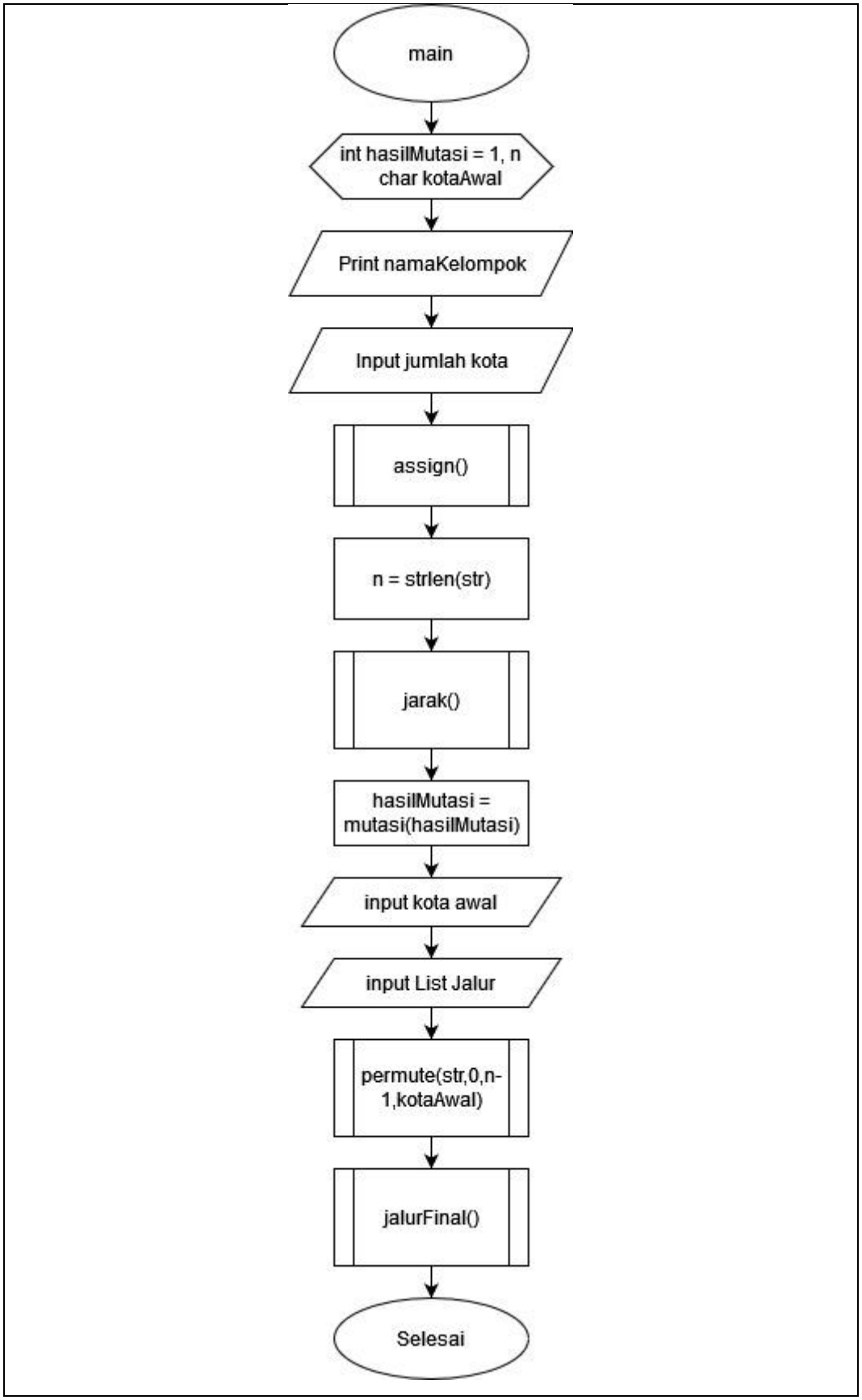
```
      FOR(j = 0 j < panjang j++)
            WRITE("Masukkan biaya dari pekerja %d pekerjaan %d:
",i+1, j+1)
            READ("%d", &arr[i][j])
      END FOR
END FOR
FOR(i=0i<panjangi++)
      WRITE("pekerja %d  ->  ", i+1)
      FOR(j = 0 j < panjang j++)
            WRITE("%d\t", arr[i][j])
      ENDFOR
END FOR
CALL permutasi(m, n, arr, a, l, panjang)
WRITE("\n\nBiaya minimum untuk menyelesaikan seluruh pekerjaan :
%d\n\n",min)
END


PROCEDURE awal()
BEGIN
WRITE("  Program Solusi Assignment Cost Problem  \n")
WRITE("    dengan Algoritma Exhaustive Search    \n")
END

FUNCTION tukar(int a[],int l,int i)
DEKLARASI :
char temp
BEGIN
temp=a[l]
a[l]=a[i]
a[i]=temp
END

void permutasi(int m,int n,int arr[][n],int a[],int l,int
panjang)
DEKLARASI
I, b[panjang], j, jumlah, val : INTEGER
Jumlah = 0
BEGIN
IF(l==panjang)
FOR(j=0j<panjangj++)
            b[j]=a[j]
      END FOR
      FOR(i=0i<panjangi++)
            val=b[i]
            jumlah+=arr[val][i]
      END FOR
      IF(min >= jumlah)
            min = jumlah
            FOR(j = 0 j<panjang j++)
                  WRITE("pekerja %d -> ",b[j]+1)
            END FOR
            WRITE("\tbiaya : %d\n",jumlah)
      END IF
END IF
ELSE
```

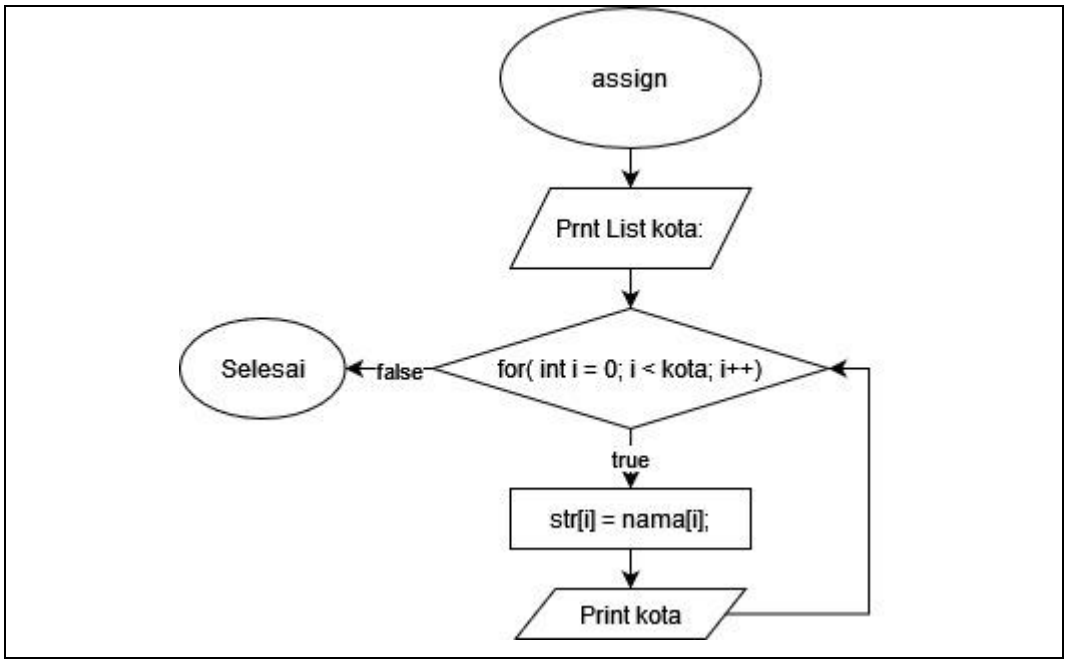
```
FOR(i=li<panjangi++)
            CALL tukar(a,l,i)
            CALL permutasi(m,n,arr,a,l+1,panjang)
            CALL tukar(a,l,i)
      END FOR
END ELSE
END
```

