

## 1. Cel projektu.

Celem projektu jest rozwiązanie problemu komiwojażera metodą podziału i ograniczeń. Sporządzenie odpowiedniego algorytmu działania i napisanie programu w języku „C”, który będzie realizował powyższy algorytm, rozwiązując dane zagadnienie.

## 2. Metoda podziału i ograniczeń dla problemu komiwojażera - algorytm Little'a (1962r.).

### ZAŁOŻENIE I

Cykl Hamiltona zawiera dokładnie jeden element z każdego wiersza i dokładnie jeden element z każdej kolumny, który zawiera się w macierzy kosztów.

### ZAŁOŻENIE II

Jeżeli od wszystkich elementów jakiegoś wiersza i jakiejś kolumny macierzy kosztów, odejmiemy stałą, to waga każdego cyklu Hamiltona zmniejszy się dokładnie o tę stałą.

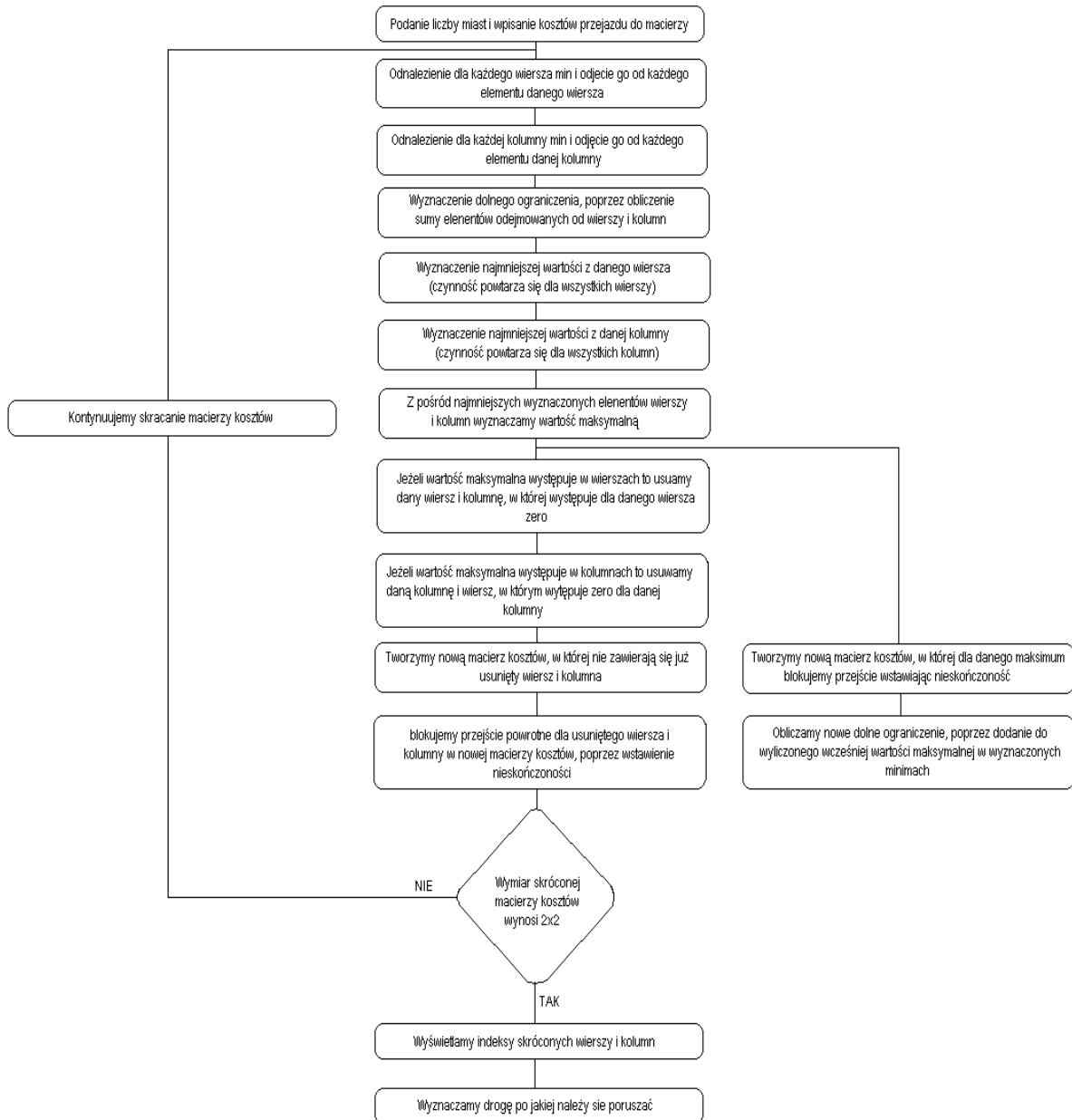
### ZAŁOŻENIE III

Jeżeli wielokrotnie zastosujemy ZAŁOŻENIE II, do momentu aż otrzymamy w każdym wierszu i każdej kolumnie co najmniej jedno zero, a pozostałe elementy macierzy kosztów są nieujemne, to całkowita suma odjętych stałych jest **dolnym ograniczeniem** długości jakiegokolwiek cyklu Hamiltona.

Rozwiązanie można podzielić na dwa podzbiory:

- złożony z rozwiązań zawierających wyróżniony łuk,
- złożony z rozwiązań nie zawierających wyróżnionego łuku.

### 3. Algorytm rozwiązania problemu komiwojażera.



4. Przykład rozwiązania problemu komiwojażera metodą podziału i ograniczeń, wykorzystując algorytm Little'a.

Macierz kosztów:

	1	2	3	4	5
1	$\infty$	12	3	45	6
2	78	$\infty$	90	21	3
3	5	56	$\infty$	23	98
4	12	6	8	$\infty$	34
5	3	98	3	2	$\infty$

- a) Szukamy najmniejszych elementów w wierszach i odejmujemy je od reszty elementów zawartych w nich:

	1	2	3	4	5	min
1	$\infty$	12	3	45	6	3
2	78	$\infty$	90	21	3	3
3	5	56	$\infty$	23	98	5
4	12	6	8	$\infty$	34	6
