

BÀI TOÁN TỐI ƯU

Lý thuyết tổ hợp

Nội dung

- Giới thiệu bài toán.
- Phát biểu bài toán tối ưu cho các mô hình thực tế.
- Phương pháp liệt kê giải quyết bài toán tối ưu.
- Phương pháp nhánh cận giải quyết bài toán tối ưu.
- Kỹ thuật rút gọn giải quyết bài toán người du lịch.

Bài toán tối ưu

Phát biểu bài toán tối ưu tổ hợp

- Tìm cực tiểu (hay cực đại) của phiếm hàm $f(x) = \min(\max)$ với điều kiện $x \in D$, trong đó D là tập hữu hạn các phần tử.
 - Tập D gọi là tập các phương án của bài toán.
 - Mỗi phần tử $x \in D$ được gọi là một phương án.
 - Hàm $f(x)$ được gọi là **hàm mục tiêu** của bài toán.
 - Phương án $x^* \in D$ đem lại giá trị nhỏ nhất (lớn nhất) cho hàm mục tiêu được gọi là **phương án tối ưu**.
 - Giá trị $f^* = f(x^*)$ được gọi là **giá trị tối ưu** của bài toán.

Bài toán cái túi

- Một nhà thám hiểm cần **đem theo** một cái túi có trọng lượng không quá b . Có n đồ vật có thể đem theo.
- Đồ vật thứ j có **trọng lượng** a_j và **giá trị sử dụng** c_j ($j=1, 2, \dots, n$).
- Hỏi nhà thám hiểm cần đem theo những đồ vật nào để cho **tổng giá trị sử dụng là lớn nhất?**

Tập các phương án của bài toán

- Một phương án $x = (x_1, x_2, \dots, x_n)$,
 - $x_i = 1$: đồ vật thứ i được đem theo,
 - $x_i = 0$: đồ vật thứ i không được đem theo.
- Tập các xâu nhị phân $x = (x_1, \dots, x_n)$ còn phải thỏa mãn điều kiện **tổng trọng lượng không vượt quá b** .
 - Nói cách khác, tập phương án D của bài toán được xác định:

$$D = \left\{ x = (x_1, x_2, \dots, x_n) : \sum_{i=1}^n a_i x_i \leq b \right\}$$

Hàm mục tiêu của bài toán

- Với mỗi phương án $x \in D$:

- giá trị sử dụng các đồ vật đem theo là:

$$f(x) = \sum_{i=1}^n c_i x_i$$

- tổng trọng lượng đồ vật đem theo là:

$$g(x) = \sum_{i=1}^n a_i x_i$$

- Phát biểu dưới dạng bài toán tối ưu:

- Trong số các vector nhị phân độ dài n thoả mãn điều kiện $g(x) \leq b$, hãy tìm vector x^* để hàm mục tiêu $f(x)$ đạt giá trị lớn nhất.

- Nói cách khác, tìm $\max\{f(x): x \in D \text{ và } g(x) \leq b\}$.

Bài toán Người du lịch

- Một người du lịch muốn đi thăm quan n thành phố T_1, T_2, \dots, T_n .
- 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 một lần, rồi quay trở lại thành phố xuất phát.
- Biết c_{ij} là chi phí đi từ thành phố T_i đến thành phố T_j ($i, j=1, 2, \dots, n$).
- Hãy tìm hành trình với tổng chi phí là **nhỏ nhất** cho người du lịch.

Tập phương án của bài toán

$T(1) \rightarrow T(2) \rightarrow \dots \rightarrow T(n) \rightarrow T(1)$

- Mỗi hành trình có dạng $x = (x_1, x_2, \dots, x_n, x_1)$.
- $x = (x_1, x_2, \dots, x_n)$ là hoán vị của $1, 2, \dots, n$
 - $x_1 = 1$ ở vị trí đầu tiên và cuối cùng.
 - \Rightarrow Có $(n-1)!$ hành trình thực sự.
- Tập phương án của bài toán:

$$D = \{x = (x_1, x_2, \dots, x_n) : x_1 = 1; x_i \neq x_j : \forall i \neq j; i, j = 2, 3, \dots, n\}$$

Hàm mục tiêu của bài toán

- Hàm mục tiêu:

$$\begin{aligned} f(x) &= \sum_{i=1}^{n-1} C[x[i]][x[i+1]] + C[x[n]][x[1]] \\ &= C[x[1]][x[2]] + C[x[2]][x[3]] + \dots + C[x[n-1]][x[n]] + C[x[n]][x[1]] \end{aligned}$$

- Phát biểu dưới dạng bài toán tối ưu:

$$\text{Tìm } \min \{ f(x) : x \in D \}$$

Bài toán cho thuê máy

- Một ông chủ có một cái máy để cho thuê.
- Đầu tháng ông ta nhận được yêu cầu thuê máy của **m** khách hàng.
- Mỗi khách hàng **i** sẽ cho biết tập **N_i** các ngày trong tháng cần sử dụng máy ($i = 1, 2, \dots, m$).
- Ông chủ chỉ có quyền hoặc từ chối yêu cầu của khách hàng **i** , hoặc nếu nhận thì phải bố trí máy phục vụ khách hàng **i** đúng những ngày mà khách hàng này yêu cầu.
- Hỏi rằng ông chủ phải tiếp nhận các yêu cầu của khách thế nào để cho **tổng số ngày sử dụng máy là lớn nhất**.

Tập phương án của bài toán

- $I = \{1, 2, \dots, m\}$: tập chỉ số khách hàng.
- S : tập hợp các tập con của I . tập hợp các trường hợp có thể
- Tập hợp tất cả các phương án cho thuê máy là:

$$D = \left\{ J \subset S : N_k \cap N_p = \emptyset, \forall k \neq p \in J \right\}$$

ko dc trùng

J là 1 phương án

Hàm mục tiêu của bài toán

- Với mỗi phương án $J \in D$, mỗi **phương án cho thuê máy** là tổng số ngày sử dụng máy theo phương án đó.
- => Hàm mục tiêu của bài toán là:

$$f(J) = \sum_{j \in J} |N_j|$$

- Phát biểu dưới dạng bài toán tối ưu:

$$\max \{ f(J) : J \in D \}$$

Bài toán phân công công việc

- Có n công việc và n thợ. Biết c_{ij} là chi phí cần trả để thợ i hoàn thành công việc thứ j ($i, j = 1, 2, \dots, n$).
- Cần phải thuê thợ sao cho các công việc đều hoàn thành và mỗi thợ chỉ thực hiện một công việc, mỗi công việc chỉ do một thợ thực hiện.
- Hãy tìm cách thuê n nhân công sao cho **tổng chi phí thuê thợ là nhỏ nhất**.

Tập phương án của bài toán

- Mỗi phương án bố trí **thợ thực hiện** các công việc tương ứng với một hoán vị $\pi = (\pi(1), \pi(2), \dots, \pi(n))$.
- \Rightarrow **Tập các phương án** Π của bài toán là tập các hoán vị của $1, 2, \dots, n$.

Hàm mục tiêu của bài toán

- Ứng với mỗi phương án, **chi phí** theo phương án là:

$$f(\pi) = C_{\pi(1),1} + C_{\pi(2),2} + \dots + C_{\pi(n),n}$$

Công việc	Thời thực hiện
1	$\pi(1)$
2	$\pi(2)$
...	...
n	$\pi(n)$

- Phát biểu bài toán tối ưu tổ hợp:

$$\min \{ f(\pi) : \pi \in \Pi \}$$

Phương pháp duyệt toàn bộ

Duyệt toàn bộ

- Ứng với với mỗi phương án:
 - tính được giá trị của hàm mục tiêu;
 - so sánh giá trị của hàm mục tiêu tại tất cả các phương án đã được liệt kê để tìm ra phương án tối ưu.
- Nhược điểm:
 - Sự **bùng nổ của các cấu hình tổ hợp**.
 - **Ví dụ:** Để duyệt được $15! = 1\,307\,674\,368\,000$ cấu hình.
 - Trên máy tính có tốc độ 1 tỷ phép tính/giây, nếu mỗi hoán vị cần liệt kê mất khoảng 100 phép tính,
 - => cần khoảng thời gian là 130 767 giây (> 36 tiếng đồng hồ).

