Một số vấn đề cơ sở của Tin học

Cấu trúc dữ liệu và thuật giải

Giáo viên: Tạ Thúc Nhu

Khoa CNTT trường ĐH Lạc Hồng

QUY HOẠCH ĐỘNG

DYNAMIC PROGRAMMING

Tổng quan về phương pháp Quy hoạch động

- Quy hoạch động thường dùng giải các bài toán tối ưu có bản chất đệ qui
- Việc tìm nghiệm tối ưu của bài toán đã cho được thực hiện dựa trên việc tìm nghiệm tối ưu của các bài toán con.
- Kết quả của các bài toán con được ghi nhận lại để phục vụ cho việc giải các bài toán lớn hơn và giải được bài toán yêu cầu.

3

Ví dụ minh họa:

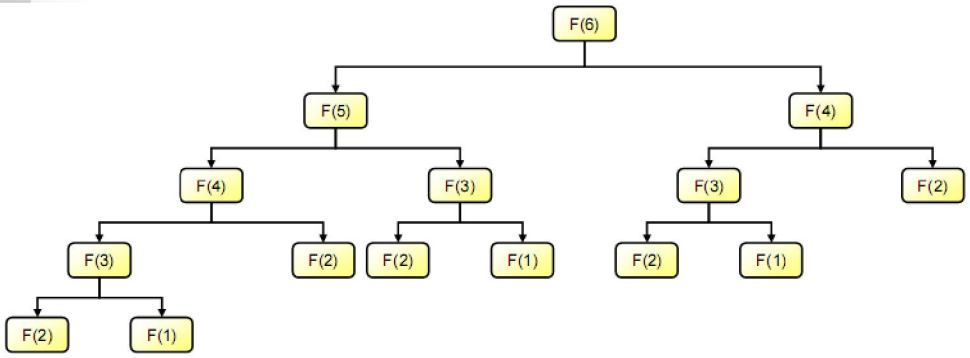
Dãy Fibonnaci là dãy các số nguyên dương được định nghĩa như sau:

$$F[i] = \begin{cases} 1 & i \le 2 \\ F[i-2] + F[i-1] & i > 2 \end{cases}$$

```
int F(int i)
{
    if (i <= 2) return 1;
    return F(i-2) + F(i-1);
}
void main()
{
    cout<<F(6)<<endl;
}
```

```
int F[100]; //Bang phuong an
void main()
{
    F[1] = F[2] = 1;
    for(int i = 3; i <= 6; i++)
        F[i] = F[i-2] + F[i-1];
    cout << F[6]<<endl;
}
```

4



5

Quá trình giải bài toán bằng quy hoạch động

1. **Tìm công thức đệ qui biểu diễn nghiệm tối ưu của bài toán lớn thông qua nghiệm tối ưu của các bài toán con.**
   i. Xác định tham số thể hiện kích thước bài toán
   ii. Lập công thức tính toán kết quả bài toán theo tham số kích thước
   iii. Xác định kết quả bài toán con nhỏ nhất.
2. **Tổ chức dữ liệu lưu trữ kết quả tính toán (Bảng phương án)**
3. **Dựa vào kết quả ghi nhận truy vết tìm ra nghiệm tối ưu.**

6

Bài toán ba lô (knapsack)

Cho n gói hàng. Gói hàng thứ i có khối lượng là A[i] và giá trị C[i]. Cần chọn những gói hàng nào để bỏ vào một ba lô sao tổng giá trị của các gói hàng đã chọn là lớn nhất nhưng tổng khối lượng của chúng không vượt quá khối lượng M cho trước. Mỗi gói chỉ chọn 1 hoặc không chọn.

Ví dụ: n = 5; M = 13

i	1	2	3	4	5
A[i]	3	4	5	2	1
C[i]	4	5	6	3	1

Tổng giá trị của các gói hàng bỏ vào ba lô: 16
Các gói được chọn: 1(3, 4) 2(4, 5) 3(5, 6) 5(1, 1)

7

Tham số thể hiện kích thước bài toán

- Kết quả bài toán là tổng giá trị lớn nhất của các món hàng được chọn trong n món sao cho tổng khối lượng không lớn hơn M cho trước, ký hiệu là F(n)
- Tham số thể hiện kích thước bài toán là số món hàng n
- Giá trị của F(n) có thể được tính từ giá trị của F(n-1) cộng thêm hoặc không cộng thêm giá trị của món hàng thứ n nhưng tổng khối lượng không lớn hơn M.
- Nếu chọn thêm món hàng thứ n thì tổng khối lượng được chọn trong (n-1) món hàng không lớn hơn (M-A[n])
- Suy ra bài toán có 2 tham số: số món hàng và khối lượng giới hạn

8

Lập công thức đệ qui

Gọi F(i, v) là tổng giá trị lớn nhất của các gói hàng được chọn trong i gói hàng sao cho tổng khối lượng không lớn hơn v .