5	3	98	3	2	$\infty$	2

	1	2	3	4	5
1	$\infty$	9	0	42	3
2	75	$\infty$	87	18	0
3	0	51	$\infty$	18	93
4	6	0	2	$\infty$	28
5	1	96	1	0	$\infty$

- b) Szukamy najmniejszych elementów w kolumnach i odejmujemy je od reszty elementów zawartych w nich:

	1	2	3	4	5
1	$\infty$	9	0	42	3
2	75	$\infty$	87	18	0
3	0	51	$\infty$	18	93
4	6	0	2	$\infty$	28
5	1	96	1	0	$\infty$
min	0	0	0	0	0

	1	2	3	4	5
1	$\infty$	9	0	42	3
2	75	$\infty$	87	18	0
3	0	51	$\infty$	18	93
4	6	0	2	$\infty$	28
5	1	96	1	0	$\infty$

Nowa macierz kosztów:

	1	2	3	4	5
1	$\infty$	9	0	42	3
2	75	$\infty$	87	18	0
3	0	51	$\infty$	18	93
4	6	0	2	$\infty$	28
5	1	96	1	0	$\infty$

- c) Liczymy **dolne ograniczenie**, które jest sumą wszystkich elementów skróconych w wierszach i kolumnach:

$$LB = 3 + 3 + 5 + 6 + 2 + 0 + 0 + 0 + 0 + 0 = 19$$

- d) Przeszukujemy każdy wiersz i kolumnę w poszukiwaniu najmniejszego elementu (Wartość zero jest pomijane w poszukiwaniach, uwzględnia się je dopiero jak wystąpi co najmniej dwa razy):

	1	2	3	4	5	
1	$\infty$	9	0	42	3	3
2	75	$\infty$	87	18	0	18
3	0	51	$\infty$	18	93	18
4	6	0	2	$\infty$	28	2
5	1	96	1	0	$\infty$	1
	1	9	1	18	3	

- e) Z odnalezionych elementów wybieramy wartość posiadającą największy koszt:

	1	2	3	4	5	
1	$\infty$	9	0	42	3	3
2	75	$\infty$	87	18	0	18
3	0	51	$\infty$	18	93	18
4	6	0	2	$\infty$	28	2
5	1	96	1	0	$\infty$	1
	1	9	1	18	3	

- f) Skracamy w zależności wystąpienia największego z minimów kolumnę (wiersz) oraz wiersz (kolumnę), w którym w skracanej kolumnie (wierszu) wystąpiło zero:

	1	2	3	4	5	
1	$\infty$	9	0	42	3	3
2	75	$\infty$	87	18	0	18
3	0	51	$\infty$	18	93	18
4	6	0	2	$\infty$	28	2
5	1	96	1	0	$\infty$	1
	1	9	1	18	3	

Dla danego przykładu usunięta zostanie kolumna 4 i wiersz 5.