Thuật toán duyệt toàn bộ

Mã giả

optimal

Bước 1 (Khởi tạo):

$XOPT = \emptyset$; // Khởi tạo phương án tối ưu ban đầu

$FOPT = -\infty (+\infty)$; // Khởi tạo giá trị tối ưu ban đầu

Bước 2 (Lặp): âm tìm max, dương tìm min

for each $X \in D$ do { // lấy mỗi phần tử trên tập phương án

$S = f(X)$; // tính giá trị hàm mục tiêu cho phương án X

if ($FOPT < S$) { // Cập nhật phương án tối ưu

$FOPT = S$; // Giá trị tối ưu mới được xác lập

$XOPT = X$; // Phương án tối ưu mới

}

}

Bước 3 (Trả lại kết quả):

Return($XOPT, FOPT$);

Ví dụ: Giải bài toán cái túi bằng phương pháp duyệt toàn bộ

$$\begin{cases} F(X) = 4x_1 + 6x_2 + 3x_3 + 5x_4 + 2x_5 \rightarrow \max, \\ 9x_1 + 8x_2 + 5x_3 + 3x_4 + 2x_5 \leq 21, \\ x_j \in \{0,1\}, j = 1,2,3,4,5. \end{cases}$$

- Lời giải:
 - Gọi $A = (a_1, a_2, \dots, a_n)$, $C = (c_1, c_2, \dots, c_n)$ tương ứng với vector trọng lượng và giá trị sử dụng các đồ vật.

Tập các phương án của bài toán: $D = \left\{ X = (x_1, x_2, \dots, x_n) : \sum_{i=1}^n a_i x_i \leq b \right\}$

Hàm mục tiêu của bài toán: $f(X) = \sum_{i=1}^n c_i x_i$.

Ví dụ (tt)

$X=(x_1, x_2, x_3, x_4, x_5)$	$X \in D = \{x_1, x_2, x_3, x_4, x_5 : \sum_{i=1}^5 a_i x_i \leq 21\}$	$F(X) = \sum_{i=1}^5 c_i x_i = ?$
0, 0, 0, 0, 0	$0 \leq 21: X \in D$	0
0, 0, 0, 0, 1	$2 \leq 21: X \in D$	2
0, 0, 0, 1, 0	$3 \leq 21: X \in D$	5
0, 0, 0, 1, 1	$5 \leq 21: X \in D$	7
0, 0, 1, 0, 0	$5 \leq 21: X \in D$	3
0, 0, 1, 0, 1	$7 \leq 21: X \in D$	5
0, 0, 1, 1, 0	$8 \leq 21: X \in D$	8
0, 0, 1, 1, 1	$10 \leq 21: X \in D$	10
0, 1, 0, 0, 0	$8 \leq 21: X \in D$	6
0, 1, 0, 0, 1	$10 \leq 21: X \in D$	8
0, 1, 0, 1, 0	$11 \leq 21: X \in D$	11
0, 1, 0, 1, 1	$13 \leq 21: X \in D$	13
0, 1, 1, 0, 0	$13 \leq 21: X \in D$	9
0, 1, 1, 0, 1	$15 \leq 21: X \in D$	11
0, 1, 1, 1, 0	$16 \leq 21: X \in D$	14
0, 1, 1, 1, 1	$18 \leq 21: X \in D$	16
1, 0, 0, 0, 0	$9 \leq 21: X \in D$	4
1, 0, 0, 0, 1	$11 \leq 21: X \in D$	6
1, 0, 0, 1, 0	$12 \leq 21: X \in D$	9
1, 0, 0, 1, 1	$14 \leq 21: X \in D$	9
1, 0, 1, 0, 0	$14 \leq 21: X \in D$	10
1, 0, 1, 0, 1	$16 \leq 21: X \in D$	9
1, 0, 1, 1, 0	$16 \leq 21: X \in D$	12
1, 0, 1, 1, 1	$19 > 21: X \notin D$	14
1, 1, 0, 0, 0	$17 \leq 21: X \in D$	10
1, 1, 0, 0, 1	$19 \leq 21: X \in D$	12
1, 1, 0, 1, 0	$20 \leq 21: X \in D$	15
1, 1, 0, 1, 1	$22 > 21: X \notin D$	∞
1, 1, 1, 0, 0	$22 > 21: X \notin D$	∞
1, 1, 1, 0, 1	$24 > 21: X \notin D$	∞
1, 1, 1, 1, 0	$25 > 21: X \notin D$	∞
1, 1, 1, 1, 1	$27 > 21: X \notin D$	∞
$X_{OPT}=(0,1,1,1,1); F_{OPT} = 16$		

Thuật toán nhánh cận

- Tìm $\min\{f(x) : x \in D\}$

- D là tập hữu hạn phần tử. $A_1=\{0,1\}$

$$D = \{ x = (x_1, x_2, \dots, x_n) \in A_1 \times A_2 \times \dots \times A_n ; x \text{ thoả mãn tính chất } P \}$$

- A_1, A_2, \dots, A_n là các tập hữu hạn,

- P là tính chất cho trên tích đề các $A_1 \times A_2 \times \dots \times A_n$

Thuật toán nhánh cận (tt)

- (a_1, a_2, \dots, a_k) : **phương án bộ phận** cấp k .
- Tìm được:

$$g(a_1, a_2, \dots, a_k) \leq \min \{f(x) : x \in D, x_i = a_i, i = 1, 2, \dots, k\} \quad (*)$$

với mọi lời giải bộ phận (a_1, a_2, \dots, a_k) , và với mọi $k = 1, 2, \dots$

- Ý nghĩa:
 - Giá trị của hàm tại phương án bộ phận (a_1, a_2, \dots, a_k) không vượt quá giá trị nhỏ nhất của hàm mục tiêu bài toán trên tập con các phương án:
- $$D(a_1, a_2, \dots, a_k) = \{x \in D : x_i = a_i, i = 1, 2, \dots, k\}$$
- Nói cách khác, $g(a_1, a_2, \dots, a_k)$ là **cận dưới** của tập $D(a_1, a_2, \dots, a_k)$.

Thuật toán nhánh cận (tt)

- Gọi \bar{x} là phương án với giá trị hàm mục tiêu **nhỏ nhất** trong số các phương án đã duyệt,
- Gọi $\bar{f} = f(\bar{x})$ là kỷ lục; \bar{x} là phương án tốt nhất hiện có
- Nếu $g(a_1, a_2, \dots, a_k) > \bar{f}$
 - $\Rightarrow \bar{f} < g(a_1, a_2, \dots, a_k) \leq \min \{ f(x) : x \in D, x_i = a_i, i=1, 2, \dots, k \}$
 - \Rightarrow tập con các phương án của bài toán $D(a_1, a_2, \dots, a_k)$ chắc chắn không chứa phương án tối ưu.
 - \Rightarrow không cần phải phát triển phương án bộ phận (a_1, a_2, \dots, a_k) .
(tức là có thể loại bỏ các phương án trong tập $D(a_1, a_2, \dots, a_k)$ khỏi quá trình tìm kiếm.

Thuật toán quay lui dựa vào hàm đánh giá cận

Procedure Try(k) {

*/*Phát triển phương án bộ phận $(a_1, a_2, \dots, a_{k-1})$ theo thuật toán quay lui có kiểm tra cận dưới trước khi tiếp tục phát triển phương án*/*

for ($a_k \in A_k$) {

if (chấp nhận a_k) {

$x_k = a_k$;

if ($k == n$)

< cập nhật kỷ lục >;

else if ($g(a_1, a_2, \dots, a_k) \leq \bar{f}$)

Try ($k+1$); phát triển phương án

}

}

}

Thuật toán nhánh cận

Procedure Nhanh_Can() {

$\bar{f} = +\infty$; / Nếu biết một phương án \bar{x} nào đó thì có thể đặt $\bar{f} = f(\bar{x})$. */*

Try(1); //Thực hiện thuật toán quay lui

if ($\bar{f} \leq +\infty$)

< \bar{f} là giá trị tối ưu , \bar{x} là phương án tối ưu >;

else

< bài toán không có phương án>;

}

Thuật toán nhánh cận – Nhận xét

- Xây dựng hàm **g** nên đạt được những điều kiện:
 - Việc tính giá trị của **g** phải đơn giản hơn việc giải bài toán tổ hợp trong vế phải của (*). [slide 23](#)
 - Giá trị của $g(a_1, a_2, \dots, a_k)$ phải sát với giá trị vế phải của (*).

Bài toán cái túi

- Có **n loại** đồ vật, đồ vật thứ **j** có trọng lượng a_j và giá trị sử dụng c_j ($j = 1, 2, \dots, n$).
- Cần chắt các đồ vật này vào một cái túi có trọng lượng là **b** sao cho **tổng giá trị sử dụng của các đồ vật đựng trong túi là lớn nhất.**

Mô hình toán học của bài toán cái túi (biến nguyên)

$$f^* = \max \left\{ f(x) = \sum_{j=1}^n c_j x_j : \sum_{j=1}^n a_j x_j \leq b, x_j \in Z^+, j = 1, 2, \dots, n \right\}, (1)$$

Trong đó Z^+ là tập các số nguyên không âm.

