TOÁN RỜI RẠC 1

CHƯƠNG 4

Giảng viên: Vũ Văn Thỏa

CHƯƠNG 4. BÀI TOÁN TỐI ƯU

- Giới thiệu bài toán
- Phương pháp duyệt toàn thể
- Phương pháp nhánh cận
- Phương pháp quy hoạch động

4.1 Giới thiệu bài toán tối ưu

- **Bài toán tối ưu**: Cho tập dữ liệu D = {x}, trong đó x = (x_1, x_2, …, x_n), với $x_i \in D_i$.

Yêu cầu: Tìm $x \in D$ sao cho $f(x) \rightarrow$ Min(Max);

- Hàm f(x) được gọi là hàm mục tiêu của bài toán.

- Mỗi phần tử $x \in D$ gọi là một phương án của bài toán.

- Phương án $X_{opt} \in D$ để hàm mục tiêu có giá trị nhỏ nhất (lớn nhất) gọi là phương án tối ưu của bài toán,

- $F_{opt} = f(x_{opt})$ được gọi là giá trị tối ưu của bài toán.

Ví dụ 1. *Bài toán cái túi*

Một nhà thám hiểm có một cái túi có thể chứa các đồ vật với trọng lượng không quá b.

Có n đồ vật cần đem theo.

Đồ vật thứ i có trọng lượng a_i, có giá trị sử dụng c_i (i= 1, 2, .., n; a_i, $c_i \in Z+$).

Hãy tìm cách chọn đồ vật có thể đem theo cho nhà thám hiểm sao cho tổng giá trị sử dụng các đồ vật trong túi là lớn nhất.

Tập phương án của bài toán cái túi

Mỗi cách chọn đồ vật $\Leftrightarrow$ xâu nhị phân có độ dài n:

- $x_i = 1 \Leftrightarrow$ đồ vật i được chọn,
- $x_i = 0 \Leftrightarrow$ đồ vật i không được chọn.

$\Rightarrow$ Tập phương án *D* của bài toán:

$$D = \{X \mid g(X) \leq b\},$$

Trong đó,

- $X = (x_1, \ldots, x_n)$, $x_i \in \{0, 1\}$;
- $g(X) = a_1x_1 + \ldots + a_nx_n$

Hàm mục tiêu của bài toán cái túi

- Cần tìm phương án Xopt =(x^*_1,..,x^*_n) $\in$ D sao cho tổng giá trị sử dụng các đồ vật trong túi là lớn nhất:

$$\mathbf{f(X) = c_1x_1 + \ldots + c_nx_n \rightarrow max}$$

$$X \in D = \{X \mid g(X) \leq b\},$$

Trong đó,

- $X = (x_1, \ldots, x_n)$, $x_i \in \{0, 1\}$;
- $g(X) = a_1x_1 + \ldots + a_nx_n$

Mô hình toán học của bài toán cái túi

Mô hình:

$$f(X) = c_1x_1 + \ldots + c_nx_n \rightarrow \max,$$
$$X \in D,\ D = \{X \mid g(X) \leq b\}$$

Trong đó,

- $X = (x_1, \ldots, x_n);\ x_i \in \{0, 1\}$
- $g(X) = a_1x_1 + \ldots + a_nx_n$;

Ví dụ bài toán cái túi:

$$\begin{cases} 5x_1 + 3x_2 + 10x_3 + 6x_4 \rightarrow \max \\ 3x_1 + 2x_2 + 5x_3 + 4x_4 \leq 8 \\ x_i \in \{0, 1\},\ i = 1, 2, 3, 4 \end{cases}$$

Đặt bài toán cái túi cho lập trình:

- **Input:**
  - Số lượng đồ vật: n;
  - Tổng khối lượng: b;
  - Vector trọng lượng: A = (a_1, .., a_n);
  - Vector giá trị sử dụng: C = (c_1, .., c_n);
- **Output:**
  - Phương án tối ưu: Xopt = (x_1, .., x_n) $\in$ D = {Xâu nhị phân độ dài n} để f(Xopt) lớn nhất;
  - Giá trị tối ưu: Fopt = f(Xopt).

Ví dụ 2: Bài toán người du lịch

Một người du lịch muốn đi tham quan n thành phố đánh số từ 1 đến n.

Xuất phát tại một thành phố x_1 nào đó, người du lịch muốn qua tất cả các thành phố còn lại mỗi thành phố đúng một lần rồi trở lại thành phố x_1.

Biết c_{ij}, i, j = 1..n, là chi phí đi lại từ thành phố i đến thành phố j.

Hãy tìm một hành trình cho người đi du lịch có tổng chi phí là nhỏ nhất.

Tập phương án của bài toán NDL:

Mỗi hành trình của người du lịch $x_1 \rightarrow x_2 \rightarrow .. \rightarrow x_n \rightarrow x_1$ $\Leftrightarrow$ có thể chọn $x_1 = 1$.

$\Leftrightarrow$ Mỗi hành trình là một hoán vị của các số 1, 2, …, n:

$$X = (x_1, x_2, .., x_n), \text{ trong đó } x_1 = 1,$$

$\Rightarrow$Tập phương án D của bài toán là tập các hoán vị

$$D = \{X = (x_1, \ldots, x_n) | x_1 = 1, x_i \neq x_j; i,j = 1.. n\}$$

Hàm mục tiêu của bài toán NDL

- Tương ứng với mỗi phương án $X = (x_1,..,x_n) \in D$, chi phí đi lại từ thành phố thứ x_i đến thành phố x_{i+1} là $C[x_i][x_{i+1}]$ (i = 1, 2,.., n-1).
- Sau đó quay lại thành phố ban đầu với chi phí là $C[x_n][x_1]$.

- Hàm mục tiêu là tổng chi phí của toàn bộ hành trình:

$$\mathbf{f(X) =} C[x_1][x_2] + \ldots + C[x_{n-1}][x_n] + C[x_n][x_1] \rightarrow \mathbf{min}$$

Mô hình của bài toán người du lịch

Mô hình toán học:

f(X) = $C[x_1][x_2] + \ldots + C[x_{n-1}][x_n] + C[x_n][x_1] \rightarrow$ **min, X** $\in$ **D**

$D = \{X = (x_1, \ldots, x_n) | x_1 = 1, x_i \neq x_j; i,j = 1..n\}$; $C = (c_{ij})$

Ví dụ bài toán người du lịch với ma trận chi phí C:

0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0

Đặt bài toán NDL cho lập trình

- **Input**:
  - Số lượng thành phố: n
  - Ma trận chi phí cấp n: C = (c_{ij})
- **Output**:
  - Phương án tối ưu: Xopt =$(x_1, x_2, .., x_n) \in D$ để f(Xopt) nhỏ nhất
  - Giá trị tối ưu của bài toán: Fopt = f(Xopt).

Trong đó:

- **f(X) =** $C[x_1][x_2] + \ldots + C[x_{n-1}][x_n] + C[x_n][x_1] \rightarrow$ **min, X** $\in$ **D**
- D = $\{X = (x_1, \ldots, x_n) | x_1 = 1, x_i \neq x_j; i,j = 1..n\}$; C = (c_{ij})

Ví dụ 3: Bài toán thuê máy

Một ông chủ có một chiếc máy cho thuê. Đầu tháng ông nhận được yêu cầu của m khách hàng thuê máy cho n ngày kế tiếp.

Các khách hàng được đánh số từ 1 đến m.

Mỗi khách hàng i ($1 \leq i \leq m$), biết tập N_i ngày cần thuê máy.

Ông chủ có quyền hoặc từ chối yêu cầu của khách hàng, hoặc nếu chấp nhận yêu cầu của khách thì phải bố trí máy theo đúng như khách yêu cầu.

Hãy tìm phương án thuê máy giúp ông chủ sao cho tổng số ngày thuê máy là nhiều nhất.

Mô hình hóa bài toán thuê máy:

- **Tập phương án**:
  - I = { 1, 2, .., m} là tập chỉ số khách hàng.
  - S là tập của tất các tập con của I.
  - Tập các phương án cho thuê máy:

$D = \{J \subset S |\ N_k \cap N_p = \varnothing, \forall k,p \in J\}$

- **Hàm mục tiêu**:
  - Với mỗi $J \in D$, tổng số ngày cho thuê máy là

$$f(J) = \sum |N_j|,\ j \in J$$

  - Cần tìm $J \in D$ sao cho

$$f(J) = \sum |N_j|,\ j \in J \rightarrow \max$$

Đặt bài toán thuê máy cho LT:

- **Input**:
  - Số lượng khách hàng: m;
  - Số ngày cần cho thuê máy: n,
  - Ma trận A[i][j] cấp mxn mô tả số ngày thuê máy mỗi khách i:

    A[i][j] = 1 nếu khách hàng i thuê máy ngày thứ j,

    A[i][j] = 0 nếu khách hàng i không thuê máy ngày thứ j;
- **Output**:
  - Phương án tối ưu của bài toán: Jopt $\subset$ S.
  - Giá trị tối ưu: Fopt = f(Jopt) là số ngày cho thuê máy lớn nhất.

Ví dụ 4: Bài toán phân việc

Có n công nhân và n việc.