- g) Blokujemy przejście powrotne, poprzez wstawienie w kolumnie 5 i wierszu 4 nieskończonego kosztu przebycia drogi:

	1	2	3	4	5
1	$\infty$	9	0	42	3
2	75	$\infty$	87	18	0
3	0	51	$\infty$	18	93
4	6	0	2	$\infty$	$\infty$
5	1	96	1	0	$\infty$

- h) Powstaje nowa skrócona macierz kosztów, oraz macierz, w której zostało zablokowane przejście dla skróconego przejścia. Wartość dolnego ograniczenia powiększona zostaje o koszt usuniętego elementu:

Macierz skrócona LB=19

	1	2	3	5
1	$\infty$	9	0	3
2	75	$\infty$	87	0
3	0	51	$\infty$	93
4	6	0	2	$\infty$

Macierz nie skrócona LB = 19 + 18 = 37

	1	2	3	4	5
1	$\infty$	9	0	42	3
2	75	$\infty$	87	18	0
3	0	51	$\infty$	$\infty$	93
4	6	0	2	$\infty$	$\infty$
5	1	96	1	0	$\infty$

Czynności te są powtarzane do uzyskania macierzy kosztów o wymiarze 2x2.

Ponownie zostają wykonane podpunkty a, b:

	1	2	3	5
1	$\infty$	9	0	3
2	75	$\infty$	87	0
3	0	51	$\infty$	93
4	6	0	2	$\infty$

	1	2	3	5	min
1	$\infty$	9	0	3	0
2	75	$\infty$	87	0	0
3	0	51	$\infty$	93	0
4	6	0	2	$\infty$	0

	1	2	3	5
1	$\infty$	9	0	3
2	75	$\infty$	87	0
3	0	51	$\infty$	93
4	6	0	2	$\infty$
min	0	0	0	0

Wartość dolnego ograniczenia pozostaje bez zmian.

Ponownie zostaje wykonany podpunkt e, f, g:

	1	2	3	5	
1	$\infty$	9	0	3	3
2	75	$\infty$	87	0	75
3	0	51	$\infty$	93	51
4	6	0	2	$\infty$	2
	6	9	2	3	

	1	2	3	5	
1	$\infty$	9	0	3	3
2	75	$\infty$	87	0	75
3	0	51	$\infty$	93	51
4	6	0	2	$\infty$	2
	6	9	2	3	

Skracamy kolumnę 5 i wiersz 2, zaś blokujemy przejście między kolumną 2 i wierszem 5.

Tworzymy nową macierz kosztów, a w drugiej blokujemy kolejne przejście i obliczamy wartość dolnego ograniczenia tak samo jak w podpunkcie h:

Macierz skrócona LB = 19

	1	2	3
1	$\infty$	9	0
3	0	51	$\infty$
4	6	0	2

Macierz nie skrócona LB = 37 + 75 = 112

	1	2	3	4	5
1	$\infty$	9	0	42	3
2	$\infty$	$\infty$	87	18	0
3	0	51	$\infty$	$\infty$	93
4	6	0	2	$\infty$	$\infty$
5	1	96	1	0	$\infty$

Ponownie powtarzamy czynności.

Ponownie zostają wykonane podpunkty a, b:

	1	2	3
1	$\infty$	9	0
3	0	51	$\infty$
4	6	0	2

	1	2	3	min
1	$\infty$	9	0	0
3	0	51	$\infty$	0
4	6	0	2	0

	1	2	3
1	$\infty$	9	0
3	0	51	$\infty$
4	6	0	2
min	0	0	0

Wartość dolnego ograniczenia pozostaje bez zmian.

Ponownie zostaje wykonany podpunkt e, f, g:

	1	2	3	
1	$\infty$	9	0	9
3	0	51	$\infty$	51
4	6	0	2	2
	6	9	2	

	1	2	3	
1	$\infty$	9	$\infty$	9
3	0	51	$\infty$	51
4	6	0	2	2
	6	9	2	

Skracamy kolumnę 1 i wiersz 3, zaś blokujemy przejście między kolumną 3 i wierszem 1.