- Trường hợp A[i] > v:

  F(i, v) = F(i -1, v)

- Trường hợp A[i] <= v:
  - Nếu gói hàng thứ i không được chọn thì:

    F(i, v) = F(i -1, v)
  - Nếu gói hàng thứ i được chọn thì:

    F(i, v) = F(i -1, v – A[i]) + C[i]

  ➔ F(i, v) = Max{ F(i -1, v); F(i -1, v – A[i]) + C[i]) }

- Bài toán nhỏ nhất ứng với i = 0 ta có: F(0, v) = 0

9

Xây dựng bảng phương án:

- **Cấu trúc bảng phương án:**
  - Dùng mảng F[0..n][0..M] chứa giá trị của các F(i, v)
- **Cách tính giá trị trên bảng phương án:**
  - Điền số 0 cho các ô trên dòng 0
  - Sử dụng công thức đệ qui và giá trị trên dòng (i -1) để tính dòng i
    - **Trường hợp A[i] > v:**

      F(i, v) = F(i -1, v)
    - **Trường hợp A[i] <= v:**

      F(i, v) = Max{ F(i -1, v); F(i -1, v – A[i]) + C[i]) }

10

Ví dụ bảng phương án:

- Trường hợp A[i] > v: F(i, v) = F(i -1, v)
- Trường hợp A[i] <= v: F(i, v) = Max{ F(i -1, v); F(i -1, v – A[i]) + C[i]) }

C	A	v / i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3	1	0	0	0	4	4	4	4	4	4	4	4	4	4	4
5	4	2	0	0	0	4	5	5	5	9	9	9	9	9	9	9
6	5	3	0	0	0	4	5	6	6	9	10	11	11	11	15	15
3	2	4	0	0	3	4	5	7	8	9	10	12	13	14	15	15
1	1	5	0	1	3	4	5	7	8	9	10	12	13	14	15	16

11

Thuật toán tạo bảng phương án

```
void TaoBangPhuongAn(F[0..n][0..M])
{
    for (v=0; v <= M; v++) F[0, v] = 0; // Điền số 0 cho dòng 0 của bảng
    for (i = 1; i <= n; i++)
        for (v=0; v <= M; v++)
        {
            F[i, v] = F[i-1, v];
            If (A[i] <= v && F[i, v] < F[i-1, v - A[i] ] + C[i])
                F[i, v] = F[ i-1, v - A[i] ] + C[i];
        }
}
```

12

Truy vết tìm lại các gói hàng đã chọn

Bắt đầu từ ô F[n, M] trên dòng n ta dò ngược về dòng 0 theo nguyên tắc:

- Nếu F[i, v] <> F[i–1, v]
  thì gói thứ i được chọn, ta truy tiếp ô F[i-1, v-A[i]].
- Nếu F[i, v] = F[i–1, v]
  thì gói thứ i không được chọn, ta truy tiếp ô F[i-1, v].

C	A	v / i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3	1	0	0	0	4	4	4	4	4	4	4	4	4	4	4
5	4	2	0	0	0	4	5	5	5	9	9	9	9	9	9	9
6	5	3	0	0	0	4	5	6	6	9	10	11	11	11	15	15
3	2	4	0	0	3	4	5	7	8	9	10	12	13	14	15	15
1	1	5	0	1	3	4	5	7	8	9	10	12	13	14	15	16

13

Thuật toán truy vết tìm lại các gói hàng đã chọn

```
void TruyVet(F[0..n][0..M])
{   Bắt đầu từ ô F[n, M] trên dòng n:  i = n; v = M;
    for (; i > 0; i --)
        if (F[i, v] != F[i-1, v])
        {
            <Món hàng thứ i được chọn >;
            v = v – A[i];
        }
}
```

14

Bài toán ba lô 2

Cho n loại hàng. Món hàng thuộc loại hàng i có khối lượng A[i] và giá trị C[i]. Số lượng các món hàng của mỗi loại không hạn chế. Cần chọn các món hàng trong từng loại để bỏ vào một ba lô sao cho tổng giá trị của các món hàng đã chọn là lớn nhất nhưng tổng khối lượng của chúng không vượt quá khối lượng M cho trước. Cho biết số lượng món hàng từng loại hàng được chọn

Ví dụ: n = 5; M = 13

i	1	2	3	4	5
A[i]	3	4	5	2	1
C[i]	4	5	6	3	1

Tổng giá trị của các món hàng bỏ vào ba lô: 19
Các món được chọn:
1 gói hàng loại 1 có khối lượng 3 và giá trị 4
5 gói hàng loại 4 có khối lượng 2 và giá trị 3

15

Xác định công thức đệ quy

Gọi F(i, v) là tổng giá trị lớn nhất của các món hàng được chọn sao cho tổng khối lượng <= v trong i loại hàng.