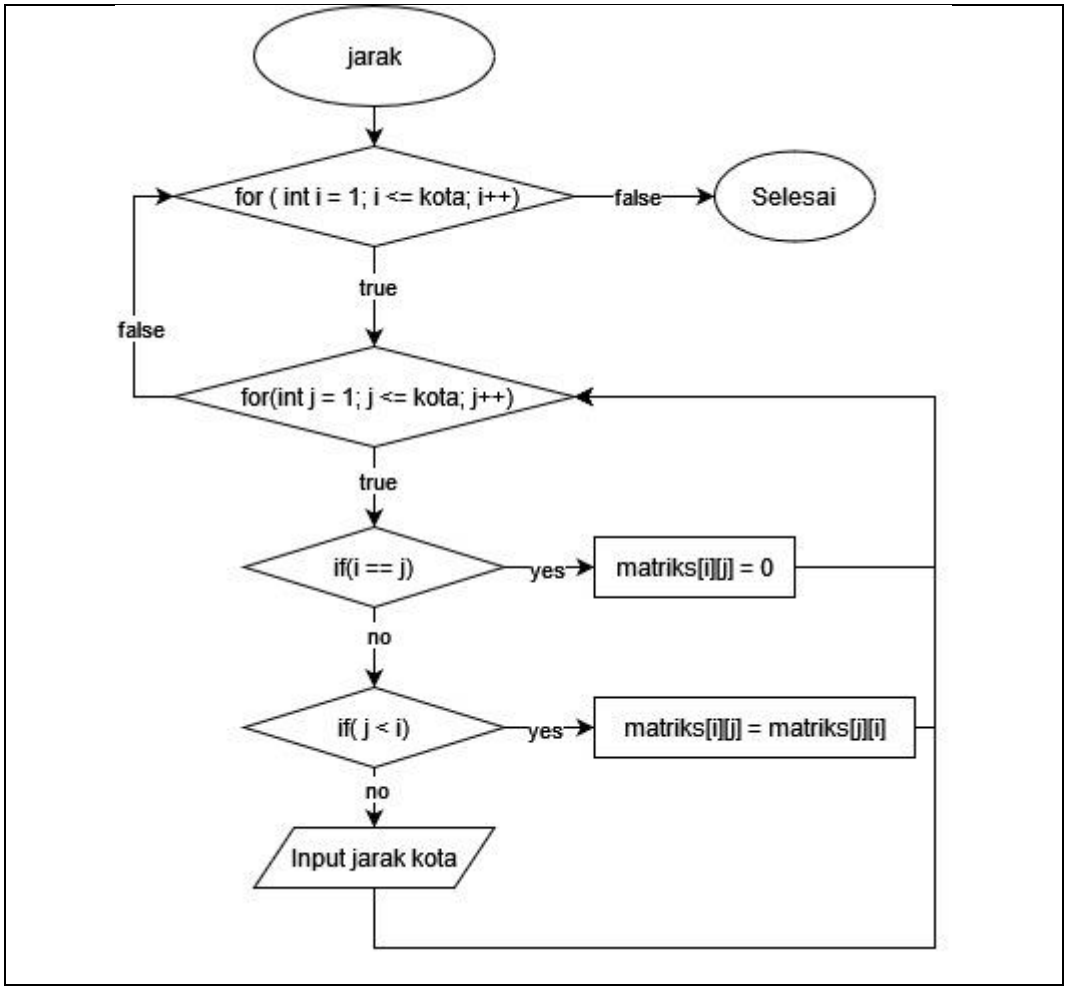
3.3 Flowchart

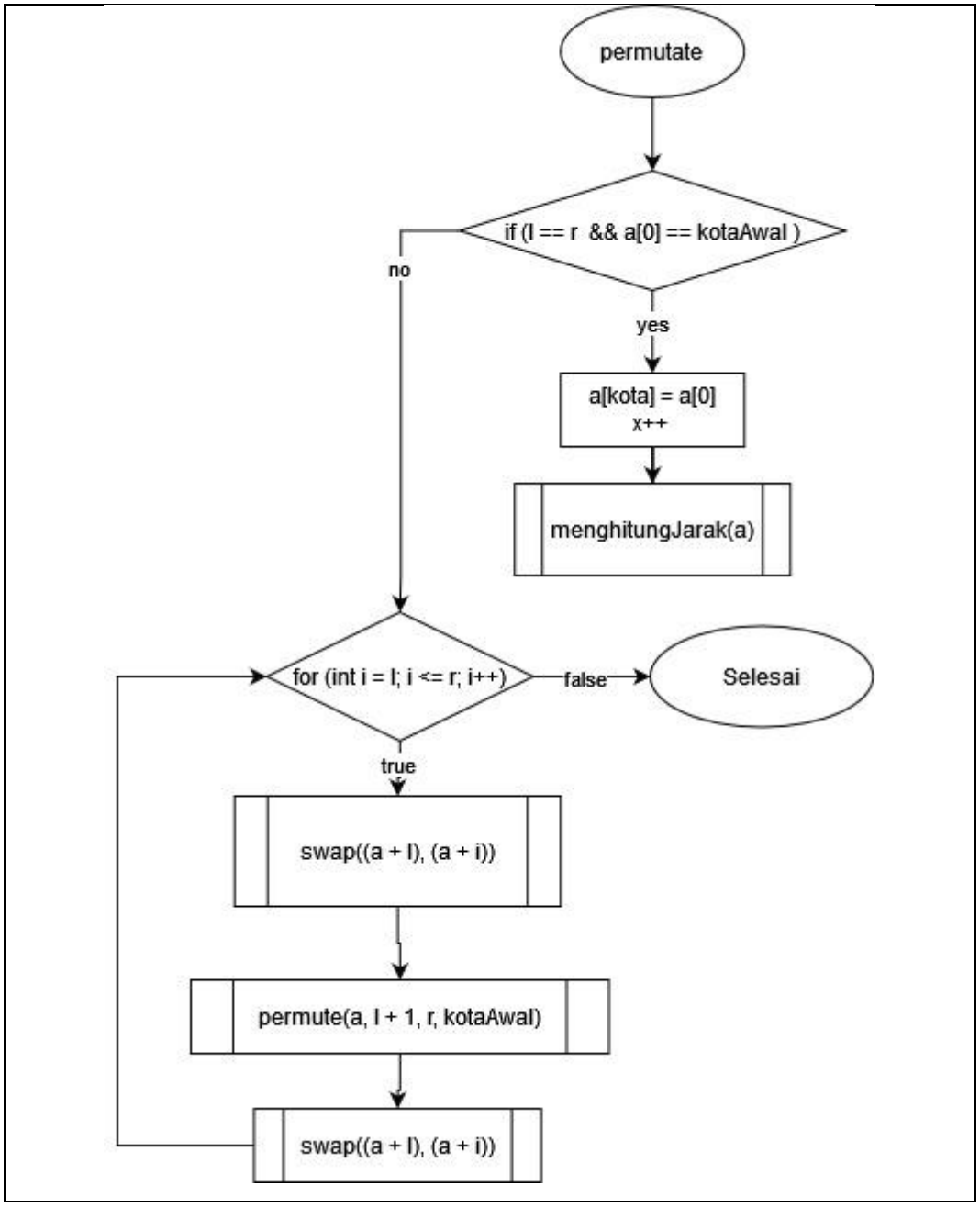
3.3.1 Flowchart Traveling Salesman Problem

