# Bài toán người du lịch

Bởi:

Khoa CNTT ĐHSP KT Hưng Yên

**Bài toán:**

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

Gọi C $_{ij}$ là chi phí đi từ thành phố Ti đến thành phố Tj. Hãy tìm một hành trình thỏa yêu cầu bài toán sao cho chi phí là nhỏ nhất.

## Phân tích, thiết kế thuật toán

Gọi ∏ là một hoán vị của {1,.., n} thì một thành phố thỏa mãn yêu cầu bài toán có dạng : T∏(1) → T∏(2) → … → T∏(n) .

Nếu ta cố định một thành phố xuất phát, chẳng hạn T1, thì có (n-1)! Hành trình

Bài toán chuyển về dạng :

Tìm Min{f($a_2$,.., $a_n$ ) : ($a_2$,.., $a_n$ ) là hoán vị của {2,..,n}}.

Với f(a1,…, an) = $C_{1,\ a2} + C_{a2,a3} + \ldots + C_{an-1,an} + C_{an,1}$

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

Thiết kế thuật toán:

Input C = (Cij )

Output - x* = (x1,...,xn) // Hành trình tối ưu

- f* = f(x*) // Giá trị tối ưu

Try (i)

for (j = 1 -> n)

if( Chấp nhận được )

{

Xác định $x_i$ theo j;

Ghi nhận trạng thái mới;

if(i == n)

Cập nhật lời giải tối ưu;

else

{

Xác định cận $g(x_1,.., x_i)$

If $g(x_1,.., x_i) \leq f^*$ )

Try (i+1);

}

// Trả bài toán về trạng thái cũ

}

o Nếu ta cố định xuất phát tư T1, ta duyệt vòng lặp từ j = 2.

o Đánh giá nhánh cận :

Đặt : CMin = Min{Cij : i, j € {1,..,n}

Giả sử vào bước i ta tìm được lời giải bộ phận cấp i là (x1,..,xi ), tức là đã đi

qua đoạn đường T1 -> T2 -> . . . ->Ti , tương ứng với chi phí :

$S_i = C_{1, x2} + C_{x2,x3} + \ldots + C_{xn-1,xn} + C_{xn,1}$

Để phát triển hành trình bộ phận này thành một hành trình đầy đủ, ta còn phải đi qua n-i+1 đoạn đường nữa, gồm n-i thành phố còn lại và đoạn quay lại T1. 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 :

$g(x_1, \ldots, x_i) = S_i + (n - i + 1)CMin$

o Điều kiện chấp nhận được của j là thành phố Tj chưa đi qua.

Ta dùng một mảng logic Daxet[] để biểu diễn trạng thài này

Daxet[ j] = ? 1 ; T j đã được đi qua

0 ; T j chưa được đi qua

Mảng Daxet[ ] phải được bằng 0 tất cả.

o Xác định xi theo j bằng câu lệnh gán : xi = j

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

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

o Cập nhật lời giải tối ưu :

Tính chi phí hành trình vừa tìm được :

Tong = S + $C_{xn,1}$ ;

Nếu (Tong < f*) thì

Lgtu = x;

f* = Tong;

o Thao tác huỷ bỏ trạng thái : Daxet[j] = 0.

Trả lại chi phí cũ : S = S- $C_{x-1xi}$

Thủ tục nhánh cận viết lại như sau :

Try(i)

for (j = 2 -> n)