- Trường hợp A[i] > v: F(i, v) = F(i-1, v)
- Trường hợp A[i] <= v:
  - Nếu loại hàng i không được chọn thì:

    F(i, v) = F(i-1, v)
  - Nếu có k món hàng loại i được chọn: (1 <= k <= v/A[i])

    F(i, v) = F(i-1, v – A[i]*k) + C[i]*k

  Do đó:

  F(i, v) = Max{F(i-1, v – A[i]*k) + C[i]*k } k ∈[0, v/A[i]]
- Bài toán nhỏ nhất ứng với i = 0 hay v=0 ta có: F(0, v) = 0

16

Công thức đệ quy

Gọi F(i, v) là tổng giá trị lớn nhất của các món hàng được chọn có tổng khối lượng <= v trong i loại hàng đầu tiên

- Với i = 0 : F(i, v) = 0
- Với i > 0 :
  - A[i] > v : F(i, v) = F(i - 1, v)
  - A[i] <= v : F(i, v) = Max{ F(i-1, v – A[i]*k) + C[i]*k }
    với k ∈[0, v/A[i]]

17

Xây dựng bảng phương án:

- **Cấu trúc bảng phương án:** dùng 2 mảng
  - Mảng F[0..n][0..M]: F[i, v] chứa giá trị của các F(i, v)
  - Mảng S[1..n][1..M]: S[i,v] chứa số món hàng loại i được chọn
    - Nếu F(i, v) = F(i - 1, v): S[i, v] = 0
    - Ngược lại S[i, v] = k
- **Cách tính giá trị trên bảng phương án:**
  - Điền số 0 cho các ô trên dòng 0 và cột 0 của bảng F
  - Sử dụng công thức đệ quy và giá trị trên dòng i -1 để tính dòng i của bảng F và bảng S

18

Ví dụ: Lập bảng phương án F

- Với i > 0 :
  - A[i] > v : F(i, v) = F(i - 1, v)
  - A[i] <= v : F(i, v) = Max{ F(i-1, v – A[i]*k) + C[i]*k }
    với k ∈[0, v/A[i]]

Bảng F[i, v]

C[i]	A[i]	i \ v	0	1	2	3	4	5	6	7	8	9	10	11	12	13
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3	1	0	0	0	4	4	4	8	8	8	12	12	12	16	16
5	4	2	0	0	0	4	4	5	8	9	9	12	13	14	16	17
6	5	3	0	0	0	4	4	5	8	9	10	12	13	14	16	17
3	2	4	0	0	0	4	4	7	8	10	11	13	14	16	17	19
1	1	5	0	0	1	4	5	7	8	10	11	13	14	16	17	19

19

Ví dụ lập bảng phương án S

Bảng F[i, v]

C[i]	A[i]	i \ v	0	1	2	3	4	5	6	7	8	9	10	11	12	13
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	3	1	0	0	0	4	4	4	8	8	8	12	12	12	16	16
5	4	2	0	0	0	4	4	5	8	9	9	12	13	14	16	17
6	5	3	0	0	0	4	4	5	8	9	10	12	13	14	16	17
3	2	4	0	0	0	4	4	7	8	10	11	13	14	16	17	19
1	1	5	0	0	1	4	5	7	8	10	11	13	14	16	17	19

Bảng S[i, v]

C[i]	A[i]	i \ v	1	2	3	4	5	6	7	8	9	10	11	12	13
4	3	1	0	0	1	1	1	2	2	2	3	3	3	4	4
5	4	2	0	0	0	0	1	0	1	1	0	1	2	0	1
6	5	3	0	0	0	0	0	0	0	1	0	0	0	0	0
3	2	4	0	0	0	0	1	0	2	1	3	2	4	3	5
1	1	5	0	1	0	1	0	0	0	0	0	0	0	0	0

20

Thuật toán tạo bảng phương án

```
void TaoBangPhuongAn(F[0..n][0..M], S[1..n][1..M])
{   <Điền số 0 cho dòng 0 và cột 0 của bảng F[0..n][0..M]>;
    for (i = 1; i <= n; i++)
        for (v=1; v <= M; v++)
        {   F[i, v] = F[i-1, v];
            S[i, v] = 0;
            If (v >= A[i])
                for(k = 1; k <= v/A[i]; k++)
                    if (F[i, v] < F[i-1, v - A[i]*k ] + C[i]*k)
                    {   F[i, v] = F[i-1, v - A[i]*k ] + C[i]*k;
                        S[i, v] = k;
                    }
        }
}
```


Truy vết tìm lại các gói hàng đã chọn

Bắt đầu từ ô S[n, M] trên dòng n ta dò ngược về dòng 1 theo nguyên tắc:

- Nếu S[i, v] <> 0 thì :
  - Loại hàng i được chọn với số lượng là S[i, v]
  - Truy tiếp ô S[i-1, v - S[i, v]*A[i]].
- Nếu S[i, v] = 0 thì :
  - Loại hàng i không được chọn,
  - Truy tiếp ô S[i-1, v].