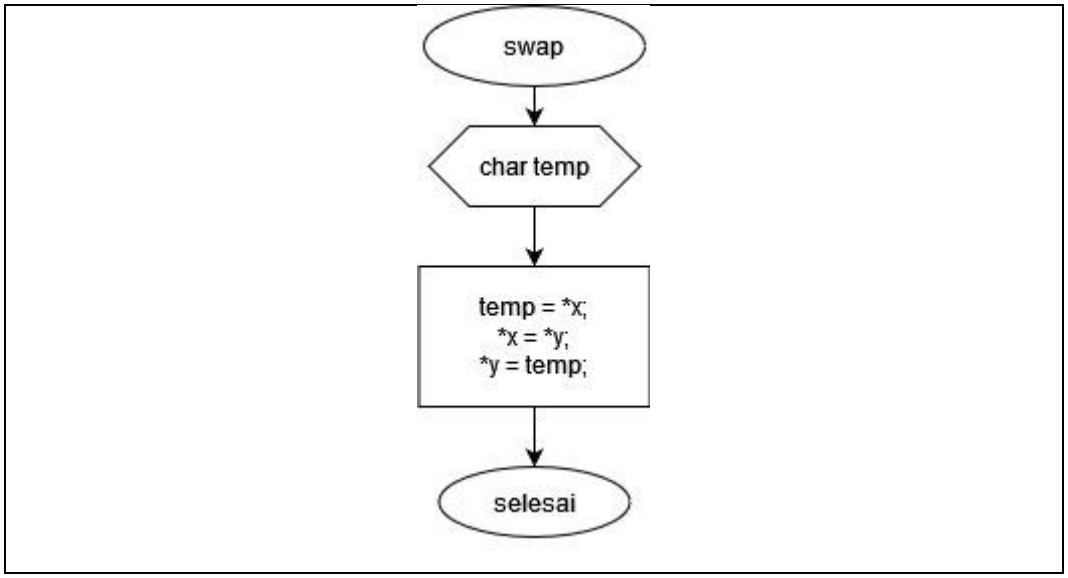


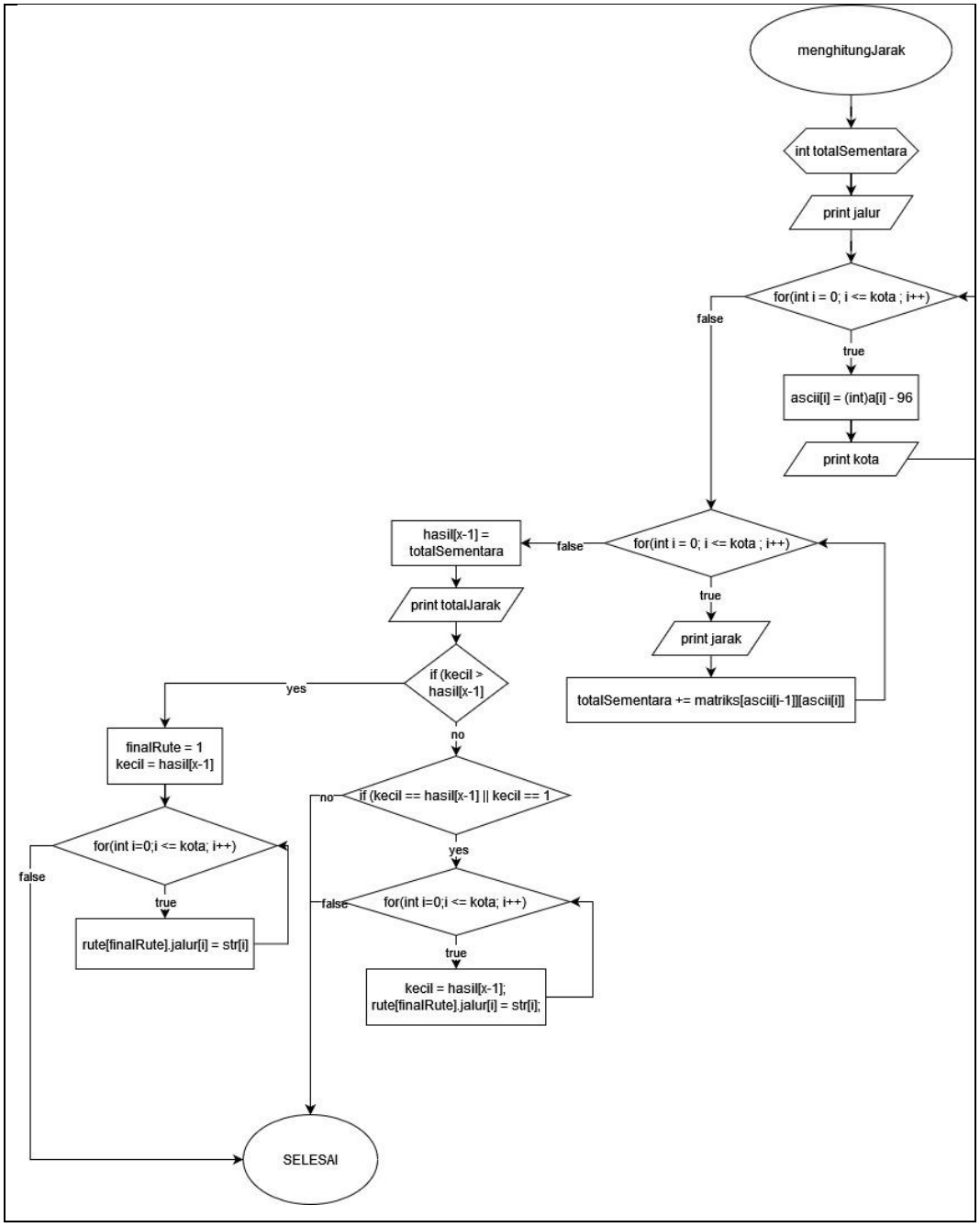

main
int hasilMutasi = 1, n
char kotaAwal
Print namaKelompok
Input jumlah kota
assign()
n = strlen(str)
jarak()
hasilMutasi =
mutasi(hasilMutasi)
input kota awal
input List Jalur
permute(str,0,n-
1,kotaAwal)
jalurFinal()
Selesai