Biết mỗi công nhân i ($1 \leq i \leq n$) có thể thực hiện công việc j với chi phí thời gian c_{ij}.

Hãy tìm phương án giao việc cho mỗi công nhân 1 việc sao cho tổng thời gian thực hiện n việc kể trên là ít nhất.

Mô hình toán học của bài toán phân việc

- **Tập phương án**
  - Gọi $X = (x_1,..,x_n)$ là một hoán vị của $\{1,..,n\}$.
  - Nếu $x_i = j$ thì công nhân thứ i sẽ thực hiện việc j.
  - Tập phương án là tập các hoán vị của $\{1, .., n\}$:

$$D = \{(x_1,..,x_n),\ x_i \neq x_j,\ i,j=1..n\}$$

- **Hàm mục tiêu**
  - với mỗi phương án $X \in D$, thời gian thực hiện của mỗi phương án là:

$$f(X) = \sum c[i][x_i],\ i = 1..n$$

  - Cần tìm $X \in D$ thỏa mãn

$$\mathbf{f(X) = \sum c[i][x_i],\ X \in D \rightarrow Min}$$

Đặt bài toán phân việc cho LT

- **Input**:
  - Số lượng công việc (công nhân): n;
  - Ma trận vuông cấp n chi phí thời gian: C[i][j];
- **Output**:
  - Phương án tối ưu: Xopt$\in$ D;
  - Giá trị tối ưu: Fopt = f(Xopt).

- Trong đó:
  - **D = {(x_1,..,x_n), $x_i \neq x_j$, i,j=1..n}**
  - **f(X) = $\sum$c[i][x_i], $X \in D \rightarrow$ Min**

Phương pháp giải:

- Phương pháp duyệt toàn thể
- Phương pháp nhánh cận
- Phương pháp quy hoạch động

4.2 Phương pháp duyệt toàn thể

- **Ý tưởng chung**:
  - Giả sử D là tập phương án, f() là hàm mục tiêu của bài toán tối ưu.
  - Với mỗi $X \in D$, tính f(X) và so sánh để tìm phương án sao cho hàm mục tiêu $f(X) \rightarrow$ min (max).

Thuật toán duyệt toàn thể tìm min:

Bước 1 (*Khởi tạo*): Xopt = ∅; //*Phương án tối ưu*

Fopt= + ∞;//*Giá trị tối ưu*

Bước 2 (*Lặp*):

for each X∈D do { //*lấy mỗi phần tử trên tập phương án*

t = f(X);// *tính giá trị hàm mục tiêu của phương án X*

if (Fopt > t) { //*Cập nhật phương án tối ưu*

Fopt = t; //*Giá trị tối ưu mới được xác lập*

Xopt = X;// *Phương án tối ưu mới* }

}

Bước 3 (*Trả lại kết quả*): Return(Xopt, Fopt);

Thuật toán duyệt toàn thể tìm max:

Bước 1 (*Khởi tạo*): Xopt = ∅; //*Phương án tối ưu*

Fopt= - ∞;//*Giá trị tối ưu*

Bước 2 (*Lặp*):

for each X∈D do { //*lấy mỗi phần tử trên tập phương án*

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

if (Fopt < t) { //*Cập nhật phương án tối ưu*

Fopt = t; //*Giá trị tối ưu mới được xác lập*

Xopt = X;// *Phương án tối ưu mới* }

}

Bước 3 (*Trả lại kết quả*): Return(Xopt, Fopt);

Áp dụng cho bài toán cái túi:

Giải bài toán cái túi bằng phương pháp duyệt toàn thể:

$$f(X) = c_1x_1 + \ldots + c_nx_n \rightarrow \max,$$
$$X \in D,\ D = \{X \mid g(X) \leq b\}$$

Trong đó,

- $X = (x_1, \ldots, x_n);\ x_i \in \{0, 1\}$
- $g(X) = a_1x_1 + \ldots + a_nx_n$;

Ví dụ bài toán cái túi:

$$\begin{cases} 5x_1 + 3x_2 + 10x_3 + 6x_4 \rightarrow \max \\ 3x_1 + 2x_2 + 5x_3 + 4x_4 \leq 8 \\ x_i \in \{0, 1\},\ i = 1, 2, 3, 4 \end{cases}$$

Thuật toán duyệt toàn thể cho bài toán cái túi

Bước 1 (Khởi tạo): Fopt = - ∞; Xopt = ∅;

Bước 2 (Lặp): Duyệt tất cả các phương án X = (x_1, …, x_n) là các xâu nhị phân độ dài n theo thứ tự từ điển:

- Tính g(X)= a_1x_1 + … + a_nx_n
- Nếu (g(X) <= b) {

  Tính f(X) = c_1x_1 + … + c_nx_n

  Nếu (f(X) > Fopt) {Fopt= f(X); Xopt= X; }

  }

Bước 3 (Trả lại kết quả): Return(Fopt, Xopt);

Ví dụ 1:

Giải bài toán cái túi bằng phương pháp duyệt toàn thể:

$$\begin{cases} 5x_1 + 3x_2 + 10x_3 + 6x_4 \rightarrow \max \\ 3x_1 + 2x_2 + 5x_3 + 4x_4 \le 8 \\ x_i \in \{0, 1\},\ i = 1, 2, 3, 4 \end{cases}$$

Giải

- **Xác định bài toán**:
  - n = 4
  - b = 8
- **Lập bảng:**

Lập bảng: $g(X) = 3x_1 + 2x_2 + 5x_3 + 4x_4$, $b = 8$; $f(X) = 5x_1 + 3x_2 + 10x_3 + 6x_4$; Fopt = - ∞; Xopt = ∅;

X	g(X)	g(X) ≤ b?	f(X)	Fopt
0,0,0,0	0	Yes	0	0
0,0,0,1	4	Yes	6	6
0,0,1,0	5	Yes	10	10
0,0,1,1	9	No	-	-
0,1,0,0	2	Yes	3	-
0,1,0,1	6	Yes	9	-
0,1,1,0	7	Yes	13	13
0,1,1,1	11	No	-	-

X	g(X)	g(X) ≤ b?	f(X)	Fopt
1,0,0,0	3	Yes	5	-
1,0,0,1	7	Yes	11	-
1,0,1,0	8	Yes	15	15
1,0,1,1	12	No	-	-
1,1,0,0	5	Yes	8	-
1,1,0,1	9	No	-	-
1,1,1,0	10	No	-	-
1,1,1,1	14	No	-	-

Kết luận: Fopt = 15; Xopt = (1,0,1,0)

Áp dụng cho bài toán người du lịch

Giải bài toán du lịch bằng phương pháp duyệt toàn thể:

f(X) = $C[x_1][x_2] + \ldots + C[x_{n-1}][x_n] + C[x_n][x_1] \rightarrow$ **min, X** $\in$ **D**

$D = \{X = (x_1, \ldots, x_n) | x_1 = 1, x_i \neq x_j; i,j = 1..\ n\}$; $C = (c_{ij})$

Ví dụ bài toán người du lịch với ma trận chi phí C:

0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0

Thuật toán duyệt toàn thể giải bài toán người du lịch

- **Bước 1** (*Khởi tạo*): Fopt = + ∞; Xopt = ∅;
- **Bước 2** (*Lặp*):

Duyệt tất cả các phương án X = (x_1, x_2, …, x_n) là các hoán vị độ dài n theo thứ tự từ điển, với x_1 = 1:

- Tính f(X) = c[x_1][x_2] + … + c[x_{n-1}][x_n] + c[x_n][x_1]
- Nếu (f(X) < Fopt) {

  Fopt= f(X); Xopt= X;

  }

- **Bước 3** (Trả lại kết quả): Return(Fopt, Xopt);

Ví dụ 2:

Giải bài toán người du lịch cho bởi ma trận chi phí dưới đây bằng phương pháp duyệt toàn thể

0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0

Giải

- **Xác định bài toán**:
  - n = 5
  - Chi phí c_{ij} cho trong ma trận C
- **Lập bảng**:

  Fopt = + ∞; Xopt = ∅;

Lập bảng: Fopt = +∞; Xopt = ∅

X	f(X)	Fopt
1, 2, 3, 4, 5	44	44
1, 2, 3, 5, 4	22	22
1, 2, 4, 3, 5	45	-
1, 2, 4, 5, 3	65	-
1, 2, 5, 3, 4	56	-
1, 2, 5, 4, 3	52	-
1, 3, 2, 4, 5	66	-
1, 3, 2, 5, 4	54	-
1, 3, 4, 2, 5	61	-
1, 3, 4, 5, 2	60	-
1, 3, 5, 2, 4	61	-
1, 3, 5, 4, 2	28	-

X	f(X)	Fopt
1, 4, 2, 3, 5	36	-
1, 4, 2, 5, 3	68	-
1, 4, 3, 2, 5	53	-
1, 4, 3, 5, 2	47	-
1, 4, 5, 2, 3	66	-
1, 4, 5, 3, 2	53	-
1, 5, 2, 3, 4	66	-
1, 5, 2, 4, 3	76	-
1, 5, 3, 2, 4	63	-
1, 5, 3, 4, 2	47	-
1, 5, 4, 2, 3	43	-
1, 5, 4, 3, 2	39	-

Kết luận:.