Ký hiệu D là tập các phương án của bài toán (1):

$$D = \left\{ x = (x_1, x_2, \dots, x_n) : \sum_{j=1}^n a_j x_j \leq b, x_j \in Z^+, j = 1, 2, \dots, n \right\}.$$

$$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \dots \geq \frac{c_n}{a_n}$$

(2)

Mô hình toán học của bài toán cái túi (biến liên tục)

- xây dựng hàm tính **cận trên**:

– Tìm:

$$g^* = \max \left\{ \sum_{j=1}^n c_j x_j : \sum_{j=1}^n a_j x_j \leq b, x_j \geq 0, j = 1, 2, \dots, n \right\}. \quad (3)$$

- **Mệnh đề:**

– Phương án tối ưu của bài toán (3) là vector

$$\bar{x} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$$

với các thành phần được xác định bởi công thức:

$$\bar{x}_1 = \frac{b}{a_1}, \bar{x}_2 = \bar{x}_3 = \dots = \bar{x}_n = 0 \text{ và giá trị tối ưu là } g^* = \frac{c_1 b}{a_1}.$$

b: trọng lượng túi

Chứng minh mệnh đề

- Xét $x = (x_1, x_2, \dots, x_n)$ là một phương án tùy ý của bài toán (3).
 - Khi đó từ bất đẳng thức (2) và do $x_j \geq 0$, ta suy ra:

$$c_j x_j \leq (c_1 / a_1) a_j x_j, j = 1, 2, \dots, n$$

- $\Rightarrow \sum_{j=1}^n c_j x_j \leq \sum_{j=1}^n \left(\frac{c_1}{a_1}\right) a_j x_j = \left(\frac{c_1}{a_1}\right) \sum_{j=1}^n a_j x_j \leq \frac{c_1}{a_1} b = g^* .$

(Mệnh đề được chứng minh).

Bài toán cái túi (tt)

- Giả sử có phương án bộ phận cấp k : (u_1, u_2, \dots, u_k)
 - Giá trị sử dụng của các đồ vật đang có trong túi:

$$\hat{o}_k = c_1 u_1 + c_2 u_2 + \dots + c_k u_k$$

- Trọng lượng còn lại của túi: $b_k = b - (a_1 u_1 + a_2 u_2 + \dots + a_k u_k)$

- Ta có:

$$\max \{f(x) : x \in D, x_j = u_j, j = 1, 2, \dots, n\}$$

$$= \max \left\{ \hat{o}_k + \sum_{j=k+1}^n c_j x_j : \sum_{j=k+1}^n a_j x_j \leq b_k, x_j \in \mathbb{Z}^+, j = k+1, k+2, \dots, n \right\}$$

$$\leq \hat{o}_k + \max \left\{ \sum_{j=k+1}^n c_j x_j : \sum_{j=k+1}^n a_j x_j \leq b_k, x_j \geq 0, j = k+1, k+2, \dots, n \right\}$$

$$= \hat{o}_k + \frac{c_{k+1} b_k}{a_{k+1}}$$

=> Cận trên cho phương án bộ phận (u_1, u_2, \dots, u_k) :

$$g(u_1, u_2, \dots, u_k) = \hat{o}_k + \frac{c_{k+1} b_k}{a_{k+1}}.$$

Thuật toán nhánh cận giải bài toán cái túi

Thuật toán Branch_And_Bound (i) {

t = (b - b_i)/A[i]; // Khởi tạo số lượng đồ vật thứ i

for (j = t; j ≥ 0; j--){

x[i] = j; // Lựa chọn x[i] là j;

b_i = b_i + a_ix_i; // Trọng lượng túi cho bài toán bộ phận thứ i.

σ_i = σ_i + c_ix_i; // Giá trị sử dụng cho bài toán bộ phận thứ i

If (i == n) <Cập nhật kỷ lục>;

*else if (σ_i + (c_{i+1}*b_i)/a_{i+1} > FOPT) //Nhánh cận được triển khai tiếp theo*

Branch_And_Bound(i+1);

b_i = b_i - a_ix_i;

σ_i = σ_i - c_ix_i;

}

}

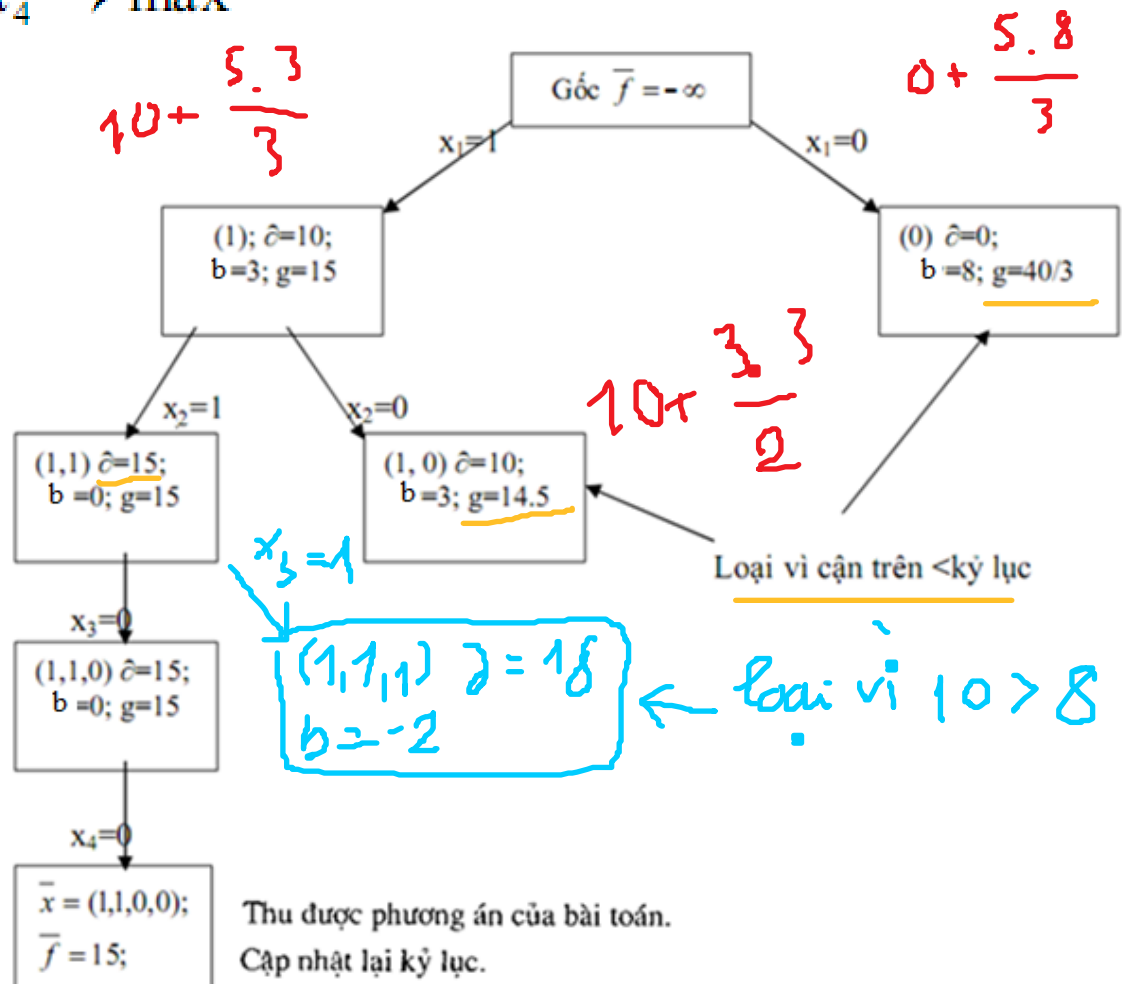
Ví dụ. Giải bài toán cái túi theo thuật toán nhánh cận

$$f(x) = 10x_1 + 5x_2 + 3x_3 + 6x_4 \rightarrow \max$$

$$5x_1 + 3x_2 + 2x_3 + 4x_4 \leq 8$$

$$x_j \in \mathbb{Z}^+, j = 1, 2, 3, 4.$$

$$\begin{aligned} \partial_k &= c_1 u_1 + c_2 u_2 + \dots + c_k u_k \\ b_k &= b - (a_1 u_1 + a_2 u_2 + \dots + a_k u_k) \\ g(u_1, u_2, \dots, u_k) &= \partial_k + \frac{c_{k+1} b_k}{a_{k+1}}. \end{aligned}$$



Chương trình giải bài toán cái túi theo thuật toán nhánh cận

```
#include <iostream.h>
#include <stdio.h>
#include <conio.h>
#define MAX 100
int A[MAX], C[MAX], F[MAX][MAX];
int XOPT[MAX], X[MAX];
int n, b, ind;
float W, FOPT=-32000, cost, weight=0;
```

```
FILE *fp;
void Init(void){
    fp = fopen("caituil.in", "r");
    fscanf(fp, "%d%d", &n, &b);
    cout<<"\n So luong do vat:"<<n;
    cout<<"\n Trong luong tui:"<<b;
    for( int i=1; i<=n; i++)
        fscanf(fp, "%d%d", &A[i], &C[i]);
    cout<<"\n Vector trong luong:";
    for( i=1; i<=n; i++)
        cout<<A[i]<<" ";
    cout<<"\n Vector gia tri su dung:";
    for( i=1; i<=n; i++)
        cout<<C[i]<<" ";
    fclose(fp);
}
```

+ hàng số'
túi
→ 10 5 3 ...