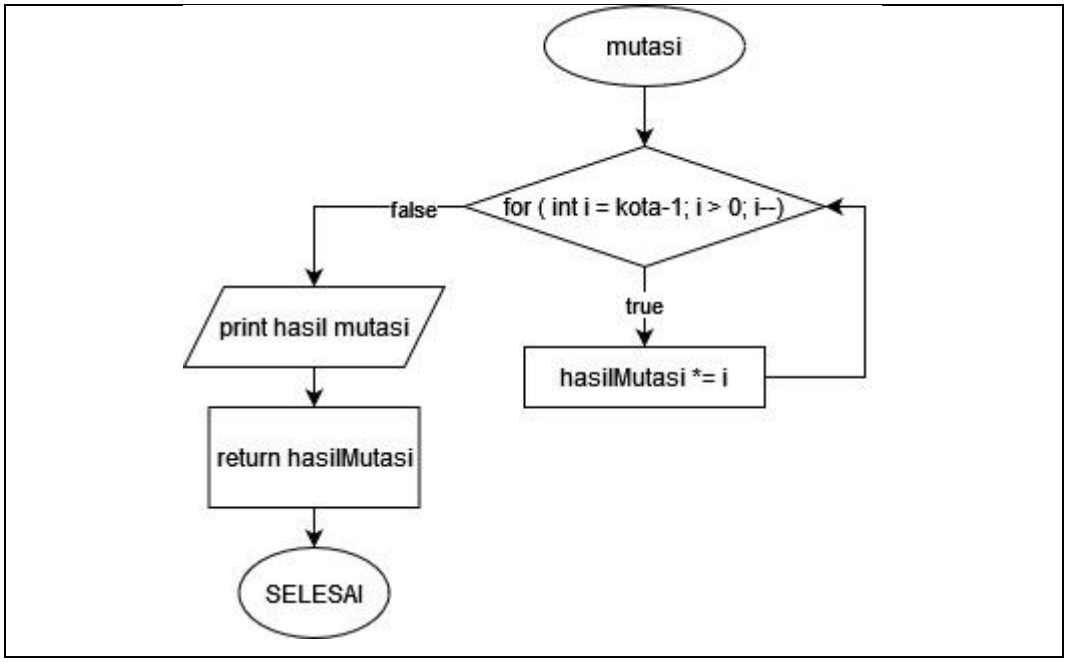

assign
Prnt List kota:
Selesai
false
for(int i = 0; i < kota; i++)
true
str[i] = nama[i];
Print kota

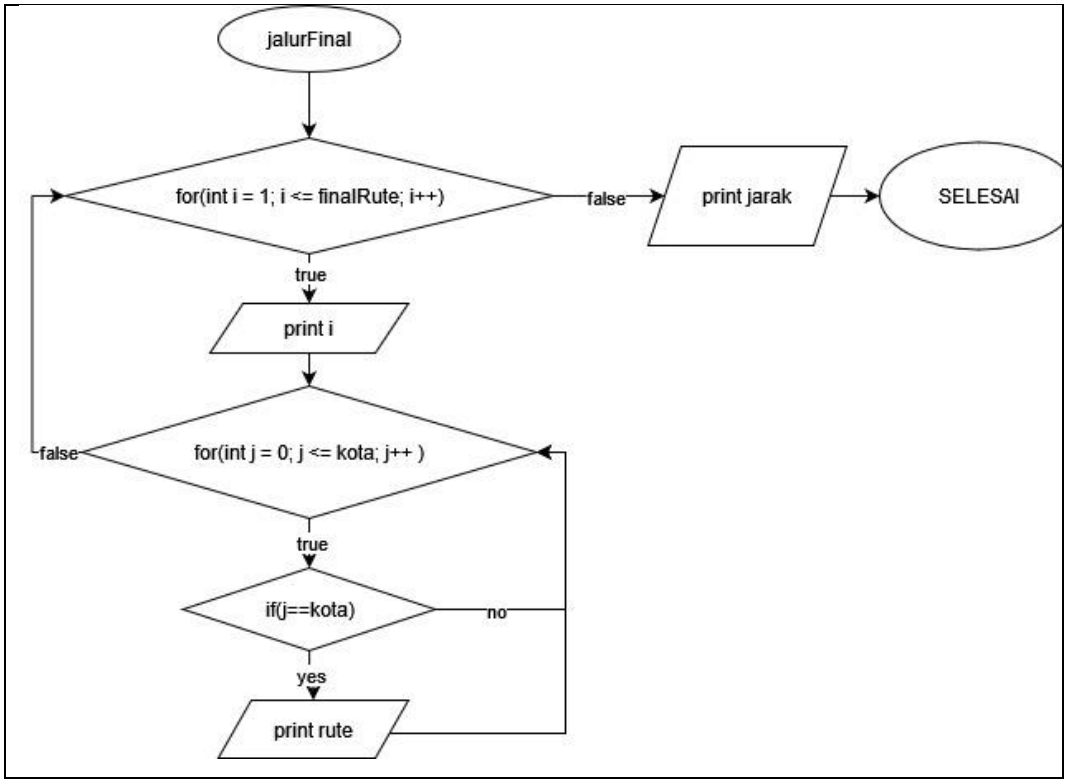

jarak
for (int i = 1; i <= kota; i++)
false
Selesai
true
false
for(int j = 1; j <= kota; j++)
true
if(i == j)
yes
matriks[i][j] = 0
no
if(j < i)
yes
matriks[i][j] = matriks[j][i]
no
Input jarak kota


permutate
if (l == r && a[0] == kotaAwal)
no
yes
a[kota] = a[0]
x++
menghitungJarak(a)
for (int i = l; i <= r; i++)
false
Selesai
true
swap((a + l), (a + i))
permute(a, l + 1, r, kotaAwal)
swap((a + l), (a + i))

