

BRANCH AND BOUND

GIẢI THUẬT NHÁNH CẬN



OVERVIEW

TỔNG QUAN

TSP

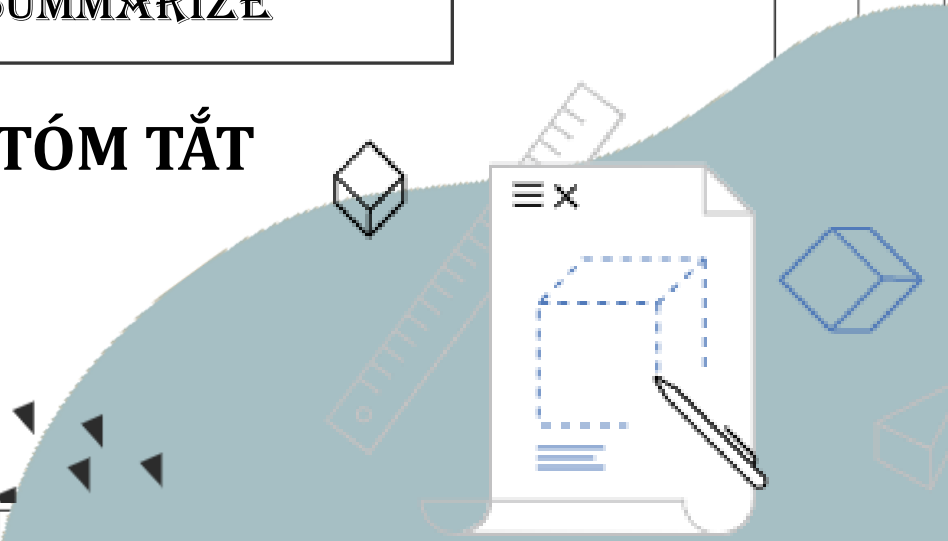
BÀI TẬP

MATRIX

BÀI TẬP

SUMMARIZE

TÓM TẮT





OVERVIEW

TỔNG QUAN





GIỚI THIỆU

Có nhiều bài toán không thể giải được trong thời gian đa thức mà bắt buộc phải sử dụng brute - force hay backtracking.

→ Duyệt qua nhiều cấu hình không cần thiết, lãng phí thời gian

→ **Cải tiến: Branch and Bound (Nhánh - Cận)**



GIỚI THIỆU

Phương pháp nhánh - cận là gì ?

- Nhánh: xây dựng cây phân nhánh, mỗi nhánh đại diện cho một cấu hình có khả năng xảy ra
- Cận: điều kiện để loại các cấu hình xấu

→ Thay vì phải duyệt qua tất cả cấu hình, trên từng bước xây dựng cấu hình, sử dụng cận để xem xét liệu cấu hình hiện tại có khả năng để trở thành cấu hình tốt nhất → nếu không, loại bỏ.

PHƯƠNG PHÁP



Phương pháp giải chung cho
các bài toán nhánh cận ?

01

Xây dựng cây phân nhánh

02

Tìm cận (cận trên/cận dưới)
cho từng node trên cây

03

Xây dựng giải thuật hoàn
chỉnh

PHƯƠNG PHÁP



Phương pháp giải chung cho
các bài toán nhánh cận ?

<khởi tạo một cấu hình bất kỳ cho **BESTCONFIG**>

def function **Attemp(i):**

for <mọi giá trị V có thể gán cho **x[i]**> {

 <thử **x[i] := V**>;

if <việc thử trên vẫn còn hi vọng tìm ra cấu hình tốt hơn
BESTCONFIG> {

if <**x[i]** là phần tử cuối cùng trong cấu hình> {

 <cập nhật **BESTCONFIG**>

 } *else* {

 <ghi nhận việc thử **x[i] := V** (nếu cần)>;

Attemp(i + 1);

 <bỏ ghi nhận việc thử **x[i] := V** (nếu cần)>;

 }

 }

}



TSP

BÀI TẬP



BÀI TẬP – TSP

ĐỀ BÀI

Cho n thành phố đánh số từ 1 đến N và các tuyến đường giao thông hai chiều giữa chúng, mạng lưới giao thông này được cho bởi mảng $C[1...N, 1...N]$ ở đây $C[i][j] = C[j][i]$ là chi phí đi đoạn đường trực tiếp từ thành phố i đến thành phố j .