Chương trình giải bài toán cái túi theo thuật toán nhánh cận (tt)

```
void Result(void) {
    cout<<"\n Ket qua toi uu:"<<FOPT;
    cout<<"\n Phuong an toi uu:";
    for(int i=1; i<=n; i++)
        cout<<XOPT[i]<<" ";
}
void Branch_And_Bound( int i) {
    int j, t =(b-weight)/A[i];
    for(j=t; j>=0; j--){      X[i] = j;
        weight = weight+A[i]*X[i];
        cost = cost + C[i]*X[i];
        if (i==n) Update Kyluc();
        else if ( cost+C[i+1]*(b-weight)/A[i+1]>FOPT)
            Branch_And_Bound(i+1);
        weight = weight-A[i]*X[i];
        cost = cost - C[i]*X[i];
    }
}
void main(void) {
    Init();
    Branch_And_Bound(1);
    Result();
}
```

Bài toán Người du lịch

- Một người du lịch muốn đi thăm quan n thành phố T_1, T_2, \dots, T_n .
- 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 một lần, rồi quay trở lại thành phố xuất phát.
- Biết c_{ij} là chi phí đi từ thành phố T_i đến thành phố T_j ($i = 1, 2, \dots, n$), hãy tìm hành trình với tổng chi phí là nhỏ nhất

Bài toán Người du lịch (tt) -> Bài toán tìm cực tiểu phiếm hàm

- Cố định thành phố xuất phát là T_1 .
 - Bài toán Người du lịch được đưa về bài toán *Tìm cực tiểu của phiếm hàm*:

$$f(x_2, \dots, x_n) = c[\underline{1}, x_2] + c[x_2, x_3] + \dots + c[x_{n-1}, x_n] + c[x_n, \underline{1}] \rightarrow \min$$

- Với điều kiện:

(x_2, x_3, \dots, x_n) là hoán vị của các số $2, 3, \dots, n$.

$c_{\min} = \min \{c[i, j], i, j = 1, 2, \dots, n; i \neq j\}$ là chi phí đi lại nhỏ nhất giữa các thành phố.

Bài toán Người du lịch (tt) – Tìm cận dưới cho phương án bộ phận

- Giả sử ta đang có phương án bộ phận (u_1, u_2, \dots, u_k) ,
- \Rightarrow hành trình bộ phận qua k thành phố:

$$T_1 \rightarrow T(u_2) \rightarrow \dots \rightarrow T(u_{k-1}) \rightarrow T(u_k)$$

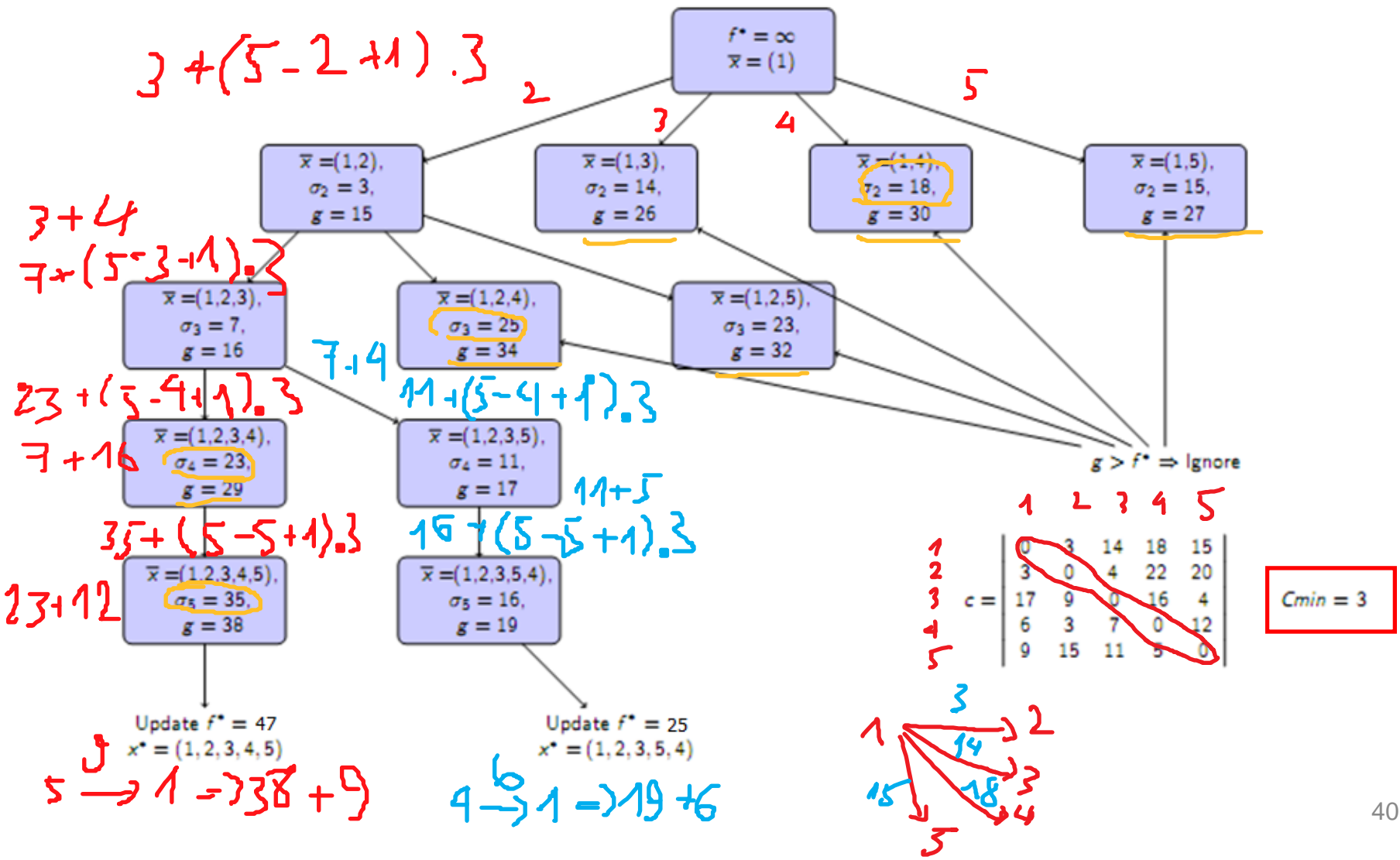
- Chi phí phải trả theo hành trình bộ phận này:

$$\partial = c[1, u_2] + c[u_2, u_3] + \dots + c[u_{k-1}, u_k] .$$

- Hành trình đầy đủ: phải đi qua **$n-k+1$ đoạn đường** nữa.
 - Chi phí phải trả để đi qua mỗi đoạn còn lại đều ***không nhiều hơn c_{\min}***
- \Rightarrow **Cận dưới** cho phương án bộ phận (u_1, u_2, \dots, u_k) :

$$g(u_1, u_2, \dots, u_k) = \partial + (n - k + 1) c_{\min} .$$

Ví dụ. Giải bài toán người du lịch bằng thuật toán nhánh cận



Chương trình giải bài toán người đi du lịch theo thuật toán nhánh cận

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <io.h>
#define MAX 20
int n, P[MAX], B[MAX], C[20][20], count=0;
int A[MAX], XOPT[MAX];
int can, cmin, fopt;

void Read_Data(void){
    int i, j; FILE *fp;
    fp = fopen("dulich.in", "r");
    fscanf(fp, "%d", &n);
    printf("\n So thanh pho: %d", n);
    printf("\n Ma tran chi phi:");
    for (i=1; i<=n; i++){
        printf("\n");
        for(j=1; j<=n; j++){
            fscanf(fp, "%d", &C[i][j]);
            printf("%5d", C[i][j]);
        }
    }
}
```

Chương trình giải bài toán người đi du lịch theo thuật toán nhánh cận (tt)

```
int Min_Matrix(void){
    int min=1000, i, j;
    for(i=1; i<=n; i++){
        for(j=1; j<=n; j++){
            if (i!=j && min>C[i][j])
                min=C[i][j];
        }
    }
    return(min);
}

void Init(void){
    int i;
    cmin=Min_Matrix();
    fopt=32000;can=0; A[1]=1;
    for (i=1;i<=n; i++)
        B[i]=1;
}
```

Chương trình giải bài toán người đi du lịch theo thuật toán nhánh cận (tt)

```
void Result(void){
    int i;
    printf("\n Hanh trinh toi uu %d:", fopt);
    printf("\n Hanh trinh:");
    for(i=1; i<=n; i++)
        printf("%3d->", XOPT[i]);
    printf("%d",1);
}
void Swap(void){
    int i;
    for(i=1; i<=n;i++)
        XOPT[i]=A[i];
}
void Update_Kyluc(void){
    int sum;
    sum=can+C[A[n]][A[1]];
    if(sum<fopt) {
        Swap();
        fopt=sum;
    }
}
```

Chương trình giải bài toán người đi du lịch theo thuật toán nhánh cận (tt)