C[i]	A[i]	v / i	1	2	3	4	5	6	7	8	9	10	11	12	13
4	3	1	0	0	1	1	1	2	2	2	3	3	3	4	4
5	4	2	0	0	0	0	1	0	1	1	0	1	2	0	1
6	5	3	0	0	0	0	0	0	0	1	0	0	0	0	0
3	2	4	0	0	0	0	1	0	2	1	3	2	4	3	5
1	1	5	0	1	0	1	0	0	0	0	0	0	0	0	0

Thuật toán truy vết tìm lại các gói hàng đã chọn

```
void TruyVet(S[1..n][1..M])
{   i = n; v = M
    while (i > 0)
    {
        if (S[i, v] != 0)
        {
            <in số lượng gói hàng i trong S[i, v] >;
            v = v – S[i, v]*A[i];
        }
        i = i – 1;
    }
}
```


Bài toán dãy con có tổng chia hết cho k

Cho một dãy A gồm n số nguyên và một số nguyên dương k. Hãy tìm một dãy con (không nhất thiết phải liên tiếp nhau) dài nhất có tổng các số chia hết cho số k.

Ví dụ: n = 6 và k = 5

I	1	2	3	4	5	6
Dãy A[i]	11	6	7	12	20	8

Chiều dài dãy con: 4
Các phần tử được chọn là : 1 2 5 6
Có giá trị tương ứng là : : 11 6 20 8

Nhắc lại phép toán mod

Giả sử r = a mod k và z = b mod k

Ta có:

1. (a + b) mod k = (r + z) mod k
2. (-r) mod k = (-r + k) mod k
3. (r + z) mod k = v → z = (v – r) mod k = (v – r + k) mod k

I	1	2	3	4	5	6
A[i]	11	6	7	12	20	8
A[i] = A[i] % 5	1	1	2	2	0	3
	0	0	0	3	4	4

25

Xác định tham số

Gọi F(i) = chiều dài dãy con dài nhất trong miền [1..i] có tổng chia hết cho k.

1. Nếu dãy con dài nhất không có A[i] thì F(i) = F(i-1)

   Với F(i-1) = chiều dài dãy con dài nhất trong miền [1..i-1] có tổng chia hết cho k

2. Nếu dãy con dài nhất có chứa A[i]: thì F(i) = F(i-1) + 1

   Gọi r = A[i] mod k

   – Nếu r = 0: thì F(i-1) = chiều dài dãy con dài nhất trong miền [1..i-1] có tổng chia hết cho k

   – Nếu r > 0 : do (r + k - r) mod k = 0 nên F(i-1) bằng chiều dài dãy con dài nhất trong miền [1..(i-1)] có tổng chia với k dư (k-r)

Tham số thể hiện kích thước của bài toán: kích thước miền và số dư của tổng chia với k

26

Lập công thức đệ qui

Gọi F(i, v) = chiều dài dãy con dài nhất trong miền [1..i] có tổng chia với k dư là v.

1. Nếu dãy con dài nhất không có A[i] thì F(i, v) = F(i-1, v)
2. Nếu dãy con dài nhất có chứa A[i]:

   Gọi r = A[i] mod k ta có F(i, v) = F(i-1, v - r) + 1
   - Nếu v – r = 0 : F(i, v) = F(i-1, 0) + 1
   - Nếu v – r > 0 và F(i-1, v - r)>0: F(i, v) = F(i-1, v - r) + 1
   - Nếu v – r < 0 và F(i-1, v-r+k)>0: F(i, v) = F(i-1, v-r+k) + 1

     thay (v - r) mod k = (v – r + k) mod k

- Bài toán nhỏ nhất ứng với i = 1:
  - F(1,v)= 0 nếu r <> v
  - F(1,v)= 1 nếu r = v

Công thức đệ quy

Gọi r = A[i] mod k

- Với i = 1:
  - F(1, v) = 0 nếu r <> v
  - F(1, v) = 1 nếu r = v
- Với i > 1 :
  - Nếu v = r thì : F(i, v) = Max {F(i-1, v), F(i-1, 0) +1 }
  - Nếu v > r và F(i-1, v- r) >0 thì