Một người du lịch xuất phát từ thành phố 1, muốn đi thăm tất cả các thành phố còn lại mỗi thành phố đúng một lần và cuối cùng quay lại thành phố 1. Hãy chỉ ra chi phí ít nhất mà người đó phải bỏ ra.

INPUT

Dòng đầu tiên là số nguyên N – số thành phố ($n \leq 15$)

N dòng sau, mỗi dòng chứa N số nguyên thể hiện cho mảng 2 chiều C ($C[i][i] = 0$).

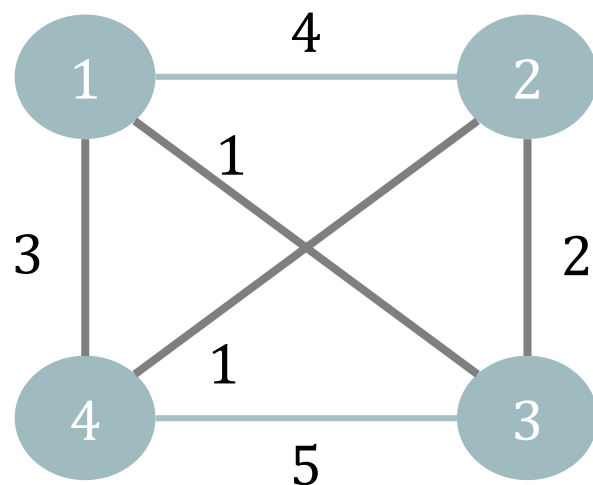
OUTPUT

Chi phí ít nhất mà người đó phải bỏ ra.

Nguồn: <https://www.spoj.com/PTIT/problems/BCTSP/>

BÀI TẬP – TSP

Ví dụ:



	1	2	3	4
1	0	4	1	3
2	4	0	2	1
3	1	2	0	5
4	3	1	5	0

Đường đi tốt nhất: $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1$

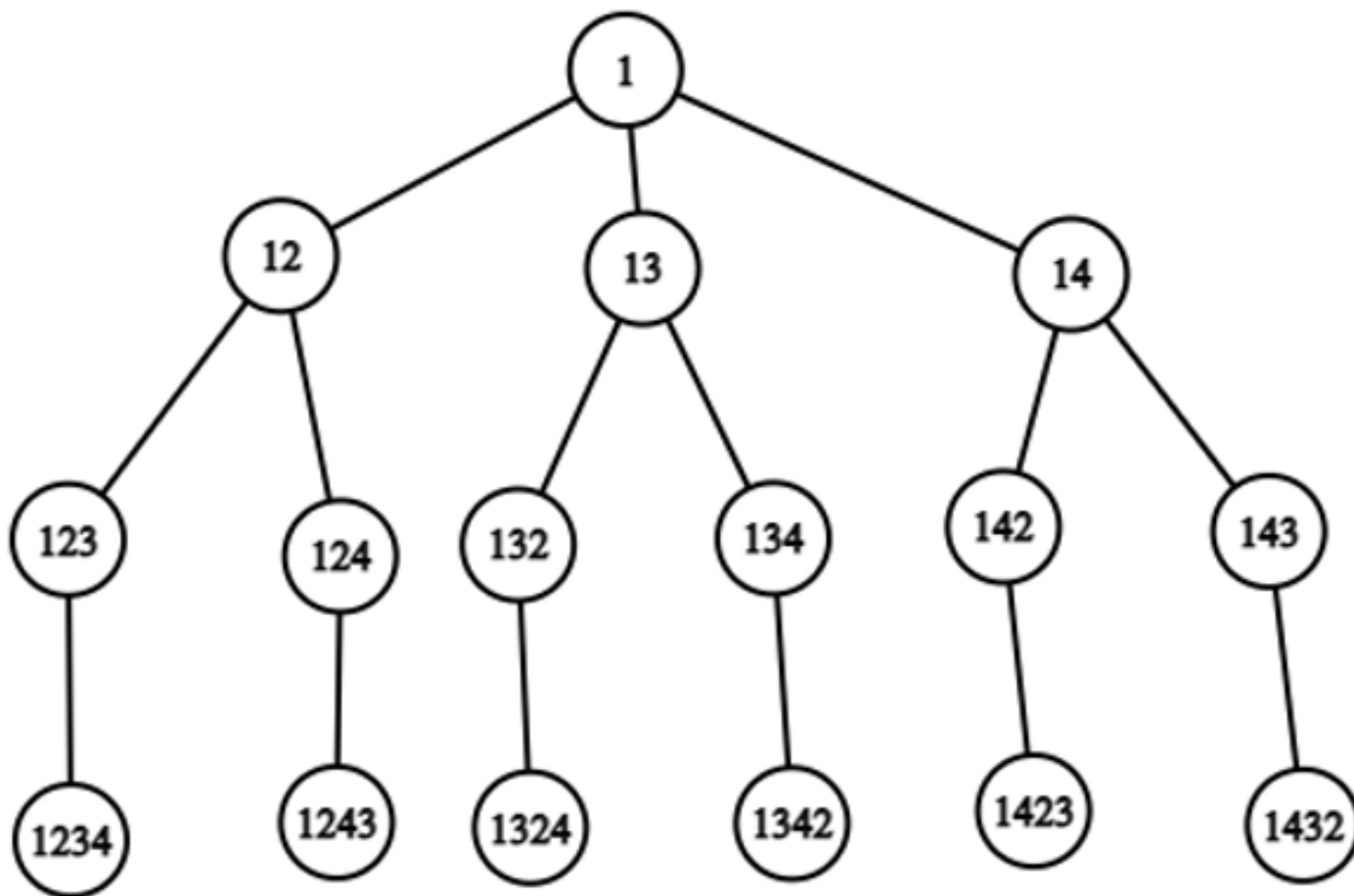
TSP – PHÂN NHÁNH

Xây dựng cây phân nhánh:

- Node gốc (bậc 0): thành phố xuất phát – thành phố 1
- Các node con của node gốc (bậc 1), tương ứng với khả năng đi ra từ thành phố bậc 0
- Các node con bậc 2, tương ứng với khả năng đi ra từ các thành phố bậc 1 ...
- Đến bậc $n - 1$: tạo thêm cạnh cuối cùng từ thành phố bậc $n - 1$ đến thành phố xuất phát (bậc 0), thu được một phương án.

TSP – PHÂN NHÁNH

Ví dụ.



TSP – TẠO CẬN

Xây dựng các kí hiệu:

- X : cấu hình đang được xây dựng hiện tại
- $F(X)$: tổng độ dài đường đi trên cấu hình X
- $\text{LOWER_BOUND}(X)$: giá trị cận dưới cho cấu hình X
- BESTCONFIG : giá trị tốt nhất đến hiện tại

Cách tính giá trị cho các biến ?

- BESTCONFIG : khởi tạo bằng ∞
- $\text{LOWER_BOUND} = F(X) + (n - \text{deg}) \times C_{\min}$
(Với C_{\min} là trọng số cạnh nhỏ nhất chưa có trong đường đi của cấu hình hiện tại)

Đặc điểm bài toán ?

