-------+الگوریتم فروشنده ی دوره گرد ----------+----------
نوشته شده توسط :|
|حسام حاتمی|

| |
|اوپتیمالی:|
| O(n^4) |
+---------- ----------+
Http://Hesam-h.ir

توضیحات الگوریتم

کدی که در ذیل مشاهده میکنید راه حلی برای حل مسئله دوره گرد است که از دور همیلتونی برای حل این مسئله استفاده مینماید .میزان کارایی آن O (n^4) میباشد . در میان کدها کامنت به اندازی کافی برای فهم کد آورده شده است .

مشکلات الگوریتم :

این الگوریتم برای پیدا کردن کوتاه ترین مسیری است که یک فروشنده میتواند به نمامی شهرها سر بزند و این کار را تنها یکبار انجام دهد و بین شهرهایی سفر نماید که مسافتشان در حد نهایی مقدار کمینه باشد.

یا به عنوان دیگر : یافتن یک دور همیلتونی با کمترین وزن گرافی داده شده .

- تعداد شهرها = No. of cities
- تعداد مسیرهای ارتباطی= Number of paths
- وزن مسیرهای ارتباطی= .Distance of the path

الگوریتم :

```
MIN_CKT_LENGTH=INFINITY
                                                                    برای هر راس (v) در گراف g
                                                  برای هر راس (V-N) در گراف g به همراه خود V و V-N متفاوت هستند
                                                              به جز راس بالا بقیه شهرها را علامتگذاری نشده قرار بده
                                                                        راس V را به عنوان بازدید شده بگذار
برای راس شروع شده مثلا (V) و رئوسی که با موفقیت از طریق این راس پیمایش میشوند مانندV-n ،دورها را پیدا کن
                                                                  این مسیری که الان بازدید کردی از V-N مینیمم قرار بده


for each vertex V in the graph G
  for each vertex V_N in the graph G such that V and V_N are different
   mark all vertices unvisited
   mark V as visited
     for staring vertex as V and succeeding vertex as V_N, find circuit
          such that path staring from V_N in that circut yeilds minimum
          pathlength from V_N for all unvisited vertices by
          visiting each vertex.( this path is obtained by greedy method).
       if currently obtained circuit length <= MIN_CKT_LENGTH then
         set MIN_CKT_LENGTH=newly obtained value
         copy the new circuit as hamiltonion circuit
       end if
     end for V_N
   end for V
```

کد سگمنت :

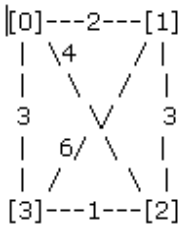
```
//for each vertex, S_V as a staring node

for(int S_V_id=0;S_V_id<n;S_V_id++)

{
     //for each and non start vertex as I
     for(i=0;i<n;i++)
     {
        //set all to unvisited
        set_visited(ALL,FALSE);
        // set staring vertex as visited
        set_visited(S_V_id,TRUE);
        //reset/init minimum circuit
        reset_min_circuit(S_V_id);
        // obtain circuit for combination of S_V and i
        new_circuit_length=get_valid_circuit(S_V_id,i);
        // if newer length is less than the previously
        //calculated min then set it as min and set the
        //current circuit in hamiltonion circuit
        if(new_circuit_length<=min_circuit_length)
             SET_HAM_CKT(min_circuit_length=new_circuit_length);
      }

}
```

مثال :



ورودی ها:

۱) بینهایت 999: (infinity) در صورتی بینهایت خواهد شد که بیش از ۹۹۹ بار به جواب نرسد، میتوان این مقدار را را نیز تغغیر داد.
۲) تعداد شهرها : (no. of cities: 4) در اینجا به صورت پیشفرض ۴ در نظر گرفته میشود.
۳) تعداد مسیرها : (no. of paths:6) در اینجا مسیرهای بین شهرها به صورت پیشفرض ۶ در نظر گرفته میشود.
۴) ورودی ها را در بدو ورود به این شکل وارد میگنیم:

نکته :
گفته بودیم که گراف ۴ راس (همانند شکل بالا) و بین رئوس آن ۶ مسیر وجود دارد .

S D Dist
path0:0 1 2 از راس ۰ به ۱ با وزن ۲
path0:0 2 4 از راس ۰ به ۲ با وزن ۴
path0:0 3 3 از راس ۰ به ۳ با وزن ۳
path0:1 2 3 از راس ۱ به ۲ با وزن ۳
path0:1 3 6 از راس ۱ به ۳ با وزن ۶
path0:2 3 1 از راس ۲ به ۳ با وزن ۱
*/

الگوریتم به شکل زیر پردازش میشود :

V(راس)	V_N	V_N-path	ckt_length, ckt (started with INFI,-)	MIN_CKT_LENGTH, HAM_CKT (started with INFI,-)
0	1	1-2-3	9,0-1-2-3-0*	9,0-1-2-3-0
	2	2-3-1	13,0-2-3-1-0	9,0-1-2-3-0
	3	3-2-1	9,0-3-2-1-0*	9,0-3-2-1-0
1	0	0-3-2	9,1-0-3-2-1*	9,1-0-3-2-1
	2	2-3-0	9,1-2-3-0-1*	9,1-2-3-0-1
	3	3-2-0	13,1-3-2-0-1	9,1-2-3-0-1
2	0	0-1-3	13,2-0-1-3-2	9,1-2-3-0-1
	1	1-0-3	9,2-1-0-3-2*	9,2-1-0-3-2
	3	3-0-1	9,2-3-0-1-2*	9,2-3-0-1-2