```
void Try(int i){
    int j;
    for(j=2; j<=n; j++){
        if(B[j]){
            A[i]=j; B[j]=0;
            can=can+C[A[i-1]][A[i]];
            if (i==n) Update_Kyluc();
            else if( can + (n-i+1)*cmin<fopt){
                count++;
                Try(i+1);
            }
            B[j]=1; can=can-C[A[i-1]][A[i]];
        }
    }
}

void main(void){
    clrscr(); Read_Data(); Init();
    Try(2); Result();
    getch();
}
```

Kỹ thuật rút gọn cho bài toán tối ưu

Kỹ thuật rút gọn giải quyết bài toán người du lịch

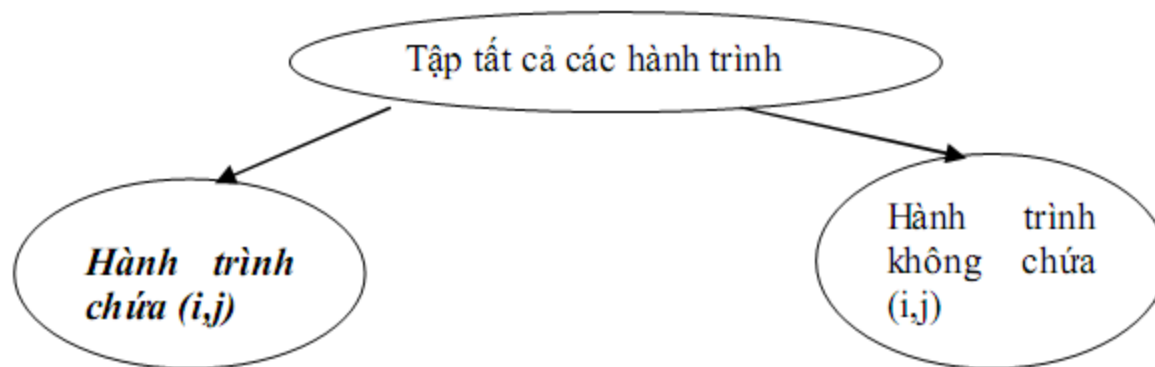
- Xét bài toán người du lịch như đã được phát biểu.
 - Gọi $C = \{c_{ij}: i, j = 1, 2, \dots, n\}$ là ma trận chi phí.
 - Mỗi hành trình của người du lịch

$$T_{\pi(1)} \rightarrow T_{\pi(2)} \rightarrow \dots \rightarrow T_{\pi(n)} \rightarrow T_{\pi(1)}$$

có thể viết lại dưới dạng

$$(\pi(1), \pi(2), \pi(2), \pi(3), \dots, \pi(n-1), \pi(n), \pi(n), \pi(1))$$

- trong đó mỗi thành phần $(\pi(j-1), \pi(j))$ được gọi là một cạnh của hành trình.



Thủ tục rút gọn

- **Nhận xét:** Tổng chi phí của một hành trình của người du lịch sẽ chứa đúng một phần tử của mỗi dòng và đúng một phần tử của mỗi cột trong ma trận chi phí C
 - Nếu ta cộng hay trừ bớt mỗi phần tử của một dòng (hay cột) của ma trận C đi cùng một số α thì độ dài của tất cả các hành trình đều giảm đi α vì thế hành trình tối ưu cũng sẽ không bị thay đổi.
 - Vì vậy, nếu ta tiến hành bớt đi các phần tử của mỗi dòng và mỗi cột đi một hằng số sao cho ta thu được một ma trận gồm các phần tử không âm mà **trên mỗi dòng, mỗi cột đều có ít nhất một số 0**, thì tổng các số trừ đó cho ta cận dưới của mọi hành trình.
- **Thủ tục rút gọn:**
 - Các hằng số trừ ở mỗi dòng (cột) sẽ được gọi là hằng số rút gọn theo dòng (cột),
 - Ma trận thu được được gọi là ma trận rút gọn.

Tính cận dưới dựa trên Thủ tục rút gọn