Bài toán yêu cầu tìm **giá trị nhỏ nhất**

→ Tạo cận dưới cho mỗi cấu hình

TSP – TẠO CẬN

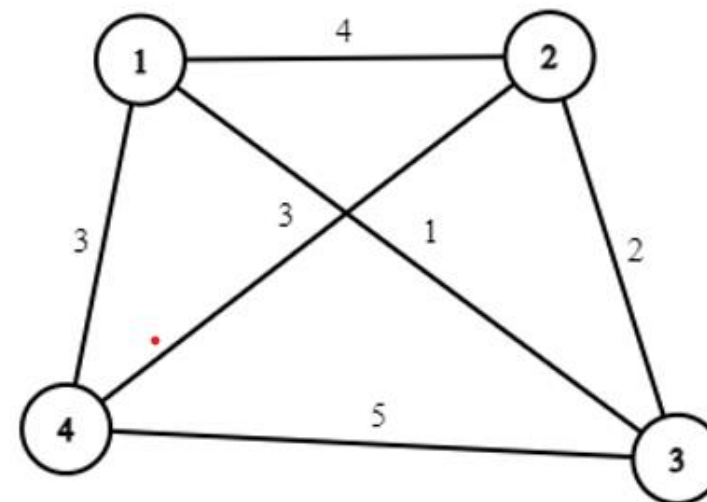
Ví dụ.

Cấu hình hiện tại: $X = 1\ 4$

$$\begin{aligned}\rightarrow \text{LOWER_BOUND} &= F(X) + (n - \text{deg}) \times C_{\min} \\ &= C(1, 4) + (n - 1) \times 1 \\ &= 6\end{aligned}$$

Cấu hình hiện tại: $X = 1\ 3\ 2$

$$\begin{aligned}\rightarrow \text{LOWER_BOUND} &= F(X) + (n - \text{deg}) \times C_{\min} \\ &= C(1, 3) + C(3, 2) + (n - 2) \times 3 \\ &= 11\end{aligned}$$



TSP – GIẢI THUẬT



Dựa theo phương pháp chung,
xây dựng giải thuật cho TSP ?

#G = (V, E)

BESTCONFIG := ∞ ;

X[0] := 1;

def **Visit**(deg, F(X), U):

for <V : $\forall (U, V) \in E, V \notin X$ > {

X[deg] := V;

if **LOWER_BOUND**(X) < **BESTCONFIG** {

if (deg = n - 1)

minimize(**BESTCONFIG**, F(X) + C(U, V) + C(V, X[0]));

else

Visit(deg+1, F(X) + C(U, V), V);

}

}

TSP – ĐỘ PHỨC TẠP

Trường hợp xấu nhất: số cấu hình phải duyệt lên đến $(N-1)! \rightarrow O(N!)$

Thực tế: số cấu hình phải duyệt thường nhỏ hơn rất nhiều

Một vài phương pháp khác:

- Backtracking: $O(N!)$
- Quy hoạch động trạng thái: $O(N \times 2^N)$

MATRIX

BÀI TẬP

BÀI TẬP – MATRIX

ĐỀ BÀI

Cho một bảng số 5×5 . Nhiệm vụ của bạn là sẽ phải điền vào ma trận sao cho tổng của các phần tử trên mỗi hàng và mỗi cột bằng một số nguyên cho trước. Mỗi phần tử trong bảng số từ 1 đến 25 và không có hai phần tử bất kì nào giống nhau.

INPUT

Dòng thứ nhất gồm 5 số là tổng của các số từ dòng thứ 1 đến dòng thứ 5 của bảng số (Mảng D).

Dòng thứ hai gồm 5 số là tổng của các số từ cột thứ 1 đến cột thứ 5 của bảng số (Mảng C).

OUTPUT

Gồm 5 dòng, mỗi dòng 5 số thể hiện bảng 5×5 là kết quả của bạn. Nếu có nhiều đáp án, hãy in ra một đáp án bất kì. Dữ liệu đầu vào luôn luôn có kết quả.

Nguồn: <https://vn.spoj.com/problems/VMMTFIVE/>

BÀI TẬP – MATRIX

Ví dụ.