Fopt = 22; Xopt = (1, 2, 3, 5, 4);

Nhận xét:

- **Ưu điểm của phương pháp duyệt toàn thể**:
  - Đơn giản dễ cài đặt.
  - Có thể thực hiện trên mọi bài toán tối ưu.
- **Nhược điểm của phương pháp duyệt toàn thể** :

  Chi phí tính toán lớn khi D lớn.

  $\Rightarrow$ trong thực tế sẽ tính gần đúng theo chi phí thời gian chấp nhận được

4.3 Phương pháp nhánh cận

- **Bài toán tìm min**: Cho tập dữ liệu D = {x}, trong đó x = $(x_1, x_2, \ldots, x_n)$, với $x_i \in D_i$.

Yêu cầu: Tìm $x \in D$ sao cho $f(x) \rightarrow$ Min;

- **Bài toán tìm max**: Cho tập dữ liệu D = {x}, trong đó x = $(x_1, x_2, \ldots, x_n)$, với $x_i \in D_i$.

Yêu cầu: Tìm $x \in D$ sao cho $f(x) \rightarrow$ Max;

Ý tưởng nhánh cận tìm min:

- Giả sử đã xây dựng được k thành phần của phương án x = (a_1, …, a_k, x_{k+1}, …, x_n).
- Đặt D(a_1,…, a_k) = {x∈D | x_i = a_i với i = 1..k}:
  - Giả sử tìm được g_k = $g(a_1, a_2,..,a_k)$ thỏa mãn:

    $f(x) \geq g_k$ với mọi $x \in D(a_1,\ldots, a_k)$
  - Khi đó có min {f(x), $\forall x \in D(a_1,\ldots, a_k)$ } ≥ g_k
  - g_k gọi là cận dưới của f(x) trên tập D(a_1,…, a_k).
- Gọi giá trị tối ưu hiện thời của bài toán tìm min là Fopt:

Nếu $g_k \geq$ Fopt thì min {f(x), $\forall x \in D(a_1,\ldots, a_k)$ } ≥ Fopt

⇔ với mọi $x \in D(a_1,\ldots, a_k)$ không phải là phương án tối ưu

- Không cần phải tiếp tục phương án bộ phận (a_1,a_2,..,a_k) ⇔ Tập D(a_1,a_2,..a_k) bị loại bỏ khỏi quá trình duyệt.

Thuật toán nhánh cận tìm min:

Thuật toán Branch_And_Bound_Min (k) {

for $a_k \in A_k$ do { if (<chấp nhận a_k>){

$x_k = a_k$;

if (k==n) {

if (f(X) < Fopt) <Cập nhật kỷ lục>;}

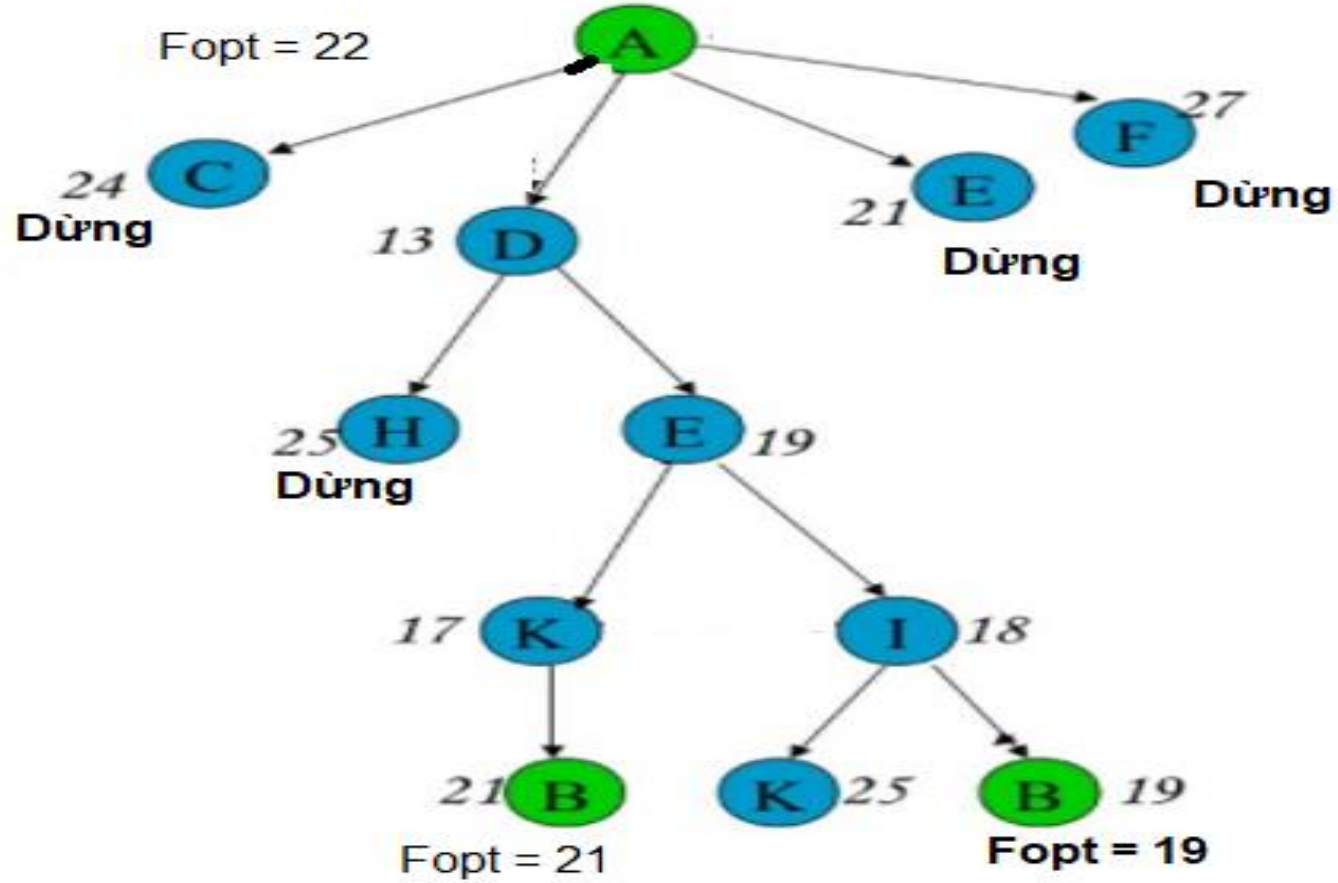
else *if* (g_k < *Fopt)*

Branch_And_Bound_Min (k+1); }

}

}

Hình ảnh cây tìm kiếm min theo nhánh cận:



Ý tưởng nhánh cận tìm max:

- Giả sử đã xây dựng được k thành phần của phương án x = (a_1, …, a_k, x_{k+1}, …, x_n).
- Đặt D(a_1,…, a_k) = {x∈D | x_i = a_i với i = 1..k}:
  - Giả sử tìm được g_k = $g(a_1, a_2,..,a_k)$ thỏa mãn:

    $f(x) \le g_k$ với mọi $x \in D(a_1,\ldots, a_k)$
  - Khi đó có max {f(x), $\forall x \in D(a_1,\ldots, a_k)$ } ≤ g_k
  - g_k gọi là cận trên của f(x) trên tập D(a_1,…, a_k).
- Gọi giá trị tối ưu hiện thời của bài toán tìm max là Fopt:

Nếu $g_k \le$ Fopt thì max {f(x), $\forall x \in D(a_1,\ldots, a_k)$ } ≤ Fopt

⇔ với mọi $x \in D(a_1,\ldots, a_k)$ không phải là phương án tối ưu

- Không cần phải tiếp tục phương án bộ phận (a_1,a_2,..,a_k) ⇔ Tập D(a_1,a_2,..a_k) bị loại bỏ khỏi quá trình duyệt.

Thuật toán nhánh cận tìm max:

Thuật toán Branch_And_Bound_Max (k) {

for $a_k \in A_k$ do { if (<chấp nhận a_k>){

$x_k = a_k$;

if (k==n) {

if (f(X) > Fopt) <Cập nhật kỷ lục>;}

else *if* (g_k > *Fopt)*

Branch_And_Bound_Max (k+1); }

}

}

Áp dụng cho bài toán cái túi:

Giải bài toán cái túi bằng phương pháp nhánh cận tìm max:

$f(X) = c_1x_1 + \ldots + c_nx_n \rightarrow \max,$

$X \in D, D = \{X \mid g(X) \leq b\}$

Trong đó,

- $X = (x_1, \ldots, x_n); x_i \in \{0, 1\}$
- $g(X) = a_1x_1 + \ldots + a_nx_n$;

Ví dụ bài toán cái túi:

$$\begin{cases} 5x_1 + 3x_2 + 10x_3 + 6x_4 \rightarrow \max \\ 3x_1 + 2x_2 + 5x_3 + 4x_4 \leq 8 \\ x_i \in \{0, 1\}, i = 1, 2, 3, 4 \end{cases}$$

Lời giải

- Giả sử $c_1/a_1 \geq \ldots \geq c_n/a_n$
- Xét phương án chọn đồ vật:

$$X = (x_1, \ldots, x_k, x_{k+1}, \ldots, x_n)$$

- Đặt b_k là trọng lượng còn lại của túi sau khi chọn đồ vật k:

$$b_o = b,\ b_k = b_{k-1} - a_k x_k$$

- Đặt δ_k là giá trị của túi sau khi chọn đồ vật thứ k:

$$\delta_0 = 0,\ \ \delta_k = \delta_{k-1} + c_k x_k$$

- Đặt $g_k = \delta_k + b_k(c_{k+1}/a_{k+1})$
- Đối với phương án X có

$$f(X) \leq g_k$$

Nếu $g_k \leq Fopt \Rightarrow f(X) \leq Fopt$

- *Không cần xây dựng tiếp các thành phần còn lại của phương án X vì không thể xây dựng X thành phương án tốt hơn phương án tối ưu Xopt tương ứng Fopt.*
- Giá trị g_k gọi là cận trên của phương án X.