```
float Reduce( float A[][max], int k) {  
    sum = 0;  
    for (i = 1; i ≤ k; i++) {  
        r[i] = < phần tử nhỏ nhất của dòng i >;  
        if (r[i] > 0 ) {  
            < Bớt mỗi phần tử của dòng i đi r[i] >;  
            sum = sum + r[i];  
        }  
    }  
    for (j = 1; j ≤ k; j++) {  
        s[j] := < Phần tử nhỏ nhất của cột j >;  
        if (s[j] > 0 )  
            < Bớt mỗi phần tử của cột j đi s[j] >;  
        sum = sum + S[j];  
    }  
    return(sum);  
}
```

→ **الخطوات**

Ví dụ. Ma trận chi phí với $n=6$ thành phố

Handwritten notes: Tuyến (green), Chưa xử (red), Min (blue), X (blue), 1 (blue circle), 2 (blue circle)

	1	2	3	4	5	6	$ r[i]$								
1	∞	<u>3</u>	93	13	33	9	-3 \rightarrow	∞	0	90	10	30	6		
2	<u>4</u>	∞	77	42	21	16	-4 \rightarrow	0	∞	73	38	17	12		
3	45	17	∞	36	16	28	-16 \rightarrow	29	1	∞	20	0	12		
4	39	90	80	∞	56	7	-7 \rightarrow	32	83	73	∞	49	0		
5	28	46	88	33	∞	25	-25 \rightarrow	3	21	63	<u>8</u>	∞	0		
6	3	88	18	46	92	∞	-3 \rightarrow	0	85	<u>15</u>	43	35	∞		
	0	0	-15	-8	0	0			1	2	3	4	5	6	

1	∞	0	75	2	30	6
2	0	∞	58	30	17	12
3	29	1	∞	12	0	12
4	32	83	58	∞	49	0
5	3	21	48	0	∞	0
6	0	85	0	35	89	∞

=> Cập dưới cho tất cả các hành trình: **81**

$$3 + 4 + 16 + 7 + 125 + 3 + 15 + 8$$

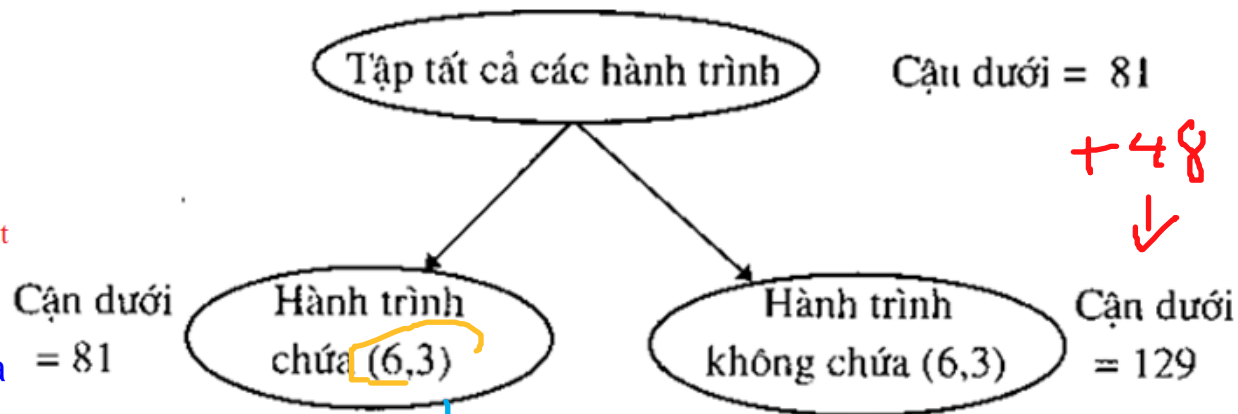
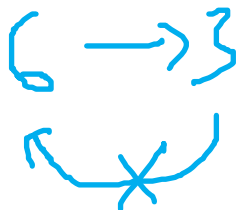
-15 -8

Ví dụ. Phân tập các phương án

	1	2	3	4	5	6
1	∞	0	75	2	30	6
2	0	∞	58	30	17	12
3	29	1	∞	12	0	12
4	32	83	58	∞	49	0
5	3	21	48	0	∞	0
6	0	85	0	35	89	∞

min trong hàng → min trong cột

xác định các vị trí 0
 tìm min của hàng và cột trừ vị trí 0 đó ra
 cộng 2 cái min lại
 tìm max của tổng đó
 vị trí của 0 mà ta chọn cũng là hành trình



	1	2	4	5	6
1	∞	0	2	30	6
2	0	∞	30	17	12
3	29	1	12	0	∞
4	32	83	∞	49	0
5	3	21	0	∞	0

	1	2	3	4	5	6
1	∞	0	27	2	30	6
2	0	∞	10	30	17	12
3	29	1	∞	12	0	12
4	32	83	10	∞	49	0
5	3	21	0	0	∞	0
6	0	85	∞	35	89	∞

Hàng số rút gọn: 48

-48

Thủ tục chọn cạnh phân nhánh (r,c)

void BestEdge(A, k, r, c, beta)

Đầu vào: Ma trận rút gọn A kích thước $k \times k$

Kết quả ra: Cạnh phân nhánh (r,c) và tổng hằng số rút gọn theo dòng r cột c là beta.

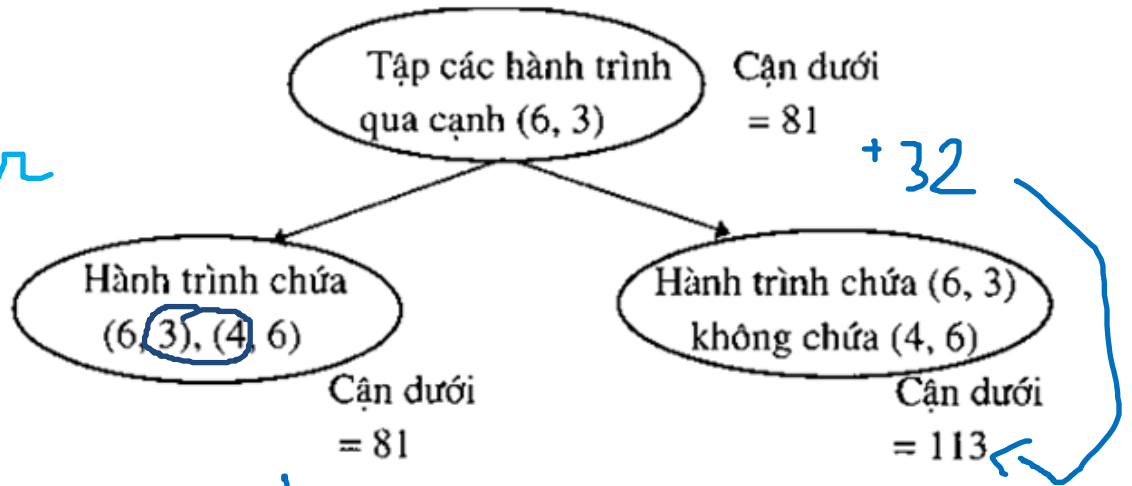
```
{  
    beta = -∞;  
    for ( i = 1; i ≤ k; i++) {  
        for ( j = 1; j ≤ k; j++) {  
            if (A[i,j] == 0) {  
                minr = <phần tử nhỏ nhất trên dòng i khác với A[i,j];  
                minc = <phần tử nhỏ nhất trên cột j khác với A[i,j];  
                total = minr + minc;  
                if (total > beta) {  
                    beta = total;  
                    r = i; /* Chỉ số dòng tốt nhất*/  
                    c = j; /* Chỉ số cột tốt nhất*/  
                }  
            }  
        }  
    }  
}
```

Phân nhánh tiếp theo (ưu tiên nhánh bên trái)

	1	2	4	5	6	
1	∞	0	2	30	6	2
2	0	∞	30	17	12	12
3	29	1	12	0	∞	1
4	32	83	∞	49	0	32
5	3	21	0	∞	0	0

min trong hàng: 3 1 2 17 0

min trong cột: 0

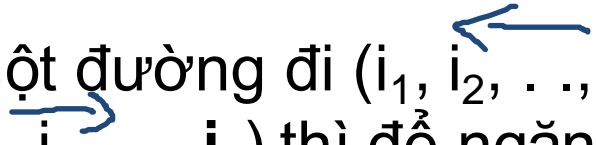