    F(i, v) = Max {F(i-1, v), F(i-1, v – r) +1 }
  - Nếu v < r và F(i-1, v – r + k)>0 thì

    F(i, v) = Max{F(i-1, v), F(i-1, v – r + k) + 1}

Tinh chế Công thức đệ quy

Gọi r = A[i] mod k

- Với i = 1:
  - F(1, v) = 0 nếu r <> v
  - F(1, v) = 1 nếu r = v
- Với i > 1 :
  - Nếu v = r thì : F(i, v) = Max {F(i-1, v), F(i-1, 0) +1 }
  - Nếu v<>r và F(i-1, (v-r+k) mod k) > 0 thì

    F(i, v) = Max {F(i-1, v), F(i-1, (v – r + k) mod k) +1 }

29

Xây dựng bảng phương án:

- **Cấu trúc bảng phương án:**
  - Dùng mảng F[1..n][0..K-1] chứa giá trị của các F(i, v)
- **Cách tính giá trị trên bảng phương án:**
  - Điền giá trị dòng 1: Nếu A[1] mod k = v thì F[1, v] = 1 ngược lại F[1, v] = 0
  - Sử dụng công thức đệ quy và giá trị trên dòng i -1 để tính dòng i

30

Ví dụ bảng phương án:

- Với i > 1 :
  - Nếu v = r thì : F(i, v) = Max {F(i-1, v), F(i-1, 0) +1 }
  - Nếu v<>r và F(i-1, (v-r+k) mod k) > 0 thì

    F(i, v) = Max {F(i-1, v), F(i-1, (v – r + k) mod k) +1 }

i	r=A[i] v	0	1	2	3	4
1	1	0	1	0	0	0
2	1	4 | 0	0 | 1	1 | 2	2 | 0	3 | 0
3	2	3 | 0	4 | 1	0 | 2	1 | 2	2 | 3
4	2	3 | 3	4 | 3	0 | 2	1 | 2	2 | 3
5	0	0 | 4	1 | 4	2 | 3	3 | 3	4 | 4
6	3	2 | 4	3 | 4	4 | 5	1 | 5	2 | 6

31

Thuật toán tạo bảng phương án

```
void TaoBangPhuongAn(F[1..n][0..k-1])
{
    for (v=0; v <= k-1; v++)  F[1, v] = (A[i]%k == v) ? 1 : 0;
    for (i = 2; i <= n; i++)
        for (v=0; v <= k-1; v++)
        {    r = A[i] % k;
             F[i, v] = F[i-1, v];
             If (v == r  && F[i, v] <= F[i-1, 0 ])
                  F[i, v] = F[i-1, 0] + 1;
             else if (F[i-1, (v - r + k)%k] >0 && F[i, v] <= F[i-1, (v - r + k)%k])
                  F[i, v] = F[ i -1, (v - r + k)%k] + 1;
        }
}
```

32

Truy vết

Bắt đầu từ ô F[n, 0] trên dòng n ta dò ngược về dòng 0 theo nguyên tắc:

- Nếu F[i–1, (v-r+k)%k] > 0 và F[i, v] > F[i–1, (v-r+k)%k]:
  - A[i] được chọn
  - Truy tiếp ô F[i -1, (v-r+k)%k].
- Ngược lại thì A[i] không được chọn, ta truy tiếp ô F[i -1, v].

i	r=A[i] v	0	1	2	3	4
1	1	0	1	0	0	0
2	1	4 | 0	0 | 1	1 | 2	2 | 0	3 | 0
3	2	3 | 0	4 | 1	0 | 2	1 | 2	2 | 3
4	2	3 | 3	4 | 3	0 | 2	1 | 2	2 | 3
5	0	0 | 4	1 | 4	2 | 3	3 | 3	4 | 4
6	3	2 | 4	3 | 4	4 | 5	1 | 5	2 | 6

33

Tìm dãy con không giảm dài nhất

Cho một dãy gồm n số nguyên. Hãy loại bỏ khỏi dãy một số phần tử để được một dãy con không giảm dài nhất. In ra dãy con đó.

Ví dụ: với n = 10 và dãy A được cho trong bảng.

Hãy chỉ ra dãy con không giảm dài nhất ?