Thuật toán nhánh cận giải bài toán cái túi

- **Bước 1**. *Sắp xếp đồ vật theo chiều giảm của tỉ số:*

$$c_1/a_1 \geq c_2/a_2 \geq \ldots \geq c_n/a_n$$

- **Bước 2**. Fopt = - ∞; Xopt = ∅;
- **Bước 3**(*Lặp*): X= ∅; $b_0 = b$; $\delta_0 = 0$;

Xét bài toán bộ phận cấp *k =1, 2,..,n:*

- *Chọn x_k bằng 1 hoặc 0;*
- *Tính trọng lượng còn lại của túi:*

$$b_k = b_{k-1} - a_k x_k$$

- *Nếu $b_k < 0$ thì dừng.*
- *Tính giá trị sử dụng của k đồ vật trong túi*: $\delta_k = \delta_{k-1} + c_k x_k$
- *Tính cận trên của phương án bộ phận cấp k (k < n)*:

$$g_k = \delta_k + b_k(c_{k+1}/a_{k+1})$$

- Nếu k = n thì

{ Nếu δ_k > Fopt thì cập nhật kết quả:

Fopt = δ_k; Xopt = X; }

- Nếu (k < n) và (g_k > ***Fopt***) xét bài toán bộ phận cấp k+1;

- **Bước 4** (*Trả lại kết quả*):
  - Giá trị tối ưu Fopt
  - Phương án tối ưu Xopt.

Ví dụ 3:

Giải bài toán cái túi sau bằng phương pháp nhánh cận:

$$\begin{cases} 5x_1 + 3x_2 + 10x_3 + 6x_4 \rightarrow \max \\ 3x_1 + 2x_2 + 5x_3 + 4x_4 \le 8 \\ x_i \in \{0, 1\},\ i = 1, 2, 3, 4 \end{cases}$$

Giải

Xác định bài toán: n = 4; b= 8

- *Sắp xếp các đồ vật theo chiều giảm của tỉ số:*

  $10/5 \ge 5/3 \ge 3/2 \ge 6/4$

$f(X) = 10x_1 + 5x_2 + 3x_3 + 6x_4$

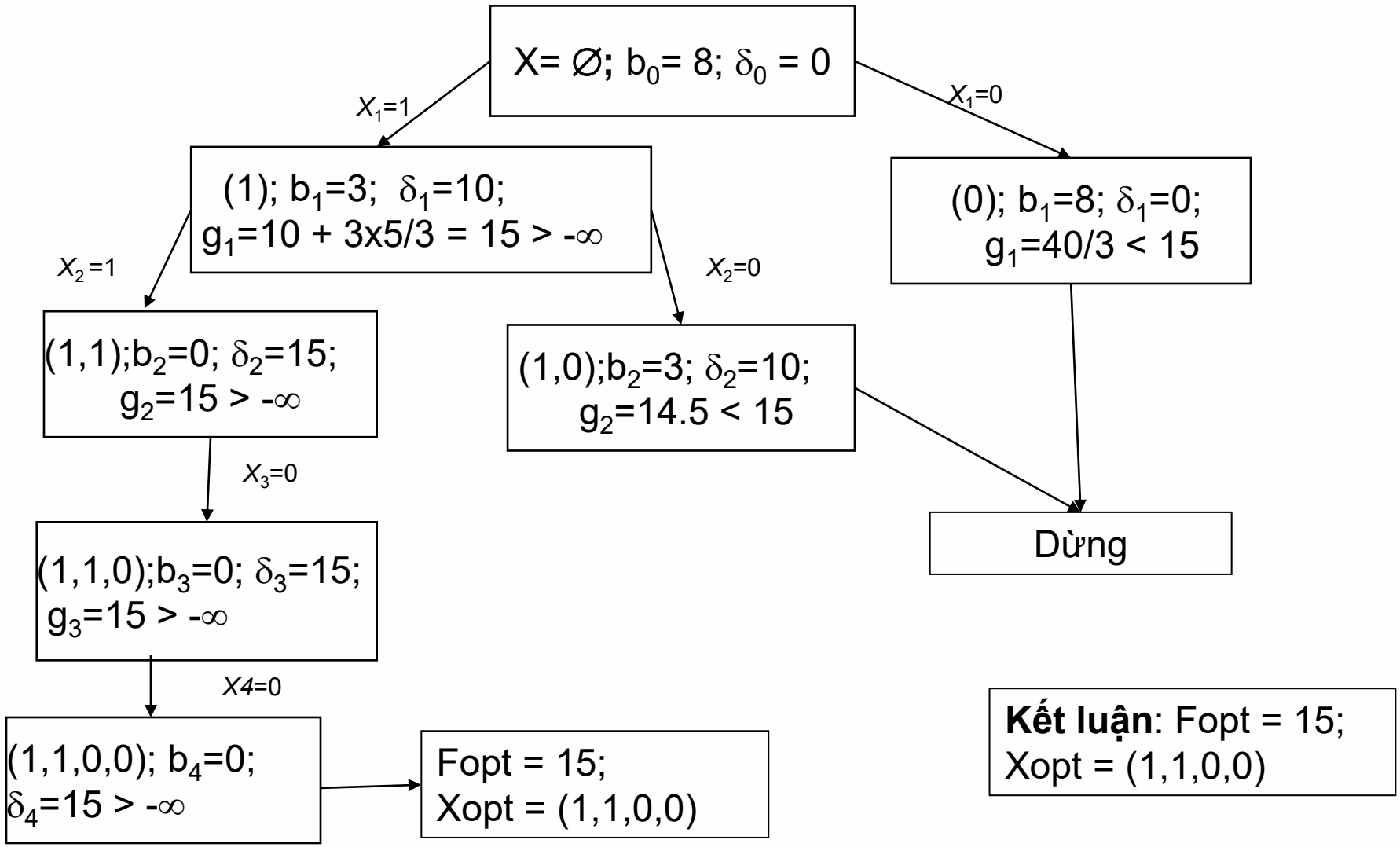
$g(X) = 5x_1 + 3x_2 + 2x_3 + 4x_4$

- **Khởi tạo**:

  Fopt = -∞; Xopt = ∅

- **Lập bảng**:

$a_1 = 5, c_1 = 10;\ a_2 = 3, c_2 = 5; a_3 = 2, c_3 = 3; a_4 = 4, c_4 = 6;$



Áp dụng cho bài toán người du lịch

Giải bài toán du lịch bằng phương pháp nhánh cận tìm min:

f(X) = $C[x_1][x_2] + \ldots + C[x_{n-1}][x_n] + C[x_n][x_1] \rightarrow$ **min, X** $\in$ **D**

$D = \{X = (x_1, \ldots, x_n) | x_1 = 1, x_i \neq x_j; i,j = 1..\ n\}$; $C = (c_{ij})$

Ví dụ bài toán người du lịch với ma trận chi phí C:

0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0

Lời giải

- Gọi c_{min} = min {*c*[i][j], *i, j = 1, 2,.., n, i≠j* } là giá trị nhỏ nhất trong các phần tử của ma trận chi phí.
- Giả sử có hành trình bộ phận qua k thành phố: $x_1 \rightarrow x_2 \rightarrow ... \rightarrow x_k$ (x_1 =1) với chi phí:

$$\delta_k = c[1][x_2] + c[x_2][x_3] + \ldots + c[x_{k-1}][x_k].$$

- Để phát triển hành trình bộ phận này thành hành trình đầy đủ, cần phải qua n-k thành phố còn lại và quay trở về thành phố 1. Như vậy, cần qua n-k+1 đoạn đường nữa.
- Vì mỗi đoạn đường đều có chi phí ≥ c_{min}, nên cận dưới của phương án bộ phận:

$$g_k = g(x_1,x_2,..,x_k) = \delta_k + (n-k+1).c_{min}.$$

Thuật toán nhánh cận giải bài toán NDL

- **Bước 1:**

Tính $c_{min} = Min \{c_{ij}\}$, i, j = 1, …, n, $i \neq j$;

- **Bước 2:** *(Khởi tạo)* Fopt = +∞; Xopt = ∅;
- **Bước 3** (*Lặp – Liệt kê hoán vị của n-1 phần tử từ 2 đến n*):

X = (1); $\delta_1 = 0$;