	1	2	4	5
1	∞	0	2	30
2	0	∞	30	17
3	29	1	∞	0
5	3	21	0	∞

	1	2	4	5	6
1	∞	0	2	30	6
2	0	∞	30	17	12
3	29	1	12	0	∞
4	0	51	∞	17	∞
5	3	21	0	∞	0

Hằng số rút gọn: 32

Ngăn cấm tạo thành hành trình con

- Khi phân nhánh dựa vào cạnh (i_u, j_v) ta phải thêm cạnh này vào danh sách các cạnh của node **bên trái nhất**.
- Nếu i_u là **đỉnh cuối** của một đường đi (i_1, i_2, \dots, i_u) và j_v là **đỉnh đầu** của đường đi (j_v, j_2, \dots, j_k) thì để ngăn ngừa khả năng tạo thành **hành trình con** ta phải **cấm cạnh** (j_k, i_1) .
- Để tìm i_1 ta đi ngược từ i_u , để tìm j_k ta đi xuôi từ j_v theo danh sách các cạnh đã được kết nạp vào hành trình.

Phân nhánh tiếp theo - cạnh (2,1)

4 → 2 → 3

2 → 1

→

	1	2	4	5
1	∞	0	2	30
2	0	∞	30	17
3	29	1	∞	0
5	3	21	0	∞

min theo dòng

min theo cột

=> Hằng số rút gọn: $17+3=20$

=> Cận dưới của nhánh phải (không chứa cạnh (2,1)) là: $81+20=101$.

2 → 1

1 * 2

	1	2	4	5
1	∞	2	30	
3	1	∞	0	
5	21	0	∞	

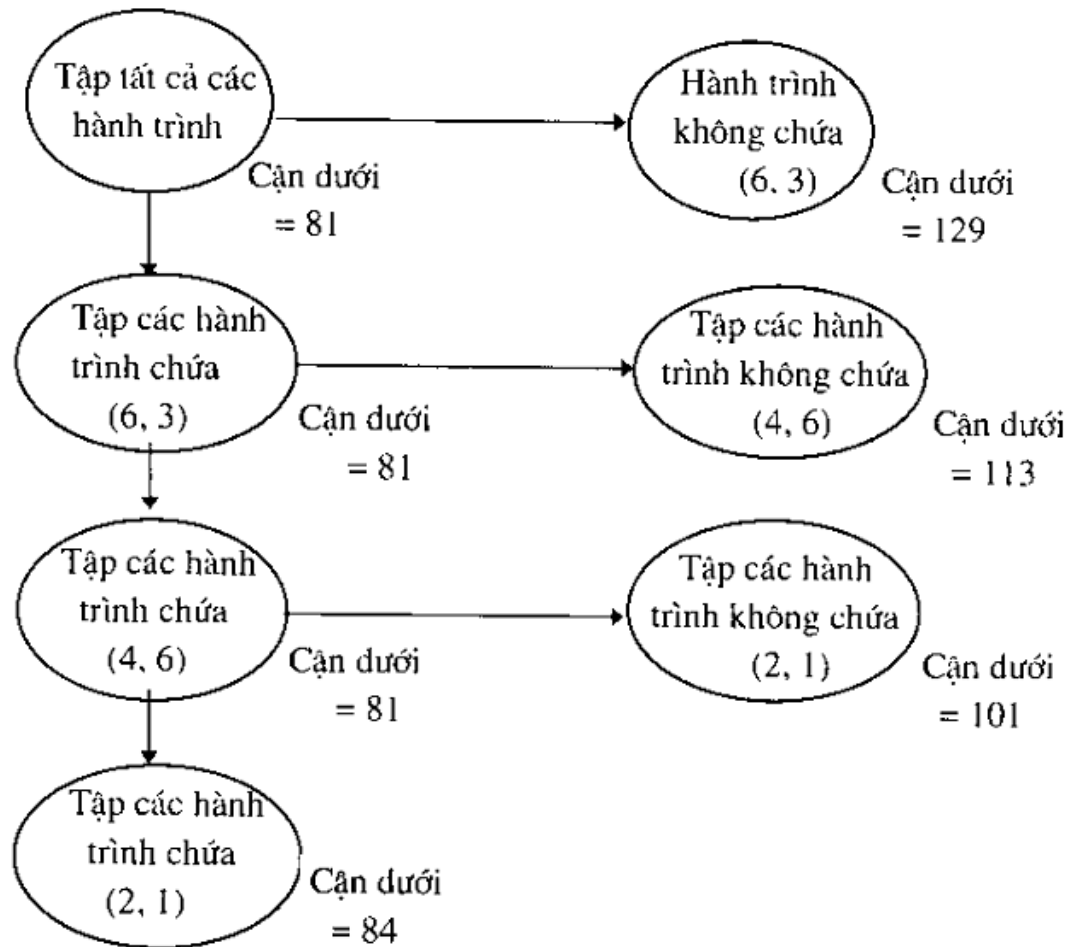
=> (rút gọn)

	2	4	5
1	∞	0	28
3	0	∞	0
5	20	0	∞

=> Cận dưới của nhánh trái (chứa cạnh (2,1)) là: $81 + 1 + 2 = 84$.

Cây tìm kiếm

(cho đến phân nhánh theo cạnh (2,1))



1 → 4 → 6 → 3 2 → 1

Cây tìm kiếm

(cho đến phân nhánh theo cạnh (1,4))

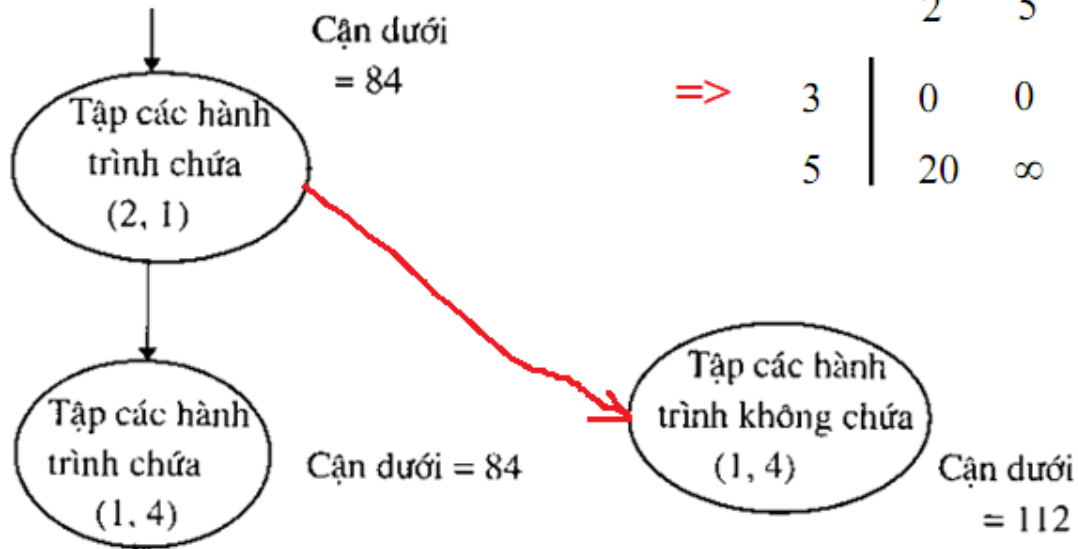
k^o chứa

	2	4	5
1	∞	0	28
3	0	∞	0
5	20	0	∞

min theo hàng → ∞
min theo cột →

⇒ Hằng số rút gọn: $28+0=28$.

	2	4	5
1	∞	0	28
3	0	∞	0
5	20	0	∞



⇒

	2	5
3	0	0
5	20	∞

có chứa

Rút gọn ma trận (sau phân nhánh)

$$\begin{array}{c|cc|} & 2 & 5 \\ \hline 3 & 0 & 0 \\ \hline 5 & 20 & \infty \\ \hline \end{array} \Rightarrow \begin{array}{c|cc|} & 2 & 5 \\ \hline 3 & \infty & 0 \\ \hline 5 & 20 & \infty \\ \hline \end{array}$$

(Red arrow from 0 to ∞ with label -3)

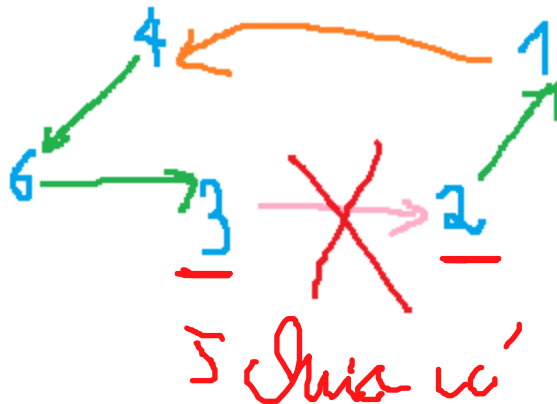
Hiện tại, có 2 đường đi:
 (2,1): đỉnh cuối là 1
 (4,6,3): đỉnh đầu là 4

$$\begin{array}{c|cc|} & 2 & 5 \\ \hline 3 & \infty & 0 \\ \hline 5 & 0 & \infty \\ \hline \end{array}$$

(Red arrow from 20 to 0 with label \Rightarrow (rút gọn))

\Rightarrow Khi phân nhánh chứa cạnh (1,4) thì phải tránh tạo hành trình con: Cấm cạnh (3,2) - tức là gán $C(3,2) = \infty$

Hằng số rút gọn: 20



Kết nạp nốt cạnh (3, 5), (5, 2) vào chu trình và thu được hành trình: **1, 4, 6, 3, 5, 2, 1** với chi phí là **104**.

Kết nạp 2 cạnh còn lại

- Ma trận cuối cùng rút gọn chỉ có thể ở một trong hai dạng sau:

		w	x			w	x	
u		∞	0		u		0	∞
v		0	∞		v		∞	0

- Xác định 2 cạnh cần nạp vào hành trình:

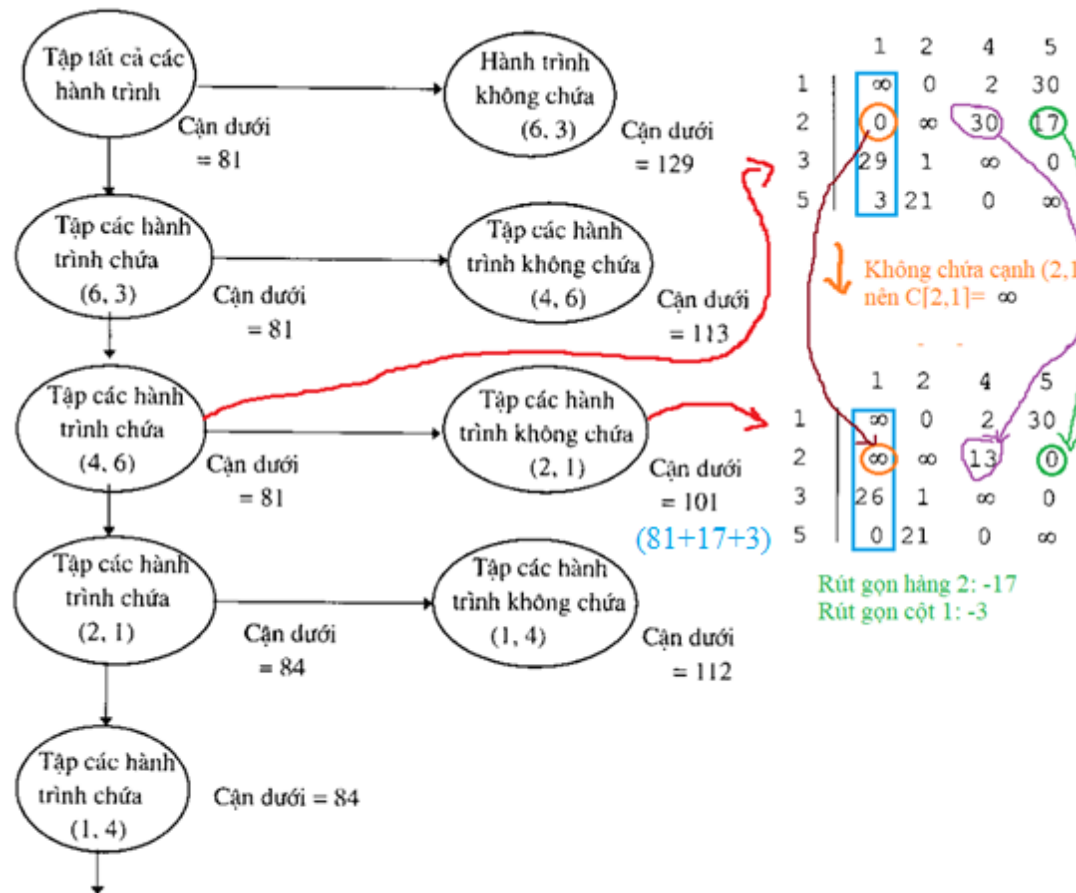
if $A[1, 1] = \infty$ then

<Kết nạp cạnh $(u, x), (v, w)$ >

else

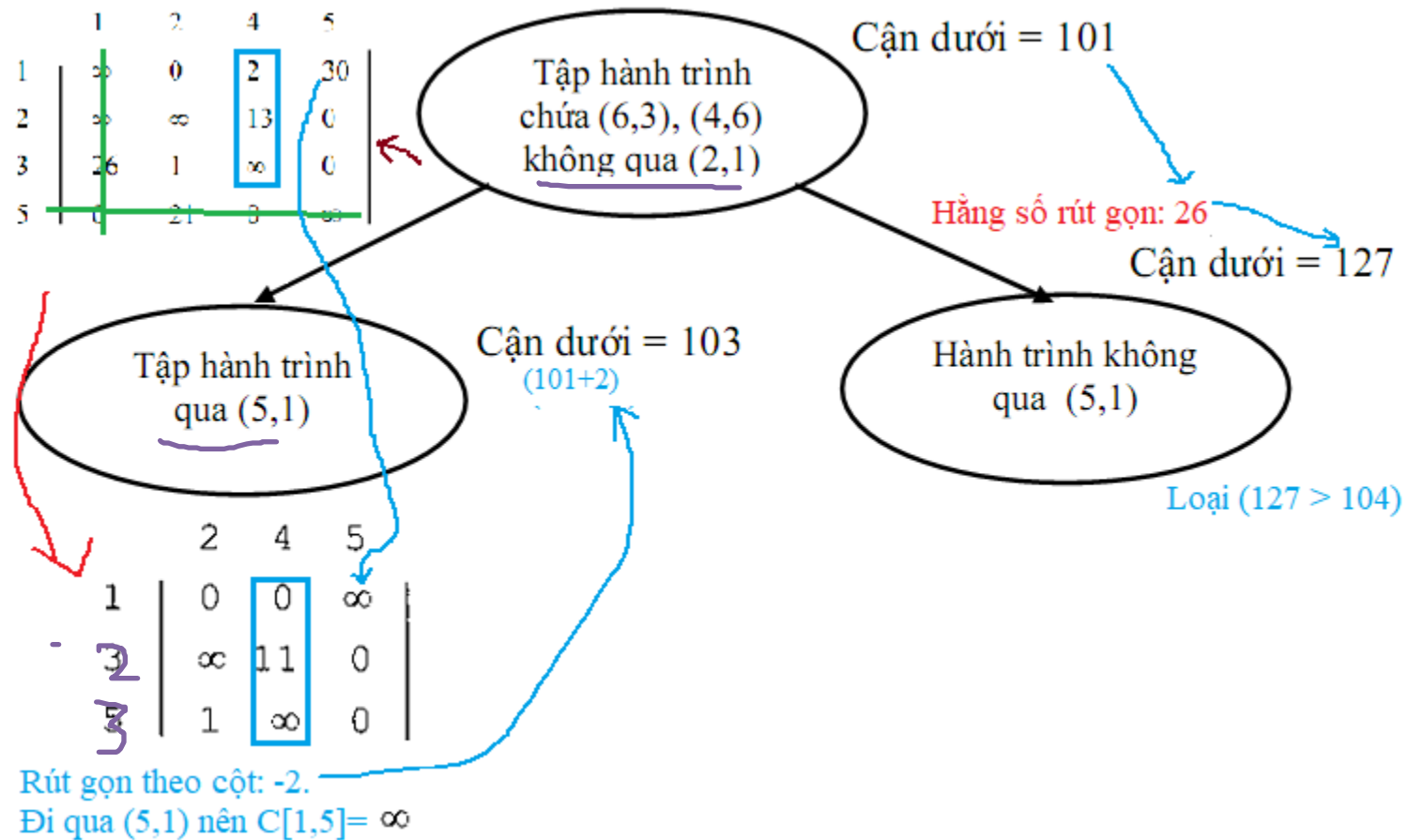
< Kết nạp cạnh $(u, w), (v, x) >;$

Xét các phương án tối ưu khác (nhánh bên phải)

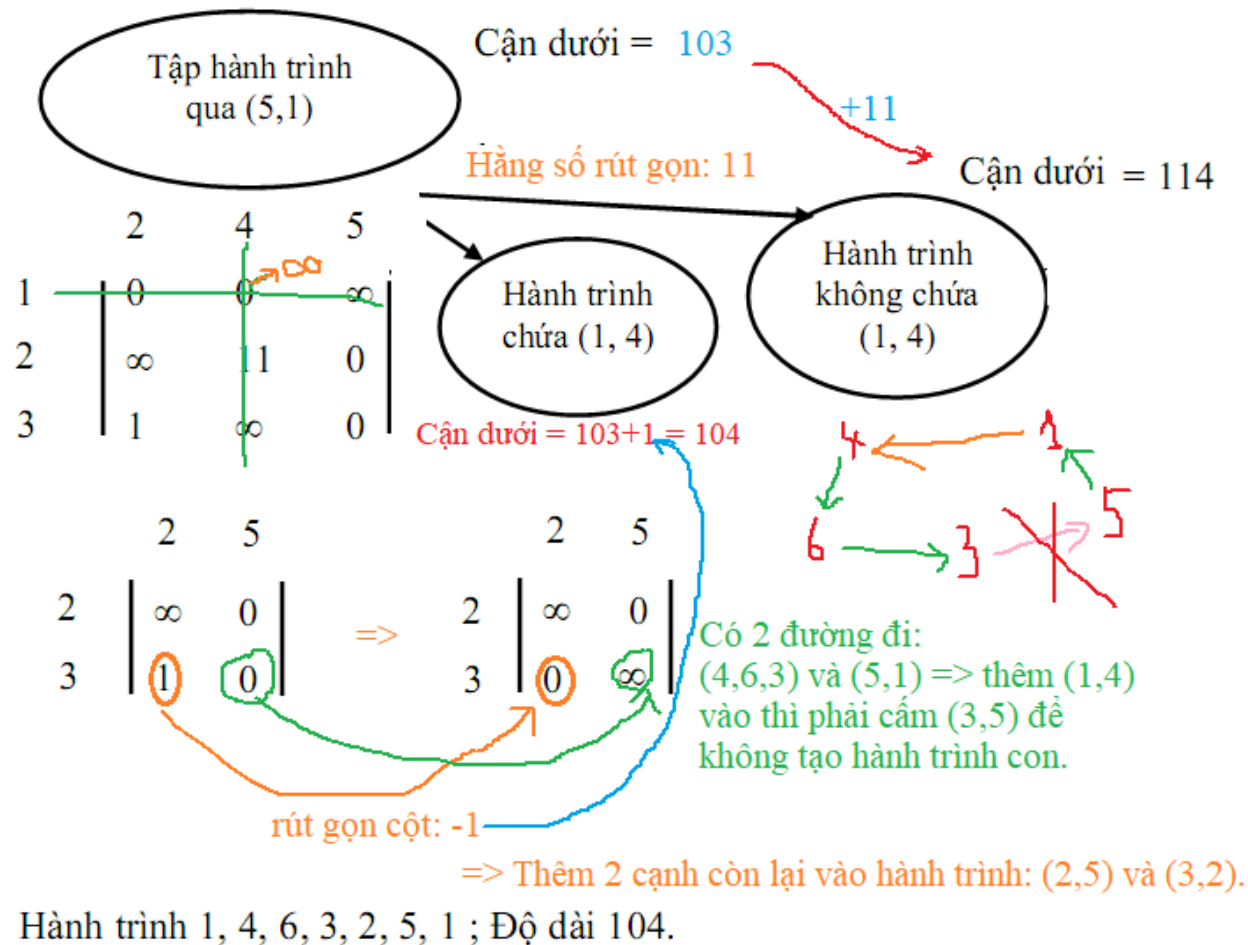


Hành trình (1, 4, 6, 3, 5, 2, 1). Độ dài hành trình: 104.

Phân nhánh tiếp theo – cạnh (5,1)



Phân nhánh tiếp theo – cạnh (1,4)



Thuật toán nhánh cận giải bài toán người du lịch