INPUT

60 86 59 38 82
61 59 57 89 59

OUTPUT

15 5 9 25 6
17 10 23 20 16
12 19 3 18 7
13 14 1 2 8
4 11 21 24 22

15	5	9	25	6	60
17	10	23	20	16	86
12	19	3	18	7	69
13	14	1	2	8	38
4	11	21	24	22	82

61 59 57 89 59

MATRIX – PHÂN NHÁNH

Duyệt lần lượt theo từng vị trí trên bảng từ ô $[1, 1]$ đến $[5, 5]$:

- $1,1 \rightarrow 1,2 \rightarrow 1,3 \rightarrow 1,4 \rightarrow 1,5 \rightarrow 2,1 \rightarrow 2,2 \rightarrow \dots \rightarrow 5,5$

Cây phân nhánh:

- Node gốc: trạng thái rỗng
- Node bậc 1: $X[1, 1] \in [1, 25]$
- Node bậc 2: $X[1, 2] \in [1, 25]$ (chưa xuất hiện trong hàng 1 và cột 2 của X)
- Node bậc 3: $X[1, 3] \in [1, 25]$ (chưa xuất hiện trong hàng 1 và cột 3 của X) ...
- Node bậc 25: $X[5, 5] \rightarrow$ hoàn chỉnh cấu hình

MATRIX – TẠO CẬN

Kí hiệu:

- $d[i]$: tổng các số đã điền trên hàng i
- $c[j]$: tổng các số đã điền trên cột j

Tạo các cận: xét đến hàng i , cột j :

- $K_{\min} \times (5 - j) < D[i] - d[i] < K_{\max} \times (5 - j)$
- $K_{\min} \times (5 - i) < C[j] - c[j] < K_{\max} \times (5 - i)$

(K_{\min} / K_{\max} là số nhỏ nhất/lớn nhất chưa được chọn vào cấu hình)

MATRIX – TẠO CẬN

Ví dụ: $D = [60 \ 86 \ 59 \ 38 \ 82]$
 $C = [61 \ 59 \ 57 \ 89 \ 59]$

Cấu hình hiện tại:

15	5	9	25	6
17	10	20	23	16
14	2	x	x	x
x	x	x	x	x
x	x	x	x	x

Tính được: $d[3] = 16$, $c[2] = 17$

Cận 1: $K_{\min} \times (5 - j) < D[i] - d[i] < K_{\max} \times (5 - j)$
 $\rightarrow 1 \times 3 < 43 < 25 \times 3$ (thỏa)

Cận 2: $K_{\min} \times (5 - i) < C[j] - c[j] < K_{\max} \times (5 - i)$
 $\rightarrow 1 \times 2 < 42 < 25 \times 2$ (thỏa)

Cấu hình hiện tại:

15	5	9	6	25
17	24	23	2	20
14	19	11	1	x
x	x	x	x	x
x	x	x	x	x

Tính được: $d[3] = 45$, $c[4] = 9$

Cận 1: $K_{\min} \times (5 - j) < D[i] - d[i] < K_{\max} \times (5 - j)$
 $\rightarrow 3 \times 1 < 14 < 22 \times 1$ (thỏa)

Cận 2: $K_{\min} \times (5 - i) < C[j] - c[j] < K_{\max} \times (5 - i)$
 $\rightarrow 3 \times 2 < 80 < 22 \times 2$ (không thỏa)

→ Loại bỏ cấu hình hiện tại

MATRIX – GIẢI THUẬT



Dựa theo phương pháp chung, xây dựng giải thuật cho bài toán ?