Xét các bài toán bộ phận cấp k = 2,..,n:{

- Chọn x_k trong các giá trị từ 2 đến n chưa sử dụng;
- Tính chi phí đến thành phố x_k: $\delta_k = \delta_{k-1} + c[x_{k-1}][x_k]$;
- Tính cận dưới của phương án bộ phận cấp k:

$$g_k = \delta_k + (n-k+1)*c_{min}.$$

Thuật toán nhánh cận (tiếp)

Bước 3 (tiếp):

- Nếu k = n thì

  Nếu δ_n + C[x_n][1] < Fopt thì cập nhật kết quả:

  Fopt = δ_n + C[x_n][1]; Xopt = X;

- Nếu (k < n) và (**g_k < Fopt**) xét bài toán bộ phận cấp k+1;

  //Nếu ($g_k \geq$ Fopt) thì không tiếp tục theo nhánh;

}

Bước 4 (*Trả lại kết quả*):

- Giá trị tối ưu Fopt
- Phương án tối ưu Xopt;

Ví dụ 4:

Giải bài người du lịch cho bởi ma trận chi phí dưới đây bằng phương pháp nhánh cận:

0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0

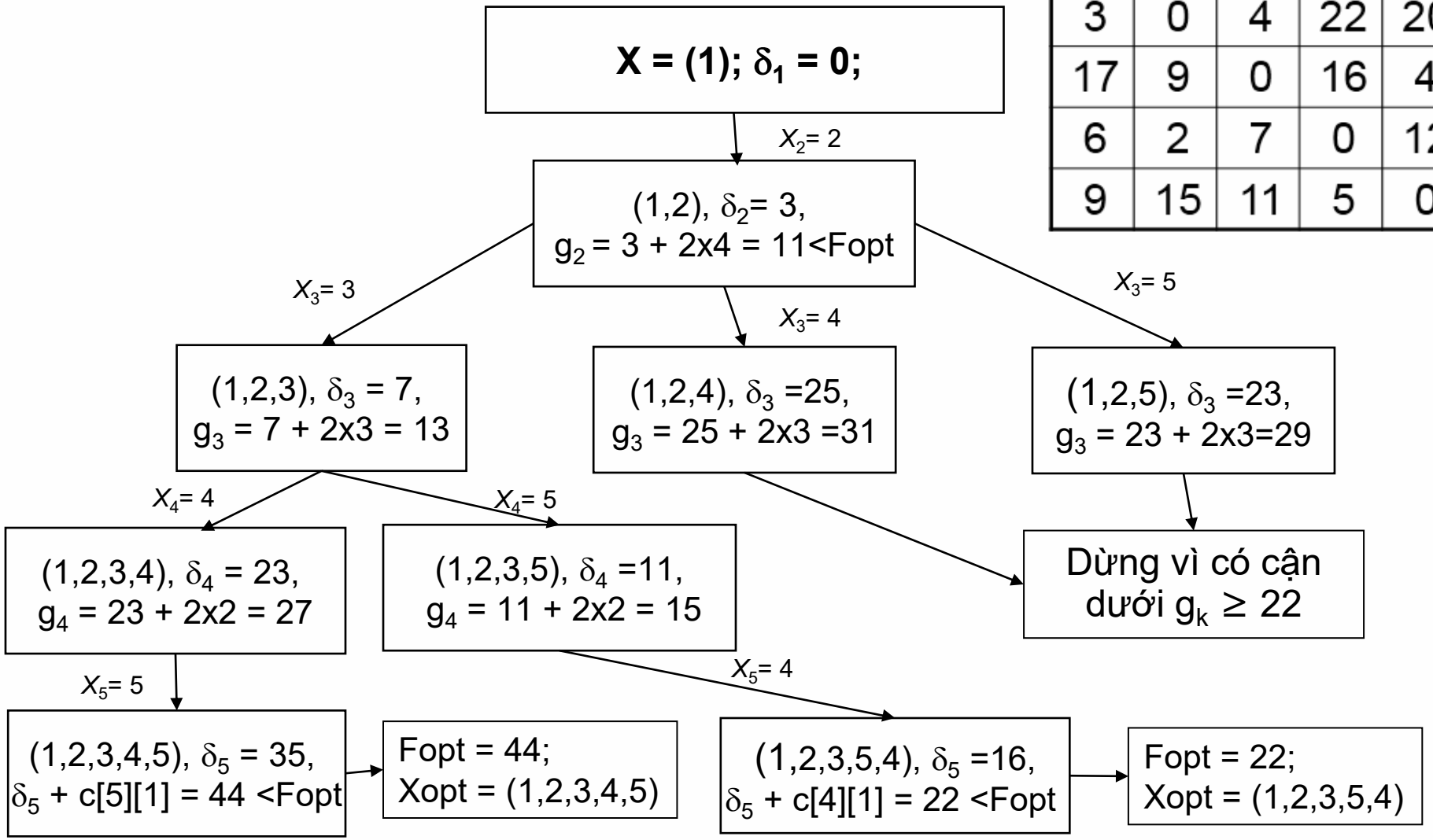
Giải

- **Xác định bài toán**:
  - n = 5
  - Cmin = 2
- **Khởi tạo**:

  Fopt = + ∞; Xopt = ∅;

Lập bảng:

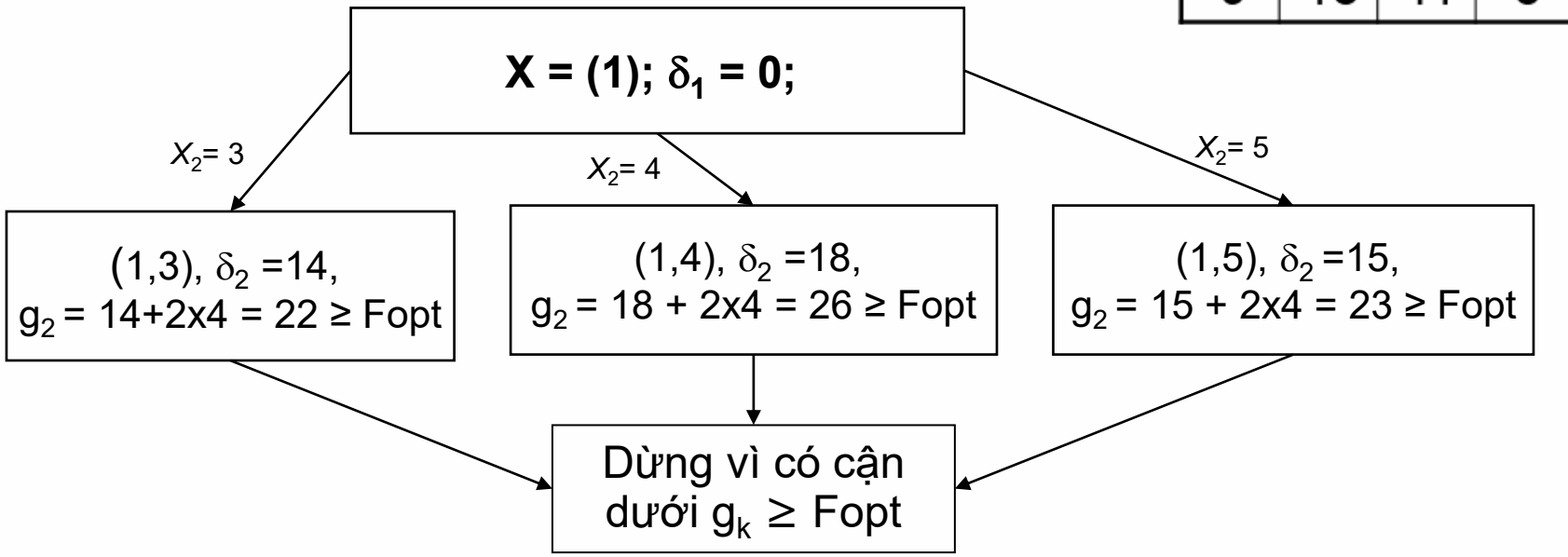
0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0



Lập bảng:

0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0

Fopt = 22;
Xopt = (1,2,3,5,4)



Kết luận: Fopt = 22;
Xopt = (1,2,3,5,4)

Ghi chú:

- Có thể chọn trước phương án xuất phát bằng cách sử dụng phương pháp tham lam:

- Chọn $x_1 = 1$;
- Trên hàng 1 của ma trận chi phí chọn $x_2 = j$ sao cho:

$$C[1][j] = \text{Min}\{c[1][t], 1 \le t \le n, t \ne 1\}$$

- Trên hàng x_{k-1} của ma trận chi phí chọn $x_k = j$ sao cho:

$$C[x_{k-1}][j] = \text{Min}\{c[x_{k-1}][t], 1 \le t \le n, t \ne x_1, \cdots, x_{k-1}\}$$

- Tiếp tục cho đến khi chọn được phương án Xopt = $(x_1, \ldots, x_n)$;

- Tính Fopt = $C[1][x_2] + \ldots + C[x_{n-1}][x_n] + C[x_n][1]$

Giải lại ví dụ 4:

Ma trận chi phí C:

0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0

Chọn trước phương án xuất phát:

Có:

- $x_1 = 1 \rightarrow x_2 = 2: 3$
- $x_2 = 2 \rightarrow x_3 = 3: 4$
- $x_3 = 3 \rightarrow x_4 = 5: 4$
- $x_4 = 5 \rightarrow x_5 = 4: 5$

$\Rightarrow$ Xopt = (1, 2, 3, 5, 4).

