Bài toán người du lịch theo phương pháp nhánh cận

1. **Bài toán**

Một người du lịch muốn tham quan n thành phố T1,…,Tn. Xuất phát từ một thành phố nào đó, người du lịch muốn đi qua tất cả các thành phố còn lại, mỗi thành phố đi qua đúng 1 lần rồi quay trở lại thành phố xuất phát.

Gọi Cij là chi phí đi từ thành phố T1 đến Tj. Hãy tìm một hành trình thỏa yêu cầu bài toán sao cho chi phí là nhỏ nhất

1. **Mô tả chi tiết bài toán**

1.Ý tưởng

Gọi là một hoán vị của {1,…,n} thì một hành trình thỏa yêu cầu bài toán có dạng: .

Nên có tất cả n! hành trình như thế.

Nếu ta cố định một thành phố xuất phát, chẳng hạn , thì có (n-1)! hành trình.

Bài toán chuyển về dạng:

Tìm Min{f(a2,…,an) : (a2,..,an) là hoán vị của {2,..,n}}

Với f(a1,..,an) =

Cách giải bài toán sẽ kết hợp đánh giá nhánh cận trong quá trình liệt kê phương án của thuật toán quay lui.

2.Thiết kế bài toán

Try(i)

For(j=1n)

If(chấp nhận được)

{

Xác định  theo j;

Ghi nhận trạng thái mới;

If(i==n)

Cập nhật lời giải tối ưu;

Else

{

Xác định cận g();

If(g() )

Try(i+1);

}

Trả lại trạng thái cũ cho bài toán;

}

* Nếu ta cố định xuất phát từ , ta duyệt vòng lặp từ j=2
* Đánh giá nhánh cận:

Đặt : Cmin = Min{:i,j {1,…,n}}

Giả sử vào các bước i ta tìm được lời giải bộ phận cấp i là (x1,…,xi), tức là đã đi qua đoạn đường , tương ứng với chi phí:

Để phát triển hành trình bộ phận này thành một hành trình đầy đủ, ta còn phải đi qua n-i+1 đoạn nữa, gồm n-I thành phố còn lại và đoạn quay lại .

Do chi phí mỗi một trong n-i+1 đoạn còn lại không nhỏ hơn Cmin, nên hàm đánh giá cận có thể xác định như sau:

g( ) =

* Điều kiện chấp nhận được của j là thành phố chưa đi qua.
* Xác định theo j bằng câu lệnh gán : = j

Cập nhật trạng thái mới: Daxet[j] = 1.

Cập nhật lại chi phí sau khi tìm tìm được : S=S+

* Cập nhật lời giải tối ưu:

Tính chi phí hành trình vừa tìm được:

Tong = S+;

Nếu (Tong <) thì

Lgtu = x;

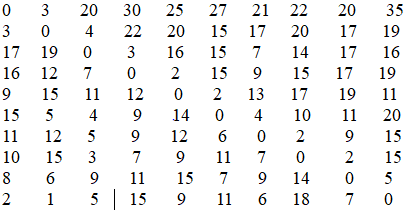
= Tong;

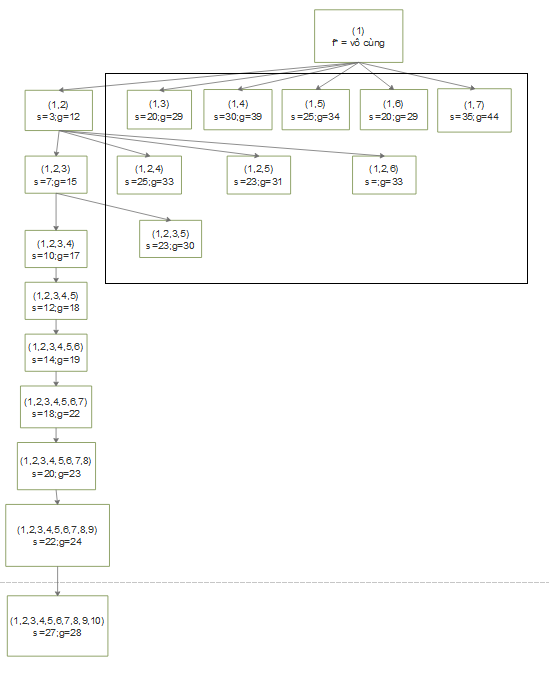
Thao tác hủy bỏ trạng thái: Daxet[j] = 0

Trả lại chi phí cũ : S=S-

1. **Thực hiện bài toán**

Ma trận chi phí:





g>=f\*

cắt các nhánh này

Cập nhật:

f\*=27+2=29

hành trình mới:

1->2->3->4->5->6->7->8->9->10

1. **Viết chương trình bài toán dạng C**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<conio.h>

using namespace std;

int c[10][10], x[10], xmin[10], chuaxet[10], k, n, s = 0, cmin, fmin = 10000;

//cmin là chi phi nho nhat giua 2 thanh pho

//s: chi phi

//xmin[] là chuoi xuat ra

void Nhap()

{

FILE \*f;

f = fopen("NguoiDuLich1.txt", "r");

fscanf(f, "%d", &n);

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

for (int j = 1; j <= n; j++)

fscanf(f, "%d", &c[i][j]);

fclose(f);

}

void KhoiTao()

{

cmin = c[1][2];

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

{

chuaxet[i] = 1;

for (int j = 1; j <= n; j++)

{

if (i != j&&c[i][j] != 0 && cmin > c[i][j])

cmin = c[i][j];

}

}

x[1] = 1;

chuaxet[1] = 0;

}

void Try(int k)

{

//printf("bat dau\n");

int g,tong,i,j;

for (j = 2; j <= n; j++)

{

if (chuaxet[j] == 1)

{

x[k] = j;

chuaxet[j] = 0;

s = s + c[x[k - 1]][x[k]];

//printf("%d\n", j);

if (k < n)

{

g = s + cmin\*(n - k + 1);

//printf("k=%3d,j=%3d,s=%3d,g=%3d\n",k,j,s,g);

if (g < fmin)

{

Try(k + 1);

}

}

if(k==n)

{

tong = s + c[x[n]][x[1]];

if (tong < fmin)

{

fmin = tong;

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

xmin[i] = x[i];//xmin[] là chuoi ket qua in ra

}

}

s = s - c[x[k - 1]][x[k]];

chuaxet[j] = 1;

}

}

}

void Xuat()

{

printf("Chi phi nho nhat la: %d", fmin);

printf("\nHanh trinh nho nhat la: ");

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

{

printf("%d", xmin[i]);

printf("->");

}

printf("1");

}

int main()

{

Nhap();

KhoiTao();

Try(2);

Xuat();

getchar();

return 0;

}

1. **Độ phức tạp của thuật toán**