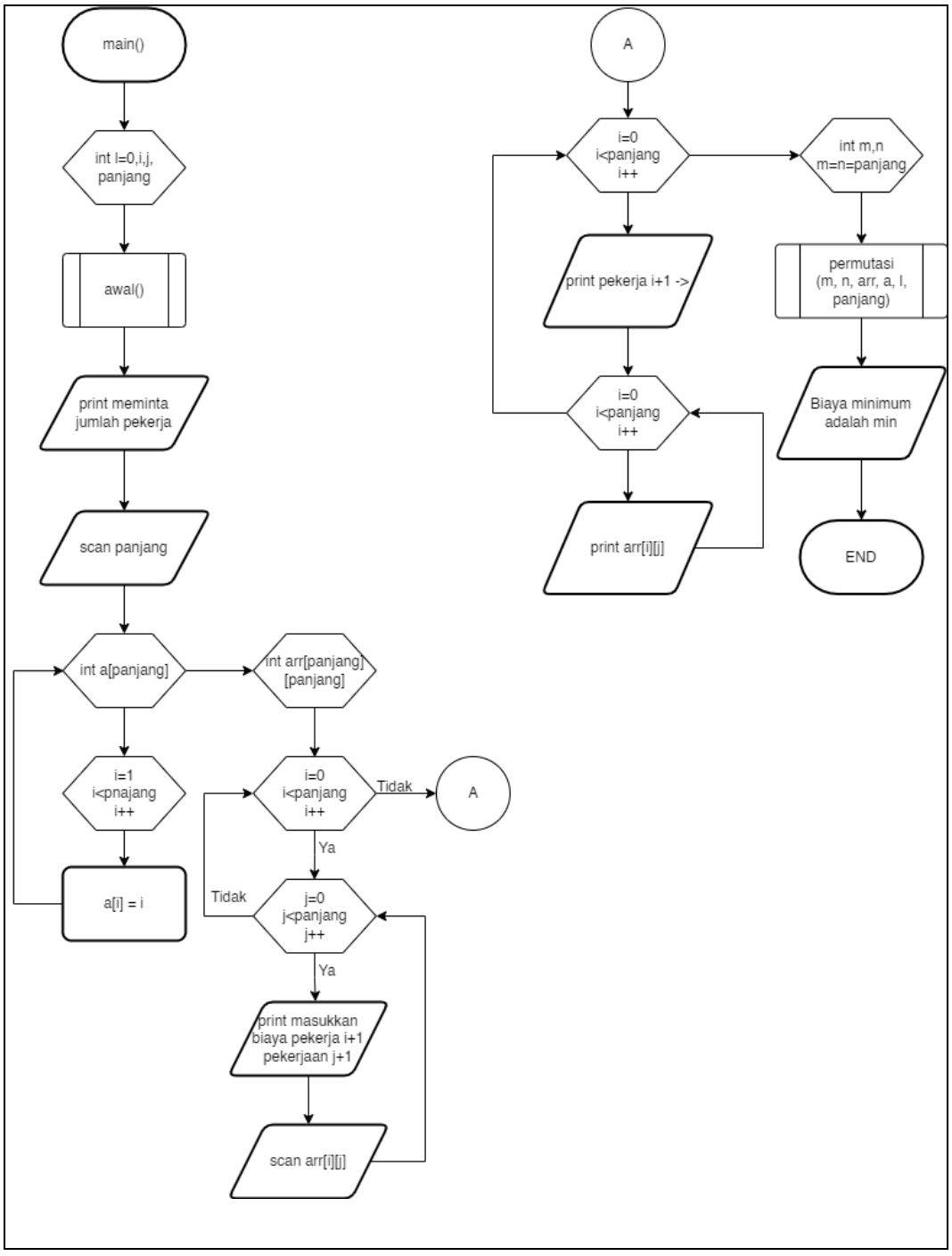

swap
char temp
temp = *x;
*x = *y;
*y = temp;
selesai

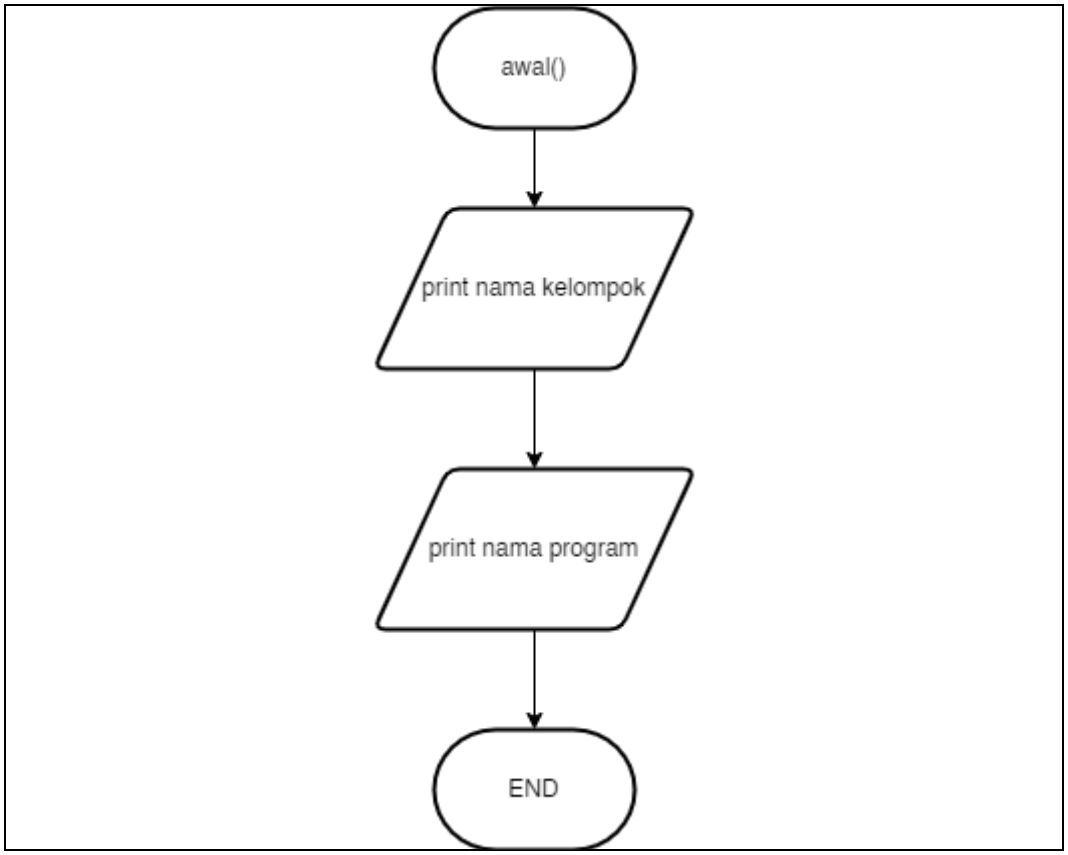

menghitungJarak
int totalSementara
print jalur
for(int i = 0; i <= kota ; i++)
false
true
ascii[i] = (int)a[i] - 96
print kota
hasil[x-1] = totalSementara
false
for(int i = 0; i <= kota ; i++)
true
print totalJarak
print jarak
if (kecil > hasil[x-1]
totalSementara += matriks[ascii[i-1]][ascii[i]]
yes
no
finalRute = 1
kecil = hasil[x-1]
if (kecil == hasil[x-1] || kecil == 1
no
yes
for(int i=0;i <= kota; i++)
false
true
rute[finalRute].jalur[i] = str[i]
for(int i=0;i <= kota; i++)
false
true
kecil = hasil[x-1];
rute[finalRute].jalur[i] = str[i];
SELESAI

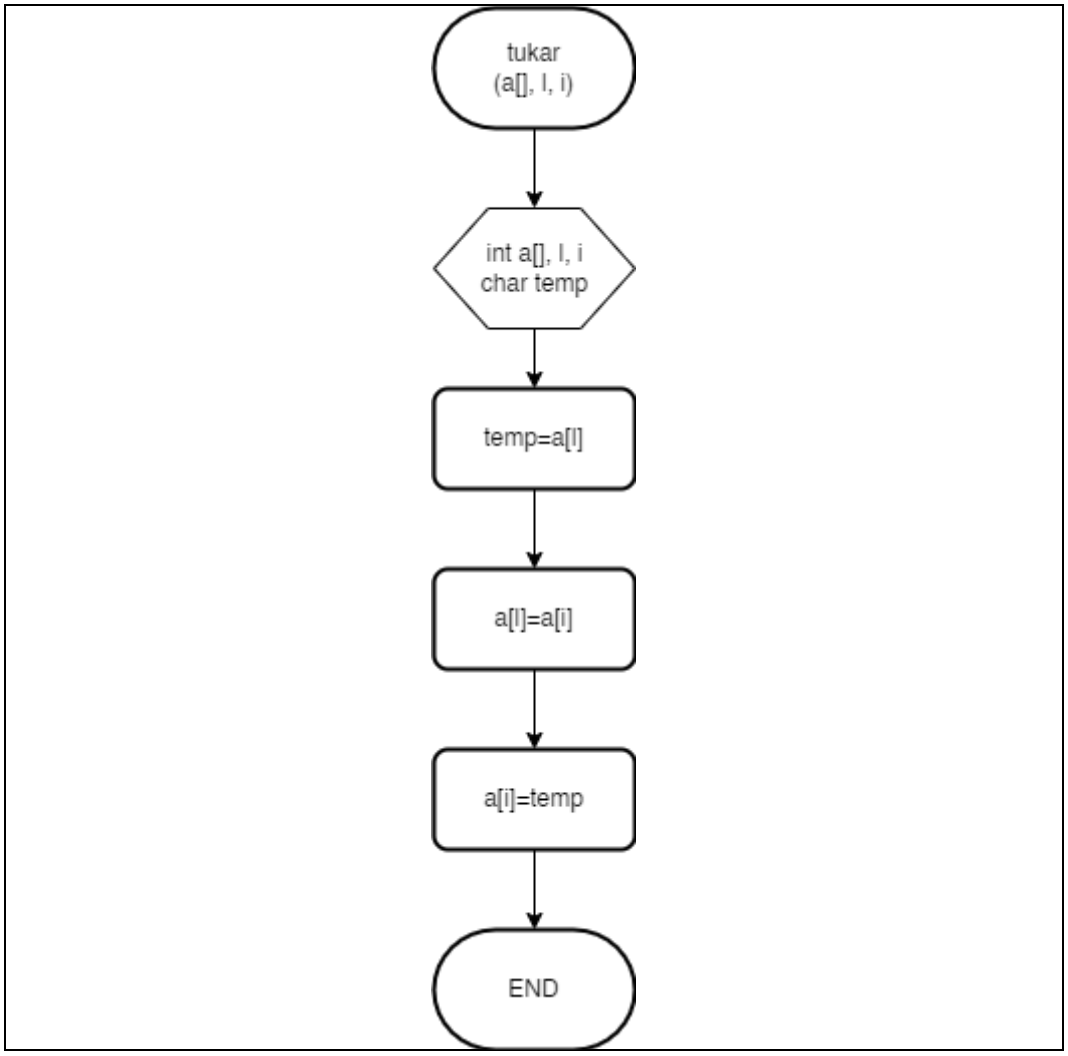

mutasi
for (int i = kota-1; i > 0; i--)
false
true
print hasil mutasi
hasilMutasi *= i
return hasilMutasi
SELESAI