Tworzymy nową macierz kosztów, a w drugiej blokujemy kolejne przejście i obliczamy wartość dolnego ograniczenia tak samo, jak w podpunkcie h:

Macierz skrócona LB = 19

	2	3
1	9	$\infty$
4	0	2

Macierz nie skrócona LB = 112 + 51 = 163

	1	2	3	4	5
1	$\infty$	9	0	42	3
2	$\infty$	$\infty$	87	18	0
3	0	$\infty$	$\infty$	$\infty$	93
4	6	0	2	$\infty$	$\infty$
5	1	96	1	0	$\infty$

Uzyskaną macierz kosztów o wymiarze 2x2 przeszukujemy w poszukiwaniu najmniejszych wartości w kolumnach i wierszach, następnie redukujemy macierz tak, aby w każdym wierszu i kolumnie występowało zero, a w kolejnym kroku dodajemy odjęte wartości do dolnego ograniczenia.

	2	3	
1	9	$\infty$	9
4	0	2	0

	2	3	
1	0	$\infty$	
4	0	2	
	0	2	

	2	3	
1	0	$\infty$	
4	0	0	

$$LB = 19 + 9 + 2 = 30$$

Ustalamy drogę po jakiej powinniśmy się poruszać na podstawie kolejno skracanych wierszy i kolumn:

$$(5\ 4)\ (2\ 5)\ (3\ 1)$$

oraz wykorzystując zawartość macierzy końcowej:

$$(1\ 2)\ (4\ 3)$$

Daje to nam następującą drogę:

$$(3\ 1)\ (1\ 2)\ (2\ 5)\ (5\ 4)\ (4\ 3)\ (3\ 1)$$

## 5. Listing programu.

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

main()
{

//Deklaracje zmiennych

int miasta[10][10], i, j, n, m, min, pomoc[10], pomoc2[10], k=0, suma=0;
int zera=0, h=0, pp;
int max, szuki, szukj, ciag[200], szukane[20], dobry[10], dobry2[10];
int indi, indj, ii, jj, x, y, indii, indjj;
int maxmin1, maxmin2, pomocnik, petla, f, b=0, kolejny=0;

//Wprowadzenie danych wejsciowych

clrscr();
puts("\t\t*** Problem komiwojazera dla n miast ***\n");
puts("\t\t***metoda podzialu i ograniczen***\n");
puts("\t\t***wg algorytmu Little'a (1962r.)***");
printf("\nPodaj ilosc miast: ");
scanf("%d", &m);
puts("\n");
puts("\t\t*** Przy braku drogi z miasta do miasta, nalezy wpisac: 999 ***\n");
n=m+1;
petla=m-1;

//Wprowadzenie indeksowania macierzy i wprowadzenie do niej danych

j=0;
for(i=0; i<n; i++)
{
miasta[i][0]=h;
h++;
}
i=0;
h=0;
for(j=0; j<n; j++)
{
miasta[0][j]=h;
h++;
}

/* for(i=1; i<=n; i++)
for(j=1; j<=n; j++)
if(i==j) miasta[i][j]=999;
pp=1; */
for(i=1; i<=m; i++)
for(j=1; j<=m; j++)
{
if(i==j)miasta[i][j]=999;
else
{
printf("Podaj odleglosc miedzy miastem %d a %d: ", i, j);
scanf("%d", &miasta[i][j]);
puts("\n");
//pomocnik=miasta[i][j];
//miasta[j][i]=pomocnik;
}
}

//Wydruk macierzy wejsciowej

clrscr();
printf("\n\nMacierz miast:");
for(i=0; i<n; i++)
{
puts("\n");
for(j=0; j<n; j++)
```

```

    printf(" %d", miasta[i][j]);
}
getch();