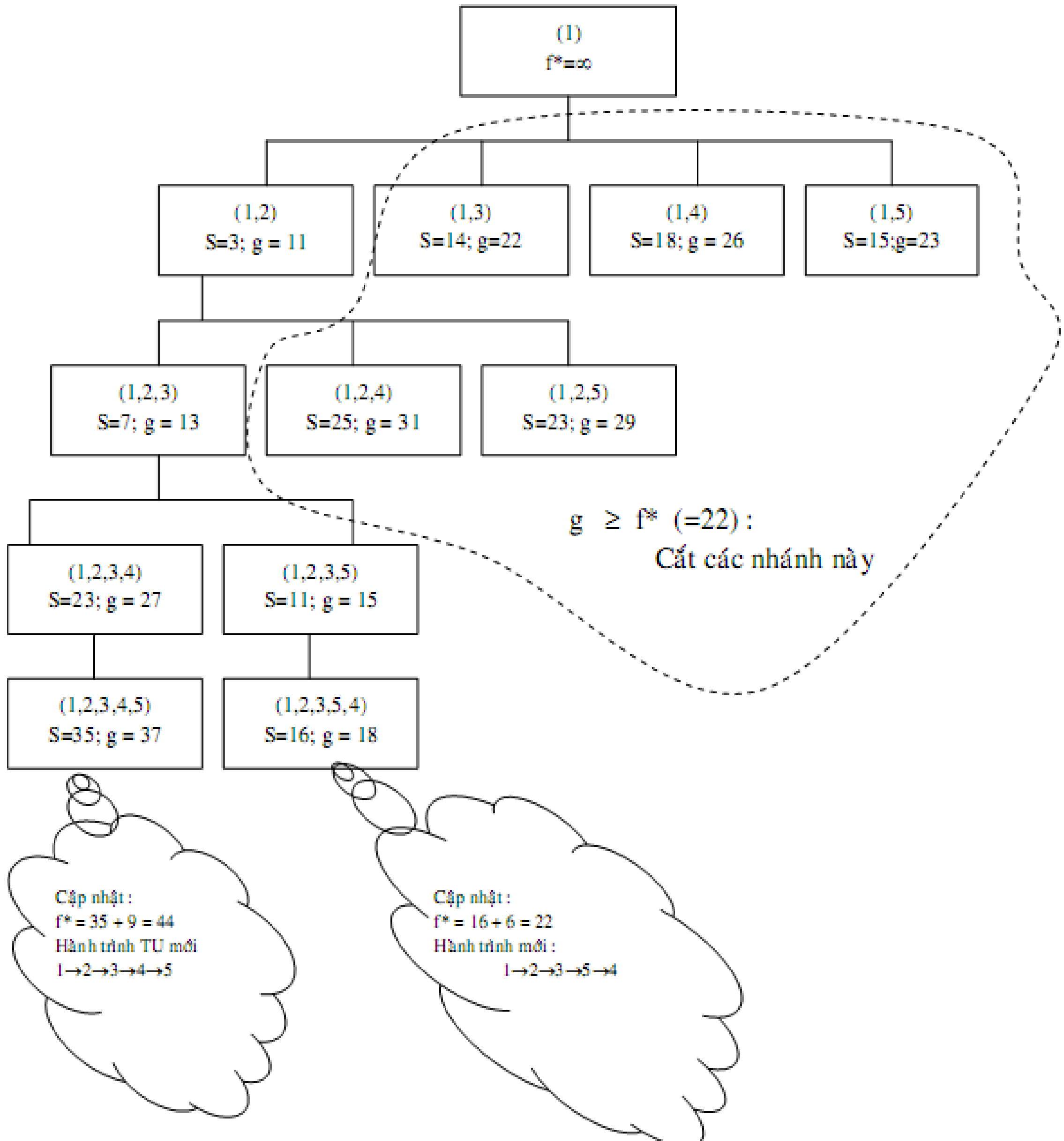
if(!Daxet[j])

```
{
x[i] = j; Daxet[j] = 1;
S = S + C[x[i-1]][x[i]];
if(i==n)
{
//Cap nhat toi uu
Tong = S + C[x[n]][x[1]];
if(Tong < f*)
{
Lgtu = x;
f* = Tong;
}
}
else
{
g = S + (n-i+1)*Cmin; //Danh gia can
if ( g < f*)
Try(i+1);
}
S = S - C[x[i-1]][x[i]];
Daxet[j] = 0;
}
```

Minh họa

Ma trận chi phí:

$$C = \begin{bmatrix} \infty & 3 & 14 & 18 & 15 \\ 3 & \infty & 4 & 22 & 20 \\ 17 & 9 & \infty & 16 & 4 \\ 6 & 2 & 7 & \infty & 12 \\ 9 & 15 & 11 & 5 & \infty \end{bmatrix}$$


(1)
f*=∞
(1,2)
S=3; g = 11
(1,3)
S=14; g=22
(1,4)
S=18; g = 26
(1,5)
S=15;g=23
(1,2,3)
S=7; g = 13
(1,2,4)
S=25; g = 31
(1,2,5)
S=23; g = 29
g ≥ f* (=22) :
Cắt các nhánh này
(1,2,3,4)
S=23; g = 27
(1,2,3,5)
S=11; g = 15
(1,2,3,4,5)
S=35; g = 37
(1,2,3,5,4)
S=16; g = 18
Cập nhật :
f* = 35 + 9 = 44
Hành trình TU mới
1→2→3→4→5
Cập nhật :
f* = 16 + 6 = 22
Hành trình mới :
1→2→3→5→4

## 4. Cài đặt