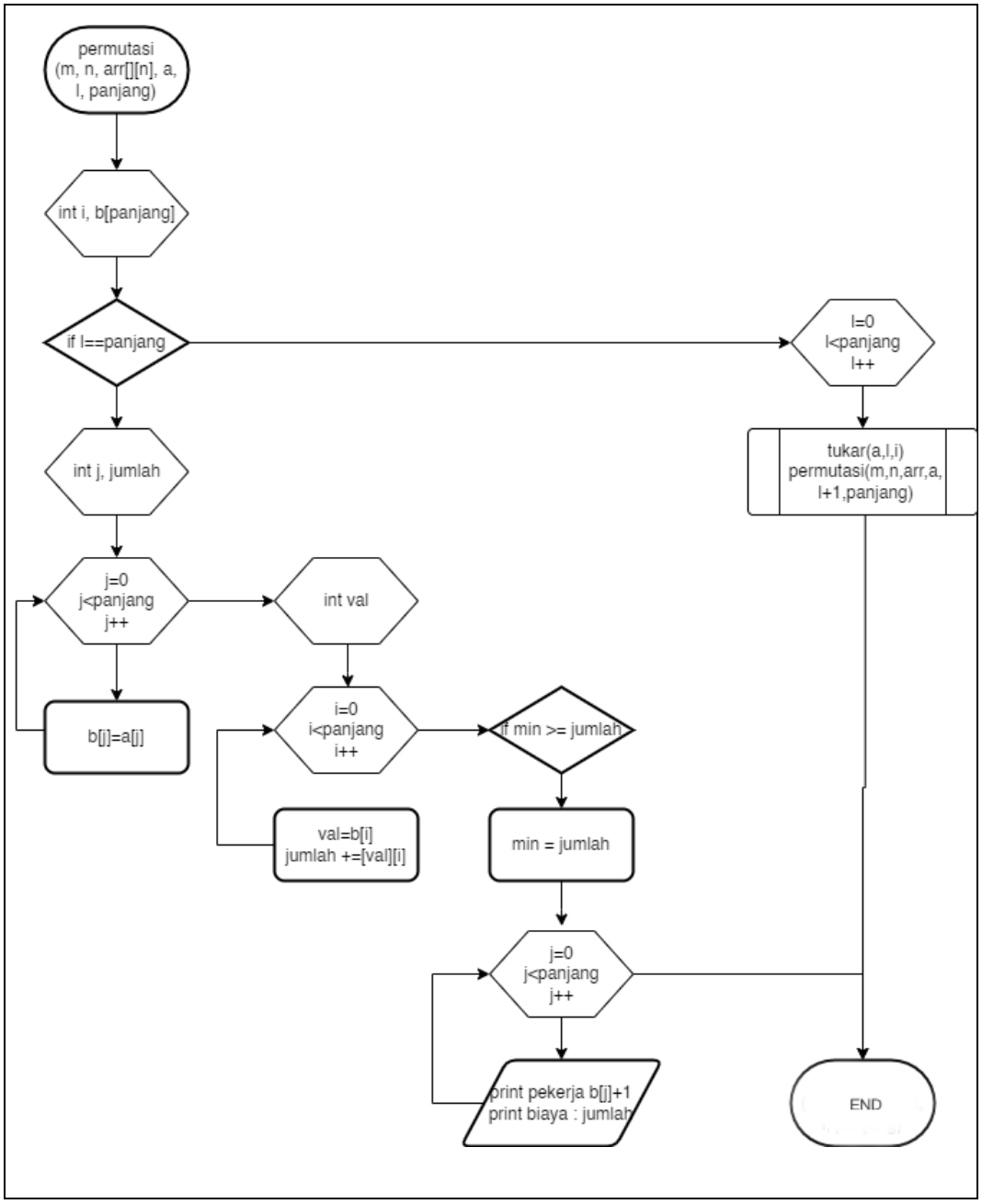

jalurFinal
for(int i = 1; i <= finalRute; i++)
false
print jarak
SELESAI
true
print i
for(int j = 0; j <= kota; j++)
false
true
if(j==kota)
no
yes
print rute

3.3.2 Flowchart Assignment Cost Problem




awal()
print nama kelompok
print nama program
END


tukar
(a[], l, i)
int a[], l, i
char temp
temp=a[l]
a[l]=a[i]
a[i]=temp
END



3.4 Kode Program

3.4.1 Kode Program Traveling Salesaman Problem

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
void swap();
void assign();
void jarak();
void menghitungJarak();
int mutasi();
void jalurFinal();
void namaKelompok();

char matriks[100][100];
int kota;
char nama[26] = "abcdefghijklnopqrstuvwxyz";
char str[] = "";
int hasil[1000000];
int x = 1;
int hasilMutasi = 0;
int ascii[] = {0};
int kecil = 1;
int finalRute = 1;

struct terpendek{
char jalur[1000];
};
struct terpendek rute[1000];

void swap(char *x, char *y){
char temp;
temp = *x;
*x = *y;
*y = temp;
}

char permute(char *a, int l, int r, char kotaAwal){
if (l == r  && a[0] == kotaAwal ) {
a[kota] = a[0];
x++;
menghitungJarak(a);
}
else{
for (int i = l; i <= r; i++){
swap((a + l), (a + i));
permute(a, l + 1, r, kotaAwal);
swap((a + l), (a + i));
}
}
}

int main(){
int hasilMutasi = 1, n;
char kotaAwal;
namaKelompok();
printf("\t\t_____________________________________________________
\n");
printf("\t\tMasukan jumlah kota : ");
scanf("%d",&kota);
assign();
n = strlen(str);
```

```
printf("\t\t_____
\n");
jarak();
hasilMutasi = mutasi(hasilMutasi);
getchar();
printf("\t\tMasukan kota awal : ");
scanf("%c",&kotaAwal);
system("cls");
printf("\t\t_____
\n");
printf("\t\tList jalur :\n");
permute(str,0,n-1,kotaAwal);
// for( int i = 1; i <= hasilMutasi; i++){
//     printf(" hasil %d = %d\n",i, hasil[i]);
// }
// printf("%d rute %c",finalRute, rute[finalRute-1].jalur[1]);
jalurFinal();
return 0;
}

void assign(){
printf("\t\tList kota :\n");
for( int i = 0; i < kota; i++){
str[i] = nama[i];
printf("\t\tKota %c\n",str[i]);
}
}

void jarak(){
for ( int i = 1; i <= kota; i++){
for (int j = 1; j <= kota; j++){
if(i == j) matriks[i][j] = 0;
else if( j < i) matriks[i][j] = matriks [j][i];
else {
printf("\t\tMasukan jarak %c ke %c : ",nama[i-1], nama[j-1]);
scanf("%d", &matriks[i][j]);
}
}
}
}

void menghitungJarak(char *a){
int totalSementara = 0;
printf("\n\t\tJalur ke-%d : \n", x-1);
for(int i = 0; i <= kota ; i++){
ascii[i] = (int)a[i] - 96;
if(i == kota){
printf("%c", a[i]);
}
else if(i == 0){
printf("\t\t%c > ",a[i]);
}
else{
printf("%c > ", a[i]);
}
}
```