$X = \langle \text{matrix } 5 \times 5 \rangle;$

def **Visit**(U, V):

for $\langle K : K \in [1, 25], K \notin X \rangle \{$

$X[U, V] := K;$

if **<cấu hình hiện tại thỏa mãn các điều của cận>**{

if $(U == 5 \text{ and } V == 5)$

<trả về cấu hình kết quả>;

else {

$d[i] += K; \quad c[j] += K;$

Visit(next(U, V));

$d[i] -= K; \quad c[j] -= K;$

}

}

}



SUMMARIZE

TÓM TẮT



NHẬN DẠNG

Phương pháp nhánh – cận có thể áp dụng để giải quyết hầu hết những bài toán yêu cầu tìm giá trị lớn nhất/nhỏ nhất có (tối ưu tổ hợp) thể cho một vấn đề/sự việc nào đó:

- Tìm đường đi cho tour du lịch quanh các thành phố với chi phí thấp nhất
- Tìm cách mua sản phẩm có tổng giá trị lớn nhất với số tiền cho trước
- Chọn địa điểm xây các trạm phát sóng sao cho toàn bộ vùng đất được phủ sóng và tốn ít chi phí nhất trong thời gian cho phép...

Nhiều bài toán sử dụng phương pháp backtracking khác cũng có thể đặt cận để loại bỏ qua các cấu hình xấu.

ƯU ĐIỂM

NHƯỢC ĐIỂM

ƯU ĐIỂM

- Chúng ta không cần thiết phải đi hết qua tất cả các node, từ đó giảm được chi phí tính toán
- Đối với bài toán có kích thước không lớn, chúng ta có thể rẽ nhánh với một lượng thời gian hợp lí để tìm được lời giải tối ưu
- Nó không cần lặp lại các node trong cây

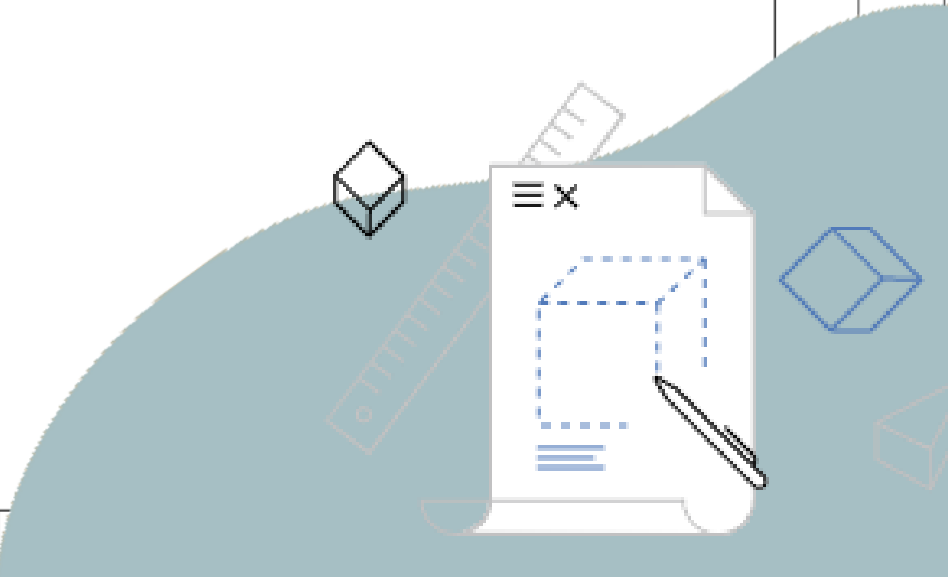
NHƯỢC ĐIỂM

- Giải thuật nhánh cận sẽ không phù hợp với bài toán có kích thước quá lớn.
- Trong trường hợp xấu nhất, nó sẽ duyệt qua hết tất cả các node trong cây



ỨNG DỤNG

Giải thuật nhánh cận được vận dụng trong các lĩnh vực tổ hợp – rời rạc, khoa học máy tính, vận trù học, phân tích độ phức tạp

