Cấu trúc dữ liệu và giải thuật

Một số mô hình thuật toán kinh điển

Nguyễn Văn Tiến

Nội dung

1. ~~Mô hình thuật toán sinh (Generative Algorithm)~~
2. ~~Mô hình thuật toán đệ quy (Recursion Algorithm)~~
3. ~~Mô hình thuật toán quay lui (Backtrack Algorithm)~~
4. ~~Mô hình thuật toán tham lam (Greedy Algorithm)~~
5. ~~Mô hình thuật toán chia và trị (Divide and Conquer Algorithm)~~
6. ~~Mô hình thuật toán nhánh cận (Branch and Bound Algorithm)~~
7. Mô hình thuật toán quy hoạch động (Dynamic programming)

Mô tả thuật toán Quy hoạch động (1)

Phương pháp quy hoạch động dùng để giải lớp các bài toán thỏa mãn những điều kiện sau:

1. Bài toán lớn cần giải có thể phân rã được thành nhiều bài toán con.

2. Sự phối hợp lời giải của các bài toán con cho ta lời giải của bài toán lớn:

- Bài toán con có lời giải đơn giản được gọi là cơ sở của quy hoạch động.

- Công thức phối hợp nghiệm của các bài toán con gọi là công thức truy hồi

Mô tả thuật toán Quy hoạch động (2)

Phương pháp quy hoạch động dùng để giải lớp các bài toán thỏa mãn những điều kiện sau:

3. Có không gian vật lý lưu trữ lời giải các bài toán con (Bảng phương án của quy hoạch động).

4. Quá trình giải quyết từ bài toán cơ sở (bài toán con) để tìm ra lời giải bài toán lớn phải được thực hiện sau hữu hạn bước dựa trên bảng phương án của quy hoạch động.

Quy hoạch động (Dynamic programming)

So sánh chia để trị và quy hoạch động

Trong giải thuật chia và trị:

Các bài toán con độc lập, sau đó các bài toán con này được giải một cách đệ quy.

Trong giải thuật quy hoạch động:

Các bài toán con là không độc lập với nhau, mà cùng có chung các bài toán con nhỏ hơn.

Các yếu tố của quy hoạch động

Cơ sở của quy hoạch động:

Những trường hợp đơn giản có thể tính trực tiếp

Cấu trúc con tối ưu:

Phương pháp chia nhỏ các bài toán cho đến khi gặp được bài toán cơ sở.

Tổng hợp:

Hệ thức truy hồi tính giá trị tối ưu của hàm mục tiêu của bài toán lớn qua giá trị tối ưu của các bài toán con thành phần.

Đánh giá giải thuật quy hoạch động

Ưu điểm:

Độ phức tạp tốt.

Nhược điểm:

Chỉ áp dụng