I	1	2	3	4	5	6	7	8	9	10
Dãy A	2	6	-7	5	8	1	-3	5	15	4

34

Xác định tham số thể hiện kích thước bài toán

- Gọi L(n) là độ dài dãy con không giảm dài nhất trong miền [1..n]
  - L(n) = L(n-1) nếu dãy con dài nhất trong miền [1..n-1] không có A[n]
  - L(n) = L(n-1) +1 nếu dãy con dài nhất trong miền [1..n-1] có A[n]
- Do đó tham số thể hiện kích thước bài toán là số phần tử n.
- Xét 2 cách ghi nhận kết quả các bài toán con:

I	1	2	3	4	5	6	7	8	9	10
Dãy A	2	6	-7	5	8	1	-3	5	15	4
Cách 1	1	2	2	2	3	3	3	3	4	4
Cách 2	1	2	1	2	3	2	2	3	4	3

35

Lập công thức đệ qui

Gọi L(i) là độ dài dãy con dài nhất trong dãy A[1..i] và có A[i] là phần tử cuối dãy con dài nhất đó.

- Nếu A[i] < A[j] với mọi j < i thì: L(i) = 1
- Ngược lại thì : L(i) = Max{ L(j) : j < i và A[j] <= A[i] } + 1
- Bài toán nhỏ nhất với đoạn A[1..1] thì L(1) = 1

I	1	2	3	4	5	6	7	8	9	10
Dãy A[i]	2	6	-7	5	8	1	-3	5	15	4
L(i)	1	2	1	2	3	2	2	3	4	3

36

Xây dựng bảng phương án:

- Mảng L[1..n]: L[i] chứa giá trị của L(i)
- Mảng Truoc[1..n]: Truoc[i] ghi chỉ số phần tử kề trước i trong dãy con dài nhất mà i là phần tử cuối dãy.

- Cách tính giá trị trên bảng phương án:
  - Gán L[1] = 1 và Truoc[1] = 0
  - Với các phần tử i từ 2 đến n:
    - Nếu A[i] < A[j] với mọi j < i thì: L(i) = 1 và Truoc(i) = 0
    - Ngược lại thì
      - L(i) = Max{ L(j) : j < i và A[j] <= A[i] } + 1
      - Truoc[i] = k sao cho L(k)= Max{ L(j) : j < i và A[j] <= A[i] }

I	1	2	3	4	5	6	7	8	9	10
Dãy A[i]	2	6	-7	5	8	1	-3	5	15	4
B[i]	1	2	1	2	3	2	2	3	4	3
Truoc[i]	0	1	0	3	4	3	3	7	8	7

Thuật toán tạo bảng phương án

```
void TaoBangPhuongAn()
{
    L[1] = 1;
    Truoc[1] = 0;
    for (i = 2; i <= n; i++)
    {   L[i] = 1;
        Truoc[i] = 0;
        for (j = i-1; j >= 1; j--)
            if (A[j] <= A[i] && L[j] >= L[i])
            {   L[i] = L[j] + 1;
                Truoc[i] = j;
            }
    }
}
```

I	1	2	3	4	5	6	7	8	9	10
A[i]	2	6	-7	5	8	1	-3	5	15	4
L[i]	1	2	1	2	3	2	2	3	4	3
Truoc[i]	0	1	0	3	4	3	3	7	8	7

Truy vết tìm lại các phần tử trên dãy con

- Bắt đầu từ phần tử i có giá trị L[i] có lớn nhất là phần tử cuối cùng trong dãy con dài nhất.
- Truy tiếp sang phần tử có chỉ số Trước[i] cho đến khi phần tử Truoc[i] = 0.

I	1	2	3	4	5	6	7	8	9	10
Dãy A[i]	2	6	-7	5	8	1	-3	5	15	4
L[i]	1	2	1	2	3	2	2	3	4	3
Truoc[i]	0	1	0	3	4	3	3	7	8	7

Thuật toán truy vết tìm lại các phần tử trên dãy con tối ưu

I	1	2	3	4	5	6	7	8	9	10
A[i]	2	6	-7	5	8	1	-3	5	15	4
B[i]	1	2	1	2	3	2	2	3	4	3
Truoc[i]	0	1	0	3	4	3	3	7	8	7

```
void TruyVet()
{
    i = n;
    for ( j = n-1; j >=1; j--)
        if (B[j] > B[i]) i = j;
    while ( i > 0)
    {
        <in thông tin A[i] thuộc dãy con tối ưu>;
        i = Truoc[i];
    }
    <in thông tin A[i] thuộc dãy con tối ưu>;
}
```

Lập lịch thuê nhân công

Có một dự án kéo dài trong T tháng, người quản lý cần phải lập lịch sử dụng công nhân mỗi tháng cho dự án. Biết rằng, số công nhân tối thiểu cần trong tháng thứ i là Scn[i]; tiền dịch vụ khi thuê 1 công nhân mới là DV; tiền đền bù khi sa thải một công nhân là ST; lương tháng mỗi công nhân phải trả là LT.

Cần phải thuê hay sa thải bao nhiêu công nhân mỗi tháng để tổng chi phí nhân công của dự án là nhỏ nhất.