```
printf("\n");
printf("\t\tTotal jarak = ");
for (int i = 1; i <= kota ; i++){
if (i == kota){
printf("%d", matriks[ascii[i-1]][ascii[i]]);
}
else{
printf("%d + ", matriks[ascii[i-1]][ascii[i]]);

}
totalSementara += matriks[ascii[i-1]][ascii[i]];
}
hasil[x-1] = totalSementara;
printf(" = %d,\n",hasil[x-1]);
if(kecil > hasil[x-1]){
finalRute = 1;
kecil = hasil[x-1];
for(int i=0;i <= kota; i++){
rute[finalRute].jalur[i] = str[i];
// printf("!%c",rute[finalRute].jalur[i]);
}
// printf("> ni jalur ke-%d terpendek... x = %d\n",finalRute,x);
}
else if(kecil == hasil[x-1] || kecil == 1){
if(kecil == hasil[x-1]) finalRute++;
for(int i=0;i <= kota; i++){
kecil = hasil[x-1];
rute[finalRute].jalur[i] = str[i];
// printf("!%c",rute[finalRute].jalur[i]);
}
// printf("= ni jalur ke-%d terpendek\n",finalRute);
}
}

int mutasi(int hasilMutasi){
for ( int i = kota-1; i > 0; i--){
hasilMutasi *= i;
}
printf("\t\t_____________________________________________________________
\n");
printf("\t\tTotal jalur yang mungkin = %d jalur\n", hasilMutasi);
return hasilMutasi;
}

void jalurFinal(){
printf("\t\t_____________________________________________________________
___ \n");
printf("\t\t_____________________________________________________________
\n\n");
printf("\t\tJalur yang paling efektif ada %d jalur\n\t\tYaitu :
\n",finalRute);
for(int i = 1; i <= finalRute; i++){
printf("\t\t%d. ",i);
for(int j = 0; j <= kota; j++ ){
if(j==kota) printf("%c\n",rute[i].jalur[j]);
else printf("%c > ",rute[i].jalur[j]);
```

```
}
}
printf("\t\tDengan jarak = %d\n\n",kecil);
printf("\t\t_____________________________________________________________
\n");
printf("\t\t_____________________________________________________________
___ \n");
system("pause");
}

void namaKelompok(){
system("cls");
printf("\t\t
_____________________________________________________________________
\n");
printf("\t\t
_____________________________________________________________
\n");
printf("\t\t| |                  TSP PROBLEM                    |
|\n");
printf("\t\t|
|.........................................................|
|\n");
printf("\t\t|
|_____________________________________________________________|
|\n");
printf("\t\t| |                1. Anak Agung Indi Kusuma Putra
| |\n");
printf("\t\t| |             2205551079                        |
|\n");
printf("\t\t| |          1. Muhammad Ibrahim                  |
|\n");
printf("\t\t| |             2205551083                        |
|\n");
printf("\t\t|
|_____________________________________________________________|
|\n");
printf("\t\t_____________________________________________________________
_____________ \n\n");
system("pause");
system("cls");
}
```

3.4.2 Assignment Problem

```
#include <stdio.h>
#include <stdlib.h>

int min=99999; //nilai untuk membandingkan dengan total biaya
void tukar(), permutasi(), awal();

int main(){
    int l = 0;
    int i,j;
    int panjang;
```

```
    awal();

    printf("Masukkan jumlah pekerja dan pekerjaan : ");
    scanf("%d", &panjang);

    int a[panjang];
    for(i = 0; i < panjang; i++){
        a[i] = i;
    }

    printf("\n");

    //memasukkan biaya tiap pekerjaan
    int arr[panjang][panjang];
    for(i = 0; i < panjang; i++){
        for(j = 0; j < panjang; j++){
            printf("Masukkan biaya dari pekerja %d pekerjaan %d:
",i+1, j+1);
            scanf("%d", &arr[i][j]);
        }
        printf("\n");
    }

    system("pause");
    system("cls");

    //print matriks biaya
    for(i=0;i<panjang;i++){
        printf("pekerja %d  ->  ", i+1);
        for(j = 0; j < panjang; j++){
            printf("%d\t", arr[i][j]);
        }
        printf("\n");
    }
    printf("\n\n");

    int m, n;
    m = n = panjang;
    permutasi(m, n, arr, a, l, panjang);

      printf("\n\nBiaya  minimum  untuk  menyelesaikan  seluruh
pekerjaan : %d\n\n",min);

}

void awal(){

    system("cls");
    printf("_____________________________________________________\n")
;
    printf("   _____________________________________________   \n");
    printf("| |                                             | |\n");
    printf("| |        Program Assignment Problem           | |\n");
    printf("| |    dengan Algoritma Exhaustive Search       | |\n");
    printf("| |                                             | |\n");
```

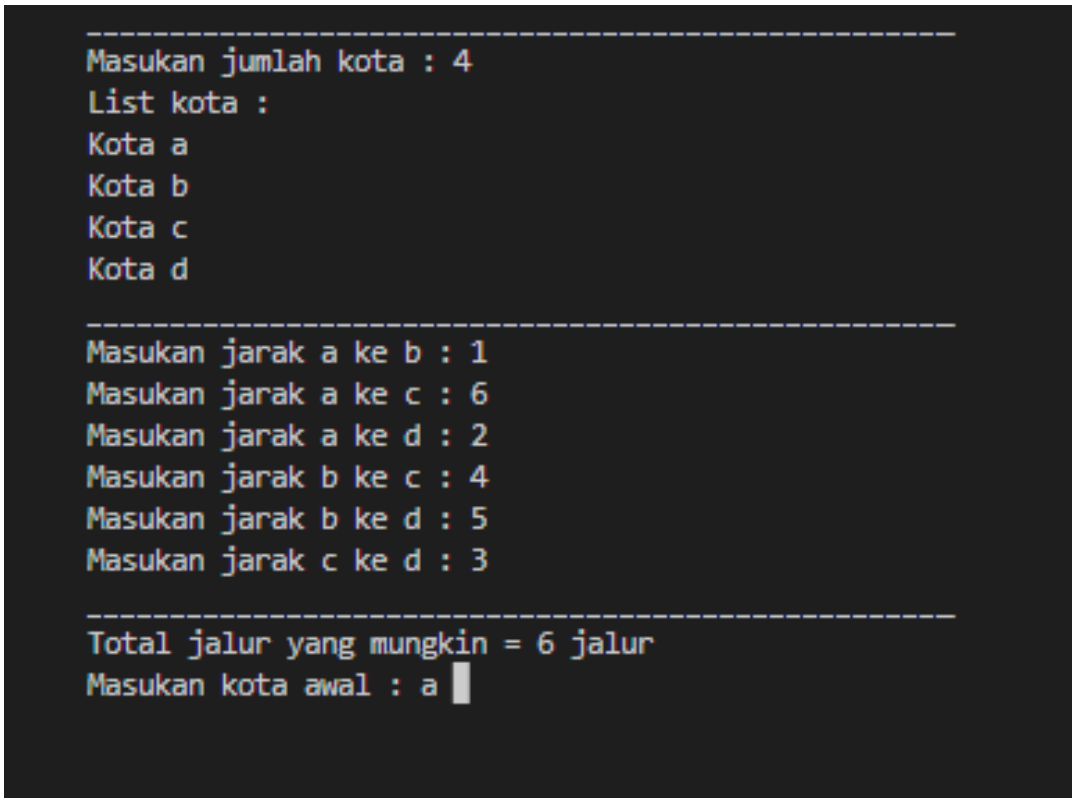
```
    printf("================================================\n")
;
    printf("| |                                            | |\n");
    printf("| |    1. Anak Agung Indi Kusuma Putra         | |\n");
    printf("| |       2205551079                           | |\n");
    printf("| |    1. Muhammad Ibrahim                     | |\n");
    printf("| |       2205551083                           | |\n");
    printf("| |____________________________________________| |\n");
    printf("_____________________________________________________\n\n
");
    system("pause");
    system("cls");


    system("cls");
    printf("_____________________________________________________\n")
;
    printf("   ___________________________________________   \n");
    printf("| |                                            | |\n");
    printf("| |        Program Assignment Problem          | |\n");
    printf("| |    dengan Algoritma Exhaustive Search      | |\n");
    printf("| |____________________________________________| |\n");
    printf("_____________________________________________________\n\n
");
}