//Szukanie maksimum z minimum, jakie mozna odjac od wierszy i kolumn

for(petla; petla>=2; petla--)
{
    for(k=0; k<10; k++)
    {
        pomoc[k]=0;
        pomoc2[k]=0;
    }
    k=1;
    for(i=1; i<=n; i++)
    {
        min=miasta[i][1];
        for(j=1; j<n; j++)
        {
            if(miasta[i][j]<=min && miasta[i][j]>=0 && miasta[i][j]<900)
                min=miasta[i][j];
        }
        pomoc[k]=min;
        k++;
    }
    k=1;
    for(i=1; i<n; i++)
    {
        for(j=1; j<n; j++)
        {
            if(miasta[i][j]<900) miasta[j][i]=miasta[i][j]-pomoc[k];
        }
        k++;
    }
    h=1;
    for(j=1; j<n; j++)
    {
        min=miasta[1][j];
        for(i=1; i<n; i++)
        {
            if(miasta[i][j]<min && miasta[i][j]>=0 && miasta[i][j]<900)
                min=miasta[i][j];
        }
        pomoc2[h]=min;
        h++;
    }
    h=1;
    for(j=1; j<n; j++)
    {
        for(i=1; i<=n; i++)
        {
            if(miasta[i][j]<900) miasta[i][j]=miasta[i][j]-pomoc2[h];
        }
        h++;
    }
}

printf("\n\n");
printf("\n najmniejsze elementy z wierszy:");
for(h=1; h<n; h++)
    printf(" %d",pomoc[h]);
printf("\n\n");
printf("\n najmniejsze elementy z kolumn:");
for(h=1; h<n; h++)
    printf(" %d",pomoc2[h]);

//Wydruk macierzy po odjeciu

printf("\n\nMacierz miast:");
for(i=0; i<n; i++)
{
    puts("\n");
    for(j=0; j<n; j++)
        printf(" %f", miasta[i][j]);
}

//obliczenie sumy Kosztow

```

```

for(h=0; h<n; h++)
{
if(pomoc[h]==-1)pomoc[h]=0;
if(pomoc2[h]==-1)pomoc2[h]=0;

suma=suma+pomoc[h]+pomoc2[h];
}
printf("\n\nLow Bound= %d", suma);
getch();

//Szukanie maksimum w minimach do skrocenia kolumny i wiersza

if(n>3)
{
//Przeszukanie wierszy w poszukiwaniu minimow
k=1;
zera=0;
for(i=1; i<n; i++)
{
min=999;
for(j=1; j<n; j++)
{

if(miasta[i][j]==0) zera++;
if(zera>=2) min=0;
else
{
if(miasta[i][j]<900 && miasta[i][j]>0 && miasta[i][j]<=min)
min=miasta[i][j];
}
pomoc[k]=min;
}
k++;
zera=0;
}
//przeszukanie kolumn w poszukiwaniu minimow
k=1;
for(j=1; j<n; j++)
{
min=999;
for(i=1; i<n; i++)
{

if(miasta[i][j]==0)zera++;
if(zera>=2)min=0;
else
{
if(miasta[i][j]<900 && miasta[i][j]>0 && miasta[i][j]<=min)
min=miasta[i][j];
}
pomoc2[k]=min;
}
k++;
zera=0;
}

//wyswietlenie minimow z wierszy i kolumn

printf("\n\n\n");
for(k=1; k<n;k++)
printf(" %d,",pomoc[k]);
printf("\n\n");
for(k=1; k<n;k++)
printf(" %d,",pomoc2[k]);

getch();
// znalezienie najwiekszego z minimow i wyswietlenie go

maxmin1=pomoc[1];
for(k=1; k<n; k++)
{
if(pomoc[k]>maxmin1)
{
maxmin1=pomoc[k];
szuki=k;
}
}

```

```

    }

}
maxmin2=pomoc2[1];
for(k=1; k<n; k++)
{
    if(pomoc2[k]>=maxmin2)
    {
        maxmin2=pomoc2[k];
        szukj=k;
    }
}
printf("\n\n najwieksze min z wierszy to %d",maxmin1);
printf("\n\n najwieksze minimum z kolumn to %d",maxmin2);
max=0;
if(maxmin1<maxmin1)
{
    max=maxmin1;
}
else
{
    max=maxmin1;
}
if(max==maxmin1)
{
    for(j=1;j<n;j++)
    if(miasta[szukj][j]==max)
    {
        indi=szukj;
        indj=j;
    }
}
else
if(max==maxmin2)
{
    for(i=1;i<n;i++)
    if(miasta[i][szukj]==max)
    {
        indi=i;
        indj=szukj;
    }
}
printf("\n\n maksymalna wartosc to %d",max);
printf("\n\n dla indeksow: i=%d, j=%d",indi,indj);
getch();

```

//poszukiwanie kolumny i wiersza, ktore zostana skrocone