Giả thiết	Kết luận
T = 3 DV=4; ST= 5; LT=6 Scn={11; 9; 10}	265 11 10 10

41

Xác định tham số thể hiện kích thước bài toán

- Tham số thể hiện kích thước bài toán là số tháng T
- Tổng chi phí nhân công trong T tháng được tính từ tổng chi phí nhân công của T-1 tháng cộng thêm chi phí trả nhân công của tháng thứ T.
- Chi phí trả nhân công của tháng thứ T bao gồm :
  - Tiền lương trả cho số nhân công của tháng T và
  - Tiền dịch vụ nếu số nhân công của tháng T lớn hơn số nhân công tháng T-1 hay tiền sa thải nếu số nhân công trong tháng T nhỏ hơn số nhân công của tháng T-1.
- Kích thước bài toán phụ thuộc vào 2 tham số: số tháng và số nhân công của tháng

42

Lập công thức đệ qui

- Scn[i] lưu số công nhân cần thuê cho tháng thứ i
- Smax là số công nhân của tháng cần nhiều người nhất
- Bài toán con nhỏ nhất ứng với i = 1 (tháng đầu tiên):

  C(1, j) = j * (DV + LT) với j = Scn[1]..Smax
- C(i, j) là chi phí tối thiểu của i tháng đầu tiên nếu tại tháng thứ i có j công nhân được thuê.

  C(i, j) = Min{ C(i-1, k) + chi phí để từ k người thành j người }

  (i=2..T; j =Scn[i]..Smax; k = Scn[i-1]..Smax)
- Kết quả bài toán là: Kq = Min{C(T, j) + chi phí sa thải j người}
  j=Scn[T]..Smax

43

Xây dựng bảng chứa C(i, j)

- Mảng C[1..T+1, 1..Smax]: C[i, j] ghi nhận giá trị C(i, j)

C(1, j) = j * (DV + LT) với j = Scn[1]..Smax

C(i, j) = Min{ C(i-1, k) + chi phí để từ k người thành j người }

(i=1..T; j =Scn[i]..Smax; k = Scn[i-1]..Smax)

C(T+1, j) = C(T, j) + (j * ST) j=Scn[T]..Smax

Scn	i \ j	9	10	11
11	1			99
9	2	99+45+12=156	99+50+6=155	99+55=164
10	3		155+50=205	155+55+4=214
	4		205+60=265	2144+66=280

44

Xây dựng bảng truy vết số công nhân

- Mảng Truoc[1..T, 1..Smax]: Truoc[i, j] := k là số người thuê ở tháng thứ i-1 để có C[i, j]

Scn	i \ j	9	10	11
11	1			99
9	2	99+45+12=156	99+50+6=155	99+55=164
10	3		155+50=205	155+55+4=214
	4		205+60=265	2144+66=280

Scn	i \ j	9	10	11
11	1			
9	2	11	11	11
10	3		10	10

45

Thuật toán tạo bảng phương án C và Truoc

```
{   for (j=Scn[1]; j <= Smax; j++)  C[1, j] = j * (DV + LT);
    for (i = 2; i <= T; i++)
        for (j=Scn[i]; j <= Smax; j++)
        {   C[i, j] = MAXINT;
            for (k=Scn[i-1]; k <= Smax; k++)
            {   X = C[i-1, k] ;
                if (k > j) X = X + (k – j)*DV;  else X = X + (j – k)*ST
                if (X< C[i, j]) { C[i, j] = X; Truoc[i, j] = k; }
            }
        }
    for (j=Scn[T]; j <= Smax; j++)  C[T+1, j] = C(T, j) + (j * ST) ;
}
```

46

Thuật toán truy vết số công nhân mỗi tháng

```
{   //Tìm số nhân công của tháng thứ T
    S = C[T+1, Scn[T] ];        k = Scn[T];
    for (j=Scn[T]+1; j <= Smax; j++)
        if (S > C[T+1, j ]) { k = j; S =C[T+1, j ]; }
    <Tháng T cần k công nhân>
    for (i=T; i > 1; i--)
    {   k = Truoc[i, k];
        <Tháng i-1 cần k công nhân>
    }
}
```