//melakukan penukaran posisi setiap setelah
void tukar(int a[],int l,int i){
    char temp;
    temp=a[l];
    a[l]=a[i];
    a[i]=temp;
}

//untuk mencari jumlah biaya terkecil
void  permutasi(int  m,int  n,int  arr[][n],int  a[],int  l,int
panjang){
    int i;
    int b[panjang];
    if(l==panjang){
            int j,jumlah=0;
            for(j=0;j<panjang;j++){
                    b[j]=a[j];
            }
            int val;
            for(i=0;i<panjang;i++){
                    val=b[i];
                    jumlah+=arr[val][i];
            }
            if(min >= jumlah){
                min = jumlah;
                for(j = 0; j<panjang; j++){
                    printf("pekerja %d -> ",b[j]+1);
                }
                printf("\tbiaya : %d\n",jumlah);
            }
```

```
    }
    else{
        for(i=l;i<panjang;i++){
            tukar(a,l,i);
            permutasi(m,n,arr,a,l+1,panjang);
            tukar(a,l,i);
        }
    }
}
```

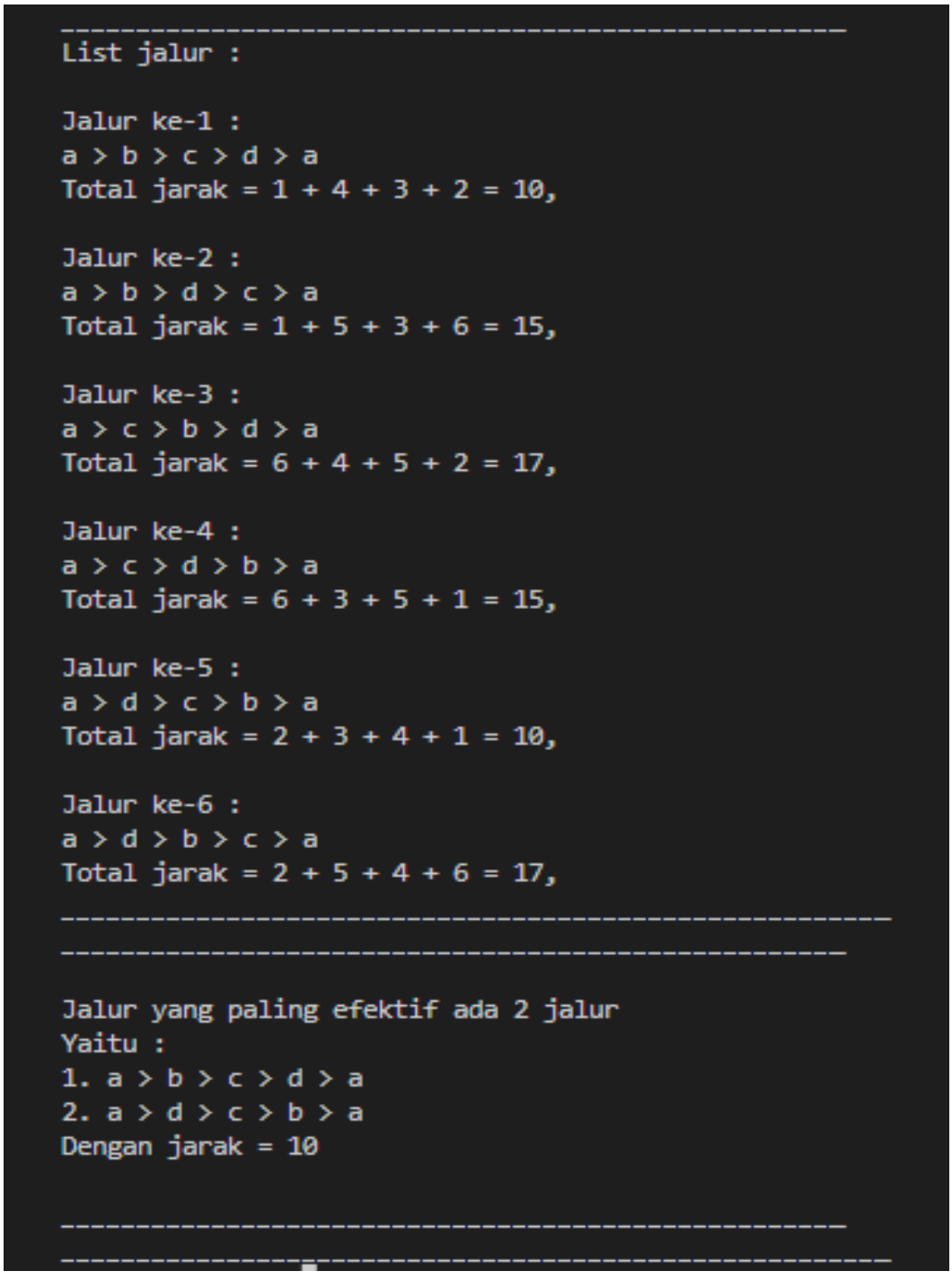
3.5 Hasil Program

3.5.1 TSP Problem

Pertama kita menginputkan jumlah kota dan jarak antar kota, lalu kita pilih kota awal untuk menentukan titik awal dan akhir dari jalur yang akan dibuat



Setelah itu, kita akan mendapatkan semua kemungkinan rute yang ada beserta dengan jarak total dari rute tersebut, setelah semua rute didapat, rute dengan jalur terpendek akan dimunculkan lagi sebagai jalur yang paling efektif



3.5.2 Assigment Problem

```
 ____________________________________________
   ________________________________________
| |                                        | |
| |        Program Assignment Problem      | |
| |    dengan Algoritma Exhaustive Search  | |
| |________________________________________| |
____________________________________________

Masukkan jumlah pekerja dan pekerjaan : 3

Masukkan biaya dari pekerja 1 pekerjaan 1: 6
Masukkan biaya dari pekerja 1 pekerjaan 2: 7
Masukkan biaya dari pekerja 1 pekerjaan 3: 4

Masukkan biaya dari pekerja 2 pekerjaan 1: 8
Masukkan biaya dari pekerja 2 pekerjaan 2: 9
Masukkan biaya dari pekerja 2 pekerjaan 3: 3

Masukkan biaya dari pekerja 3 pekerjaan 1: 4
Masukkan biaya dari pekerja 3 pekerjaan 2: 5
Masukkan biaya dari pekerja 3 pekerjaan 3: 1
```

```
                                  ________
pekerja 1  ->  6        7       4
pekerja 2  ->  8        9       3
pekerja 3  ->  4        5       1


pekerja 1 -> pekerja 2 -> pekerja 3 ->  biaya : 16
pekerja 1 -> pekerja 3 -> pekerja 2 ->  biaya : 14
pekerja 3 -> pekerja 1 -> pekerja 2 ->  biaya : 14
```