```

if(max==maxmin1)
{
    for(j=1;j<n;j++)
    if(miasta[indi][j]==0)
    {
        jj=j;
        ii=indi;
    }
}
if(max==maxmin2)
{
    for(i=1;i<n;i++)
    if(miasta[i][indj]==0)
    {
        ii=i;
        jj=indj;
    }
}
szukane[b]=miasta[ii][0];
indii=szukane[b];
b++;
szukane[b]=miasta[0][jj];
indjj=szukane[b];
b++;
printf("\n\n skracamy wiersz: %d i kolumne %d",indii,indjj);

```

//Blokowanie przejścia z powrotem do miasta x

```
miasta[indj][indii]=999;
```

```
//Drukowanie macierzy po przekształceniach
```

```
printf("\n\nMacierz miast:");
for(i=0; i<n; i++)
{
    puts("\n");
    for(j=0; j<n; j++)
        printf(" %d", miasta[i][j]);
}
```

```
//Zmniejszanie macierzy i wpisywanie do niej danych
```

```
for(i=0; i<n; i++)
{
    miasta[i][j]=-1;
}
for(j=0; j<n; j++)
{
    miasta[ii][j]=-1;
}
getch();
printf("\n\nMacierz miast:");
for(i=0; i<n; i++)
{
    puts("\n");
    for(j=0; j<n; j++)
        printf(" %d", miasta[i][j]);
}
```

```
getch();
```

```
h=0;
for(i=0; i<n; i++)
{
    for(j=0; j<n; j++)
    {
        if(miasta[i][j]>0)
        {
            ciag[h]=miasta[i][j];
            h++;
        }
    }
}
```

```
//puts("\n");
for(h=0; h<n*n; h++)
{
    //printf(" %d ", ciag[h]);
}
```

```
n--;
h=0;
for(i=0; i<n; i++)
{
    for(j=0; j<n; j++)
    {
        miasta[i][j]=ciag[h];
        h++;
    }
}
```

```
//Drukowanie macierzy po przekształceniach
```

```
printf("\n\nMacierz miast:");
for(i=0; i<n; i++)
{
    puts("\n");
    for(j=0; j<n; j++)
        printf(" %d", miasta[i][j]);
}
```

```
//Dopisywanie do kosztu i szukanej drogi, niezbędnych danych
```

```
for(i=1; i<n; i++)
```

```

{
for(j=1; j<n; j++)
{
if(miasta[i][j]>0 && miasta[i][j]<900)
{
suma=suma+miasta[i][j];
}
}
}
if(m>3)
{
if((miasta[1][1]==999 && miasta[1][2]==999) || (miasta[1][2]==999 && miasta[2][2]==999) || (miasta[2][2]==999 && miasta[2][1]==999) ||
(miasta[2][1]==999 && miasta[1][1]==999))
puts("\nOdnies sie do macierzy poprzedniej, by odnalezc ostatnie elementy szukanej drogi\n");

else

if(miasta[1][1]==999 || miasta[2][2]==999)
{
szukane[b]=miasta[2][0];
b++;
szukane[b]=miasta[0][1];
b++;
szukane[b]=miasta[1][0];
b++;
szukane[b]=miasta[0][2];
}
else
{
szukane[b]=miasta[1][0];
b++;
szukane[b]=miasta[0][1];
b++;
szukane[b]=miasta[2][0];
b++;
szukane[b]=miasta[0][2];
}
}
}
//Wyswietlanie koncowej drogi i kosztu

printf("\n\nKoncowy koszt podrozy wynosi: %d\n", suma);
puts("\n\nSzukane: ");
for(f=0; f<2*m; f++)
printf("%d ", szukane[f]);
kolejny=szukane[0];
puts("\nSzukana droga:\n");
for(i=0; i<m; i++)
{
for(f=0; f<2*m; f=f+2)
{
if(szukane[f]==kolejny)
{
dobry[b]=szukane[f];
f++;
dobry2[b]=szukane[f];
kolejny=szukane[f];
f=0;
b++;
}
}
}

for(b=0; b<=m-1; b++)
{
printf("(%d %d)",dobry[b], dobry2[b]);
}
}
getch();

//Koniec programu
clrscr();
puts("\t\t\t***Program napisany na zajecia***\n");
puts("\t\t\t***ze Sterowania Procesami Dyskretnymi***\n");
puts("\t\t\t***prowadzacy: Dr inz. Janusz Paplinski***\n");
getch();
return 0;
}

```