- Có $x_5 = 4 \rightarrow x_1 = 1: 6$

Tính Fopt = 22

Giải lại ví dụ 4 (tiếp):

0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0

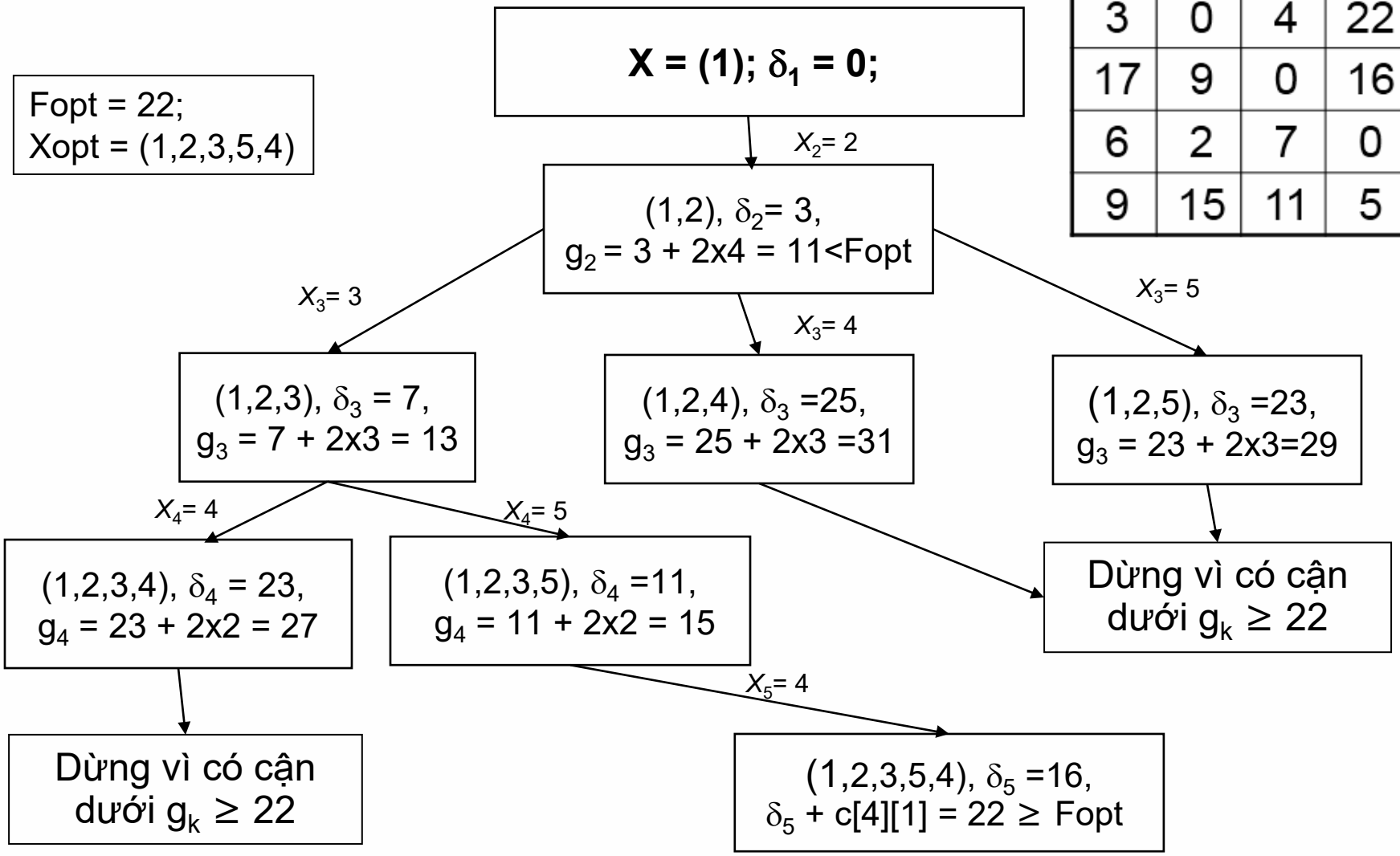
Có n = 5; Cmin = 2;

Khởi tạo: Fopt = 22; Xopt = (1, 2, 3, 5, 4)

Lập bảng:

Lập bảng:

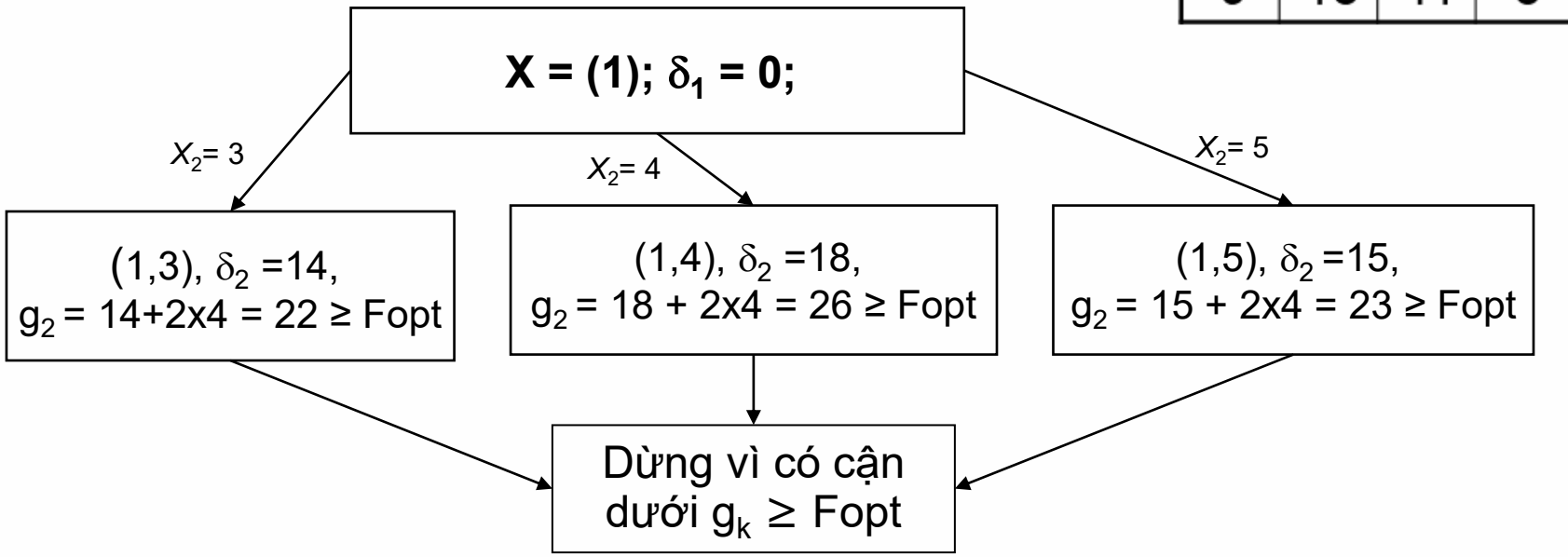
0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0



Lập bảng:

0	3	14	18	15
3	0	4	22	20
17	9	0	16	4
6	2	7	0	12
9	15	11	5	0

Fopt = 22;
Xopt = (1,2,3,5,4)



Kết luận: Fopt = 22;
Xopt = (1,2,3,5,4)

Cài đặt thuật toán nhánh cận giải bài toán NDL

Khai báo dữ liệu:

```
int n, pa[100], vs[100];
long c[100][100];
long Fopt = 1000000000;
Long cmin = 1000000000;
int Xopt[100];
```

- **Hàm tính chi phí của phương án pa[]:**

```
long Check(int pa[], int n){
  long s = 0; int i;
  for (i = 1; i < n; i++)
      s = s + c[pa[i]][pa[i+1]];
   s+= c[a[n]][1];
return (s);
  }
```

Duyệt nhánh cận tìm Xopt và Fopt

```
void HvDq(int k, long d) {
  if ( k> n) {long tmp= check(pa, n);
     if (Fopt> tmp) {mt= tmp;
    for (int i = 1; i <= n; i++)
          Xopt[i]= pa[i]; return;}
   for (int i = 2; i <= n; i++)
          if (vs[i] == 0){
      pa[k] = i; vs[i] = 1;
      d = d + c[pa[k-1]][pa[k]];
       long g = d + cmin*(n-k+1);
         if (g < mt) HvDq(k+1, d);
       vs[i] = 0;
       d = d - c[pa[k-1]][pa[k]];
                  }
 }
```

- **Hàm thực hiện nhánh cận**

```
void NhanhCan(){ int i, j;
   for (i=1; i<= n; i++)
      for(j= 1; j <=n ; j++)
         if (i != j && cmin > c[i][j])
              cmin = c[i][j];
   long d = 0;
   a[1] = 1; vs[1] = 1;
     HvDq(2, d);
   }
```

4.3 PHƯƠNG PHÁP QUI HOẠCH ĐỘNG

- Mở đầu
- Qui hoạch động
- Một số ví dụ

4.3.1 Mở đầu

- **Ví dụ 1:**

Dãy Fibonacci được định nghĩa bằng đệ qui:

$f_1 = f_2 = 1;$

$f_n = f_{n-1} + f_{n-2}$ với mọi $n > 2$.