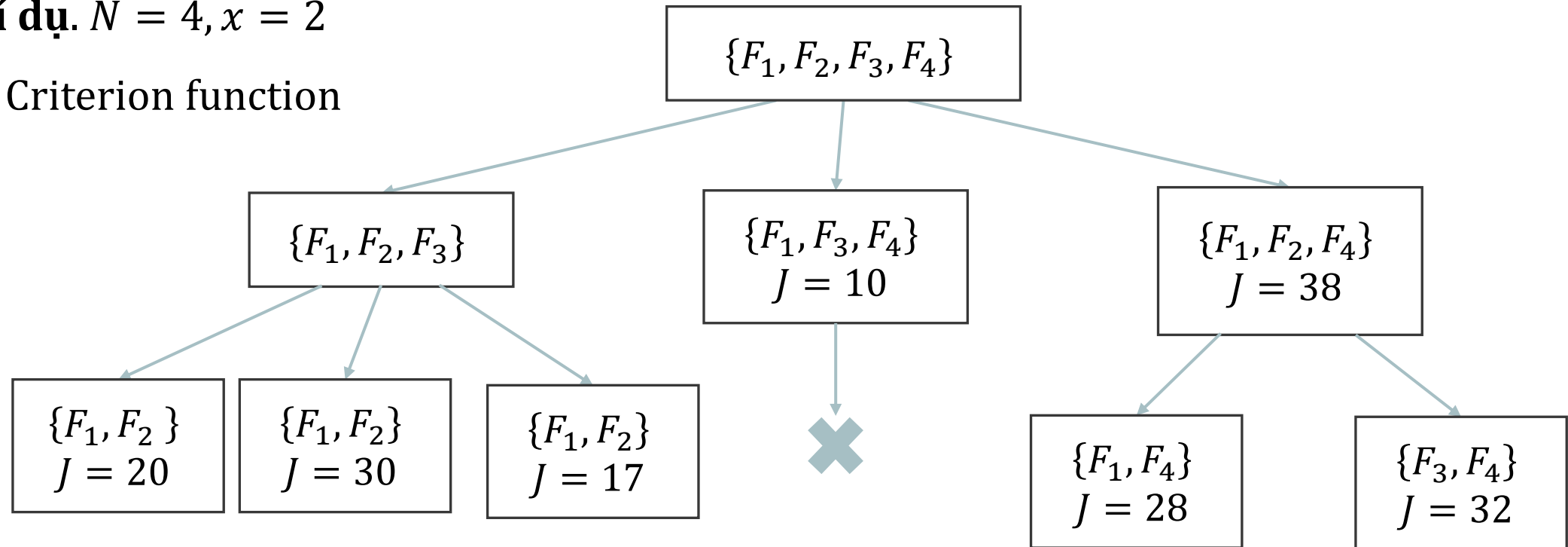
- Trích chọn đặc trưng (Feature selection) trong Machine Learning
 - Dự đoán cấu trúc (Structured prediction) trong Computer Vision
 - Quy hoạch tuyến tính, quy hoạch phi tuyến tính
 - Nghịch đảo tập hợp (Set inversion)
 - Vấn đề cắt giảm cổ phiếu
 - Ước lượng tham số
 - Tính toán phát sinh chủng loài
 - ...
- 

ỨNG DỤNG

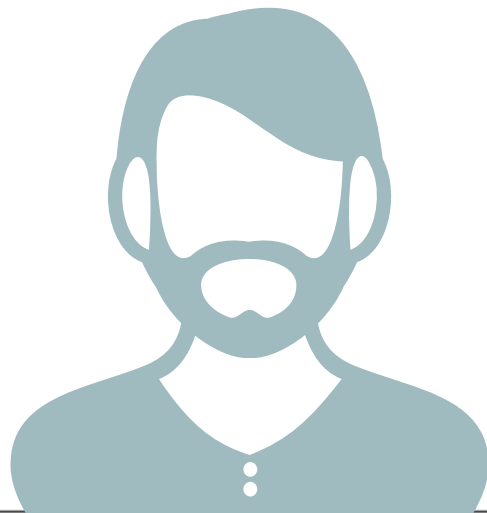
GIẢI THUẬT NHÁNH CẬN TRONG FEATURE SELECTION

Ví dụ. $N = 4, x = 2$

J : Criterion function



Q&A



THANKS FOR
LISTENING