47

Bài tập

1. Có N gói kẹo, gói thứ i có Ai cái kẹo. Không được bóc bất kỳ một gói kẹo nào, cần chia N gói kẹo thành hai phần sao cho độ chênh lệch số kẹo giữa hai gói là ít nhất.
2. Cho n loại tờ giấy bạc. Tờ giấy bạc thứ i có mệnh giá A[i]. Số tờ mỗi loại không giới hạn. Cần chi trả cho khách hàng số tiền M đồng. Hãy cho biết mỗi loại tiền cần bao nhiêu tờ sao cho tổng số tờ là ít nhất. Nếu không đổi được, thì thông báo "KHONG DOI DUOC".
3. Cho n loại tờ giấy bạc. Tờ giấy bạc thứ i có mệnh giá A[i]. Giả thiết loại tiền mệnh giá A[i] có B[i] tờ (i := 1, n). Cần chi trả cho khách hàng số tiền M đồng. Hãy cho biết mỗi loại tiền cần bao nhiêu tờ sao cho tổng số tờ là ít nhất. Nếu không đổi được, thì thông báo "KHONG DOI DUOC".
4. Cần cắm k loại hoa khác nhau vào n lọ xếp thẳng hàng sao cho loại hoa có số hiệu nhỏ được đặt trước hoa có số hiệu lớn. Với mỗi loại hoa i ta biết giá trị thẩm mỹ khi cắm hoa đó vào lọ j là v[i,j]. Hãy tìm phương án cắm các loại hoa trên vào n lọ sao cho tổng giá trị thẩm mỹ là lớn nhất.
5. Một hộp thư điện tử cho phép gửi đính kèm một hoặc nhiều file vào một thư điện tử sao cho tổng dung lượng các file đính kèm trên thư điện tử không vượt quá kích thước M KByte cho trước. Để có số thư điện tử gửi đi là ít nhất, người ta cần chọn trong N file dữ liệu các file để đính kèm vào một email sao cho tổng dung lượng của các file đính kèm là lớn nhất nhưng không vượt quá kích thước M. Giả sử, M $\leq$ 100; N $\leq$ 50 và file thứ i trong N file dữ liệu có kích thước là Ai KByte (là số nguyên dương). Hãy trình bày thuật toán để chọn trong N file dữ liệu các file đính kèm vào một thư điện tử theo yêu cầu trên, liệt kê kích thước các file đã chọn.

48

6. Việc xây dựng công trình cấp nước sạch ở các bản vùng cao rất khó khăn, phải xây dựng kéo hệ thống dẫn nước từ dưới thung lũng lên bản băng qua các ngọn đồi. Để thuận lợi cho việc dẫn nước, người ta đặt các trạm bơm nối tiếp trên từng ngọn đồi dẫn từ thung lũng lên bản vùng cao, theo nguyên tắc ngọn đồi sau phải cao hơn ngọn đồi trước, hai trạm bơm kề nhau không nhất thiết đặt trên hai ngọn đồi kề nhau. Giả sử có N ngọn đồi ($N \leq 100$), mỗi ngọn đồi được gán một số thứ tự tăng theo hướng từ thung lũng lên vùng cao, ngọn đồi thứ i có độ cao Ai (là một số nguyên dương).
   1. Hãy trình bày giải thuật chọn lựa ra các ngọn đồi đặt các trạm bơm sao cho số ngọn đồi được chọn là nhiều nhất.
   2. Sử dụng Pascal, C/C++, hoặc Java cài đặt giải thuật trên.
7. Một công ty máy tính nhận được N hợp đồng ($N \leq 50$) lắp đặt hệ thống máy tính tại N công ty. Hợp đồng thứ i có giá trị là Ci (số nguyên) và cần Ai nhân sự để thực hiện. Do số lượng nhân sự của công ty có hạn, nên Công ty muốn ưu tiên chọn một số hợp đồng để thực hiện trước sao cho tổng giá trị của các hợp đồng đã chọn là lớn nhất nhưng tổng số nhân sự thực hiện các hợp đồng đó không vượt quá số lượng M nhân sự ($M \leq 100$) hiện có của công ty. Hãy trình bày thuật giải để chọn lựa các hợp đồng theo yêu cầu trên, cho biết tổng giá trị của các hợp đồng đã chọn; giá trị hợp đồng và số lượng nhân sự của các hợp đồng đã chọn.

6. Một công ty máy tính nhận được N hợp đồng ($N \leq 50$) lắp đặt hệ thống máy tính tại N công ty. Hợp đồng thứ i có giá trị là Ci (số nguyên) và cần Ai nhân sự để thực hiện. Do số lượng nhân sự của công ty có hạn, nên Công ty muốn ưu tiên chọn một số hợp đồng để thực hiện trước sao cho tổng giá trị của các hợp đồng đã chọn là lớn nhất nhưng tổng số nhân sự thực hiện các hợp đồng đó không vượt quá số lượng M nhân sự ($M \leq 100$) hiện có của công ty. Hãy trình bày thuật giải để chọn lựa các hợp đồng theo yêu cầu trên, cho biết tổng giá trị của các hợp đồng đã chọn; giá trị hợp đồng và số lượng nhân sự của các hợp đồng đã chọn.