```
void TSP( Edges, cost, A) {  
    cost=cost + Reduce(A, n-Edges);  
    if (cost < MinCost){  
        if (edges == n-2){  
            <bổ xung nốt hai cạnh còn lại>;  
            MinCost :=Cost;  
        }  
        else {  
            BestEdge(A, n-edges, r, c, beta);  
            LowerBound = Cost + beta;  
            <Ngăn cấm tạo thành hành trình con>;  
            NewA = < A loại bỏ dòng r cột c>;  
            TSP(edges+1, cost, NewA);/*đi theo nhánh trái*/  
            <Khôi phục A bằng cách bổ xung dòng r cột c>;  
            if (LowerBound < MinCost){  
                /* đi theo nhánh phải*/  
                A[r, c] =∞;  
                TSP (edges, cost, A);  
                A[r,c] :=0;  
            }  
        }  
        < Khôi phục ma trận A>;/* thêm lại các hằng số rút gọn vào  
                                các dòng và cột tương ứng*/  
    }  
}/* end of TSP*/;
```



Bài tập 1

- Giải bài toán cái túi sau:

$$\begin{cases} 5x_1 + x_2 + 9x_3 + 3x_4 \rightarrow \max, \\ 4x_1 + 2x_2 + 7x_3 + 3x_4 \leq 10, \\ x_j \in \{0,1\}, j = 1,2,3,4. \end{cases}$$

$$\begin{cases} 9x_3 + 5x_1 + 3x_4 + x_2 \rightarrow \max \\ 7x_3 + 4x_1 + 3x_4 + 2x_2 \leq 10. \end{cases}$$



Bài tập 2

- Giải bài toán người du lịch với ma trận chi phí như sau:

∞	31	15	23	10	17
16	∞	24	07	12	12
34	03	∞	25	54	25
15	20	33	∞	50	40
16	10	32	03	∞	23
18	20	13	28	21	∞