3	0	0-1-2	9,3-0-1-2-3*	9,3-0-1-2-3
	1	1-0-2	13,3-1-0-2-3	9,3-0-1-2-3
	2	2-1-0	9,3-2-1-0-3*	9,3-2-1-0-3

کد الگوریتم دوره گرد :

```
#include<stdio.h>
#include<conio.h>
#define ALL -1
#define MAXCITIES 10

enum BOOL{FALSE,TRUE};
long*visited;//نودهای  پیمایش شده  اینجا علامتگذاری میشوند
long*min_circuit;//min inner circuit for given node as start node at position indexed 0
long*ham_circuit;//optimal circuit with length stored at position indexed 0
long min_circuit_length;//min circuit lenth for given start node

int n;//city count
long matrix[MAXCITIES][MAXCITIES];//nondirectional nXn symmetric matrix
//to store path distances as sourceXdestination
long INFI;// INFINITY value to be defined by user

// function resets minimum circuit for a given start node
//with setting its id at index 0 and setting furthr node ids to -1
void reset_min_circuit(int s_v_id)
{
        min_circuit[0]=s_v_id;
        for(int i=1;i<n;i++) min_circuit[i]=-1;
}

// marks given node id with given flag
// if id==ALL it marks all nodes with given flag
void set_visited(int v_id,BOOL flag)
{
        if(v_id==ALL)       for(int i=0;i<n;i++) visited[i]=flag;
        else                visited[v_id]=flag;
}

// function sets hamiltonion circuit for a given path length
//with setting it at index 0 and setting furthr nodes from current min_circuit
void SET_HAM_CKT(long pl)
{
        ham_circuit[0]=pl;
        for(int i=0;i<n;i++)      ham_circuit[i+1]=min_circuit[i];
        ham_circuit[n+1]=min_circuit[0];
}

//function sets a valid circuit by finiding min inner path for a given
//combination start vertex and next vertex to start vertex such that
// the 2nd vertex of circuits is always s_n_v and start and dest node is
//always s_v for all possible values of s_n_v, and then returns the
// valid circuit length for this combination

long get_valid_circuit(int s_v,int s_n_v)
{
        int next_v,min,v_count=1;
        long path_length=0;
```

```
	min_circuit[0]=s_v;
	min_circuit[1]=s_n_v;
	set_visited(s_n_v,TRUE);
	path_length+=matrix[s_v][s_n_v];
	for(int V=s_n_v;v_count<n-1;v_count++)
	{	min=INFI;
			for(int i=0;i<n;i++)
				if(	matrix[V][i]<INFI && !visited[i]
					&& matrix[V][i]<=min
				  )
					min=matrix[V][next_v=i];
		set_visited(next_v,TRUE);
		V=min_circuit[v_count+1]=next_v;
		path_length+=min;
	}
	path_length+=matrix[min_circuit[n-1]][s_v];
	return(path_length);
}

void main()
{
	clrscr();
	printf("Make sure that infinity value < sum of all path distances\nSet Infinity at (signed long):");
	scanf("%ld",&INFI);
	int pathcount,i,j,source,dest;
	long dist=0;
	long new_circuit_length=INFI;
	printf("Enter no. of cities(MAX:%d):",MAXCITIES);
	scanf("%d",&n);
	printf("Enter path count:");
	scanf("%d",&pathcount);

	printf("Enter paths:< source_id destination_id distance >\n ids varying from 0 to %d\n",n-1);
	//init all matrix distances to infinity
	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			matrix[i][j]=INFI;

	//populate the matrix
	for(i=0;i<pathcount;i++)
	{
		printf("[path %d]:",i);
		scanf("%d %d %ld",&source,&dest,&dist);
		if(source!=dest)
		    matrix[source][dest]=matrix[dest][source]=dist;
	}

	visited=new long[n];
	min_circuit=new long[n];
	ham_circuit=new long[n+2];
	min_circuit_length=INFI;
	// algorithm
	//for each vertex, S_V as a staring node
	for(int S_V_id=0;S_V_id<n;S_V_id++)
	{	//for each and non start vertex as i
		for(i=0;i<n;i++)
		{	//set all to unvisited
			set_visited(ALL,FALSE);
			// set staring vertex as visited
			set_visited(S_V_id,TRUE);
			//reset/init minimum circuit
			reset_min_circuit(S_V_id);
			// obtain circuit for combination of S_V and i
			new_circuit_length=get_valid_circuit(S_V_id,i);
			// if newer length is less than the previously
			//calculated min then set it as min and set the
			//current circuit in hamiltonion circuit
```

```
                                        if(new_circuit_length<=min_circuit_length)
                                                SET_HAM_CKT(min_circuit_length=new_circuit_length);
                                }
                }
// if any circuit found
if(min_circuit_length<INFI)
{
                printf("\n\nMinimum circuit length is: %ld\nCircuit is:\n",min_circuit_length);
                for(i=1;i<n+2;i++) printf("<%ld> ",ham_circuit[i]);
}
else            printf("\n\nNo hamiltonian circuit !");
getch();
delete []visited;
delete []min_circuit;
delete []ham_circuit;
}
```