Xác định phần tử thứ n của dãy Fibonacci?

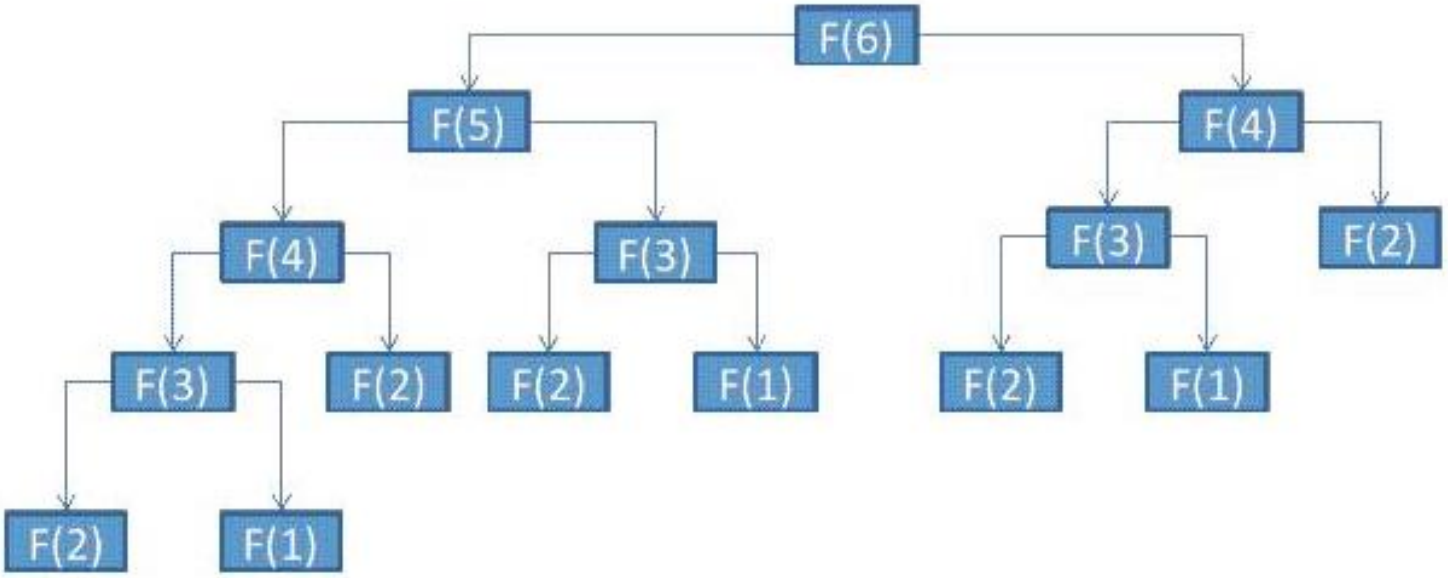
(1) Cài đặt đệ qui

```
long fibonacci(int n) {
   if (n <= 2) return 1;
   return (fibonacci(n-1) + fibonacci(n-2));
 }
```

Nhận xét:

- Quá trình tính toán có thể minh họa như mô hình cây tính toán.

Nhận xét:



Mô hình hoạt động của chương trình đệ qui với n = 6

Nhận xét:

- Với n = 6

$\Rightarrow$ Để tính F(6) chương trình sẽ gọi tiếp F(5) và F(4) để tính, …

$\Rightarrow$ chương trình phải tính 1 lần F(5), 2 lần F(4), 3 lần F(3), 5 lần F(2), 3 lần F(1).

- Độ phức tạp tính toán: $O(2^n)$

Nhận xét:

- Cài đặt đệ qui có thể không hiệu quả về mặt thời gian do tính toán trùng lặp

- Cài đặt đệ qui có thể không hiệu quả về mặt không gian do lưu trữ trùng lặp

(2) Cài đặt kiểu qui hoạch động 1:

```
long f[100];
long fibonacci(int n) {
   f[1] = f[2] = 1;
   for (int i = 3; i <= n; i++)
                f[i] = f[i-1] + f[i-2];
   return(f[n]);
 }
```

(2) Cài đặt kiểu qui hoạch động 2:

```
long fibonacci(int n) { long a, b, c;
   a = b = 1; c =1;
   for (int i = 3; i <= n; i++){
                c = a + b;
                 a = b; b = c;
   }
   return(c);
 }
```

Nhận xét:

- Quá trình tính F(n) được chia thành n giai đoạn dựa trên hệ thức truy hồi (công thức qui hoạch động):

$$F(n) = F(n-1) + F(n-2)$$

- Khởi tạo: F(1) = F(2) = 1;
- Kết quả: F(n);

Ví dụ 2: Bài toán cái túi

Bài toán:

Input:

- Trọng lượng túi b; Số lượng đồ vật n;
- Dãy trọng lượng a[1], ..., a[n];
- Dãy giá trị c[1], ..., c[n];

Output:

- Giá trị lớn nhất mt các đồ vật có thể xếp vào túi;
- Các đồ vật được chọn: pa[1], ..., pa[n];

(1) Giải thuật duyệt toàn thể

a) Phương pháp giải

- Mỗi phương án chọn đồ vật tương ứng một xâu nhị phân X[] độ dài n, trong đó:

  X[i] = 0 nếu đồ vật i không được chọn,

  X[i] = 1 nếu đồ vật i được chọn,

- Duyệt tất cả các xâu nhị phân:

  $\sum X[i]*a[i] \leq b$ và $\sum X[i]*c[i] \rightarrow$ Max

b) Thiết kế giải thuật

Khai báo dữ liệu:

```
int n;
long b, a[100], c[100];
long mt = -1;
int pa[100], x[100];
```

//Hàm kiểm tra điều kiện

```
long Check(int x[], int n){
  long s = 0, tmp = 0; int i;
  for (i = 1; i <= n; i++) s = s + x[i]*a[i];
  if (s <= b)
        for (i = 1; i <= n; i++) tmp = tmp + x[i]*c[i];
  else tmp = -1;
  return (tmp);
  }
```

// Hàm duyệt toàn thể tìm mt

```
int x[100], pa[100];
void XnpKt(int x[], int n) { int i;
   for (i = 1; i<= n; i++) x[i] = 0;
   while (1) {long tmp = Check(x, n);
     if (mt < tmp) {
        mt = tmp;
        for (i = 1; i<= n; i++) pa[i] = x[i];
       }
```

```
 i = n;
while (i > 0 && x[i] == 1) {x[i] = 0; i--;}
If (i == 0) return; else x[i] = 1;
}

}
```

c) Đánh giá độ phức tạp

Gọi T(n) là thời gian giải bài toán cái túi với n đồ vật.

$\Rightarrow T(n) = O(n2^n)$

(2) Cài đặt theo nhánh cận

Khai báo dữ liệu:

```
int n;
long b, a[100], c[100];
long mt = -1;
int pa[100], x[100];
```

//Hàm kiểm tra điều kiện

```
long Check(int x[], int n){
  long s = 0, tmp = 0; int i;
  for (i = 1; i <= n; i++) s = s + x[i]*a[i];
  if (s <= b)
        for (i = 1; i <= n; i++) tmp = tmp + x[i]*c[i];
  else tmp = -1;
  return (tmp);
  }
```

// Hàm duyệt nhánh cận tìm mt

```
void XnDq(int k, long w, long d) {
   if ( k > n) { long tmp = Check(x, n);
    if (mt < tmp){ mt = tmp;
     for int i = 1; i <= n; i++) pa[i]= x[i]; return;}
    for (int i = 1; i >= 0; i--) {x[k] = i; w = w – x[k]*a[k];
      d = d + x[k]*c[k]; long g = d + w*(c[k+1]/a[k+1]);
          if (w >= 0 && g > mt) XnDq(k+1, w, d);
         w = w + x[k]*a[k]; d = d - x[k]*c[k];
                    }
   }
```

// Hàm thực hiện nhánh cận

```
Void NhanhCan(){ int i, j; long tmp;
   for (i= n; i> 1; i--)
      for(j= 1; j < i; j++) if (c[j]/a[j] < c[j+1]/a[j+1]){
  tmp = c[j]; c[j] = c[j+1]; c[j+1] = tmp;
 tmp = a[j]; a[j] = a[j+1]; a[j+1] = tmp; }
    long w = b; long d = 0; mt = -1;
       XnDq(1, w, d);
    }
```

(2) Giải thuật qui hoạch động

a) Phương pháp giải

- Gọi F(i, j) là giá trị lớn nhất khi chọn các đồ vật 1, 2, ..., i sao cho tổng trọng lượng của chúng không vượt quá j, với j = 0, 1, ..., b; i = 0, 1, ..., n;

- Có F(i, 0) = 0 với mọi i; F(0, j) = 0 với mọi j;

$$F(i, j) = \text{Max}\{F(i-1, j), F(i-1, j-a(i)) + c(i)\} \quad (*)$$

với $j - a(i) \geq 0$;

- Kết quả: mt = Max{F(n, j)}, với j = 0, 1, ..., b;

Tính pa[]:

- Để tìm pa[] đưa vào hàm Pr(i, j);

- Tính Pr(i, j):

- Khởi tạo Pr(i, j) = 0 với mọi i, j;

- Pr(i, j) = i nếu F(i, j) = F(i-1, j – a[i]) + c(i);

Test bài toán:

Input:

- Trọng lượng túi b = 5; Số lượng đồ vật n = 3;
- Dãy trọng lượng a[1] = 2, a[2] = 3, a[3] = 3;
- Dãy giá trị c[1] = 4, c[2] = 8, c[3] = 13;

Test:

- Lập bảng tính F[i][j] và Pr[i][j];
- Tính mt;
- Tính: pa[1], ..., pa[k];

Kết quả test:

- mt = 15;
- Số đồ vật được chọn: k = 2;
- Đồ vật được chọn: 1, 3;

b) Nhận xét:

- Thay vì tính trực tiếp mt ta tham số hóa bài toán dưới dạng F(i, j);
- Tìm công thức qui hoạch động biểu diễn mối liên hệ giữa F(i, j) với giá trị của nó đã biết trong các giai đoạn tính toán trước đó;
- Độ phức tạp tính toán: O(n*b);
- Chỉ giải bài toán với các dữ liệu số nguyên;

4.3.2 Qui hoạch động

- Cơ sở của phương pháp
- Lược đồ tính toán

1. Cơ sở của phương pháp

- **Ý tưởng của phương pháp**

- Sử dụng nguyên lý: “chia để trị”

+ Chia bài toán thành các bài toán con cùng dạng;

+ Kết quả của bài toán là tổng hợp kết quả của các bài toán con;

- Cách tiếp cận: “dưới-lên”

+ Tính toán trong các trường hợp đơn giản;

+ Kết quả của giai đoạn tiếp theo được tính dựa trên các kết quả đã biết;

Phạm vi áp dụng:

+ Các bài toán có được bằng việc tổ hợp các nghiệm của các bài toán con;

+ Các bài toán tối ưu hoá rời rạc;

+ Tổng quát:

- Các hệ thống có quá trình phát triển theo thời gian
- Bài toán tối ưu với hàm mục tiêu là tổng của một số hàm tách biến và được tính qua nhiều giai đoạn

Nguyên tắc của phương pháp:

(1) Nguyên tắc đánh số các giai đoạn

- Chia bài toán đã cho thành các giai đoạn và đánh số từ trên xuống hoặc từ dưới lên;
- Tính giá trị tại giai đoạn xuất phát (giá trị khởi tạo);
- Tính giá trị tại các giai đoạn sau sẽ dùng kết quả đã có ở các giai đoạn trước nó.

(2) Nguyên tắc tham số hóa bài toán

- Chuyển bài toán thành hàm F() phụ thuộc vào tham số giai đoạn;

- Tìm công thức qui hoạch động biểu diễn quan hệ phụ thuộc của F() giữa các giai đoạn

(3) Nguyên tắc lồng

- Lồng bài toán ban đầu vào một bài toán rộng hơn hay một họ các bài toán, và do đó bài toán ban đầu là một trường hợp riêng của họ bài toán này.
- Giải họ bài toán nhờ có các thông số đã biết;
- Thử kết quả của họ bài toán với các thông số khác nhau cho tới một lúc tìm được thông số ứng với bài toán xuất phát thì kết thúc.

(4) Nguyên tắc tối ưu (Bellman)

- Trong một dãy tối ưu của các lựa chọn thì một dãy con của nó cũng là tối ưu;

- Tổng hợp các trường hợp tối ưu riêng lẻ sẽ cho kết quả tối ưu chung;

2. Lược đồ tính toán

- Phân tích bài toán (biểu diễn bài toán dưới dạng một bài toán nhiều giai đoạn);
- Xây dựng giải pháp qui hoạch động (lập công thức truy hồi – qui hoạch động);
- Lập bảng tính toán (sử dụng các mảng để tính toán các giá trị theo kiểu dưới-lên);
- Tổng hợp kết quả (kiến tạo một lời giải cho bài toán từ các thông tin đã tính toán);

4.3.3 Một số bài toán cụ thể

- Bài toán cái túi (*Knapsack Problems)*
- Phép nhân tổ hợp nhiều ma trận

1. Cài đặt bài toán cái túi

- Khai báo dữ liệu:

```
int n;
long b, a[100], c[100];
long mt = -1, f[100][100];
int pr[100][100], pa[100];
```

//Hàm Qui hoạch động:

```
void Qhd(){
    long j;
    int i;
  for (i = 0; i<= n; i++)
     for (j = 0; j<= b; j++){f[i][j] = 0; pr[i][j] = 0;}
```

```
for (i = 1; i<= n; i++)
    for (j = 0; j<= b; j++){
      f[i][j] = f[i-1][j];  pr[i][j] = 0;
      if (j - a[i] >= 0 && f[i][j] < f[i-1][j-a[i]] + c[i]){
        f[i][j] = f[i-1][j-a[i]] + c[i];  pr[i][j] = i;
        }
    }
}
```

//Hàm xuất

```
void Xuat(){
    long cs = 0, j; int i;
    MoTepGhi();
    for (j = 0; j<= b; j++)
       if (mt < f[n][j]) {mt = f[n][j]; cs = j;}
    if (cs > 0)
```

```
for (i = n; i >= 1; i--) { pa[i] = pr[i][cs];
        if (cs-a[i] > 0) cs- = a[i]; else break;
        }
fo << mt << endl;
for (int i= 1;i<= n; i++)
        if (pa[i] > 0) fo <<pa[i] <<"  ";
fo.close();
}
```

2. Phép nhân tổ hợp nhiều ma trận

Bài toán

- Cần tính $M = M_1 \times M_2 \times \ldots \times M_n$

Trong đó:

M_i là ma trận cấp $m[i-1] \times m[i]$ $(i = 2..n)$

- Hãy xác định thứ tự thực hiện các phép nhân ma trận sao cho số phép tính là tối thiểu.

Ví dụ:

- Tính $M = M_1 \times M_2 \times M_3$

Với M_1 cấp [10×20]; M_2 cấp [20×50];

M_3 cấp [50×5] $\Rightarrow$ M cấp [10x5]

- Tính thông thường:

$M = (M_1 \times M_2) \times M_3$ có số phép toán là:

$10 \times 20 \times 50 + 10 \times 50 \times 5 = 12500$

$M = M_1 \times (M_2 \times M_3)$ có số phép toán là:

$10 \times 20 \times 5 + 20 \times 50 \times 5 = 6000$

Phân tích bài toán:

- Gọi F(l, r) là số phép tính tối thiểu của bài toán nhân ma trận: $M_l \times M_{l+1} \times \ldots \times M_r$, với $l \leq r$;
- Kết quả của bài toán ban đầu là F(1, n);

- Cần tìm vị trí k cua phép toán thực hiện cuối cùng của phép nhân

$$(M_l x M_{l+1} x \ldots x M_{k-1}) \; x \; (M_k x M_{k+1} x \ldots x M_r)$$

$\Rightarrow$ Vt(l, r) = k;

Giải pháp qui hoạch động:

- **Trường hợp 1**:

  Khi r = l :

  F(l, l) = 0; Vt(l, l) = 0;

- **Trường hợp 2**:

  Khi r = l+1:

  F(l, l+1) = m[l-1] * m[l] * m[l+1];

  vt(l, l+1) = l+1;

Trường hợp 3: Khi r > l+1:

$F(l, r) = \min\{ F(l, v-1) + m[l-1] * m[v-1] * m[r] + F(v, r) \mid l+1 \le v \le r \}$;

(v là các vị trí phép toán thực hiện cuối cùng khác nhau)

$= F(l, v'-1) + m[l-1] * m[v'-1] * m[r] + F(v', r)$; (v' là vị trí tối ưu trong số các vị trí của v);

$\Rightarrow Vt(l, r) = v'$;

Cài đặt hàm Qui hoạch động

```
void Qhd(){
 int l, r, v;
 for (l = 1; l <= n; l++)
   for (r = 1; r <= n; r++) {
     f[l][r] = 1000000;     vt[l][r] = 0;     }
for (int r =1; r <= n; r++)
  for (int l = r; l >= 1; l--)
    If (r == l) f[l][l] = 0;
```

```
else if (r == l+1) {
        f[l][r] = m[l-1]*m[l]*m[l+1]; vt[l][r] = l+1;}
      else
   for (v = l+1; v <= r; v++)
     if (f[l][r] > f[l][v-1] + m[l-1]*m[v-1]*m[r] +
  f[v][r]) {
 f[l][r] = f[l][v-1] + m[l-1]*m[v-1]*m[r] + f[v][r];
 vt[l][r] = v; }
}
```

Nhận xét:

- Độ phức tạp tính toán: $O(n^3)$

Yêu cầu:

Sinh viên tự hoàn chỉnh các chương trình:

+ Bài toán cái túi;

+ Bài toán nhân tổ hợp các ma trận;

Thảo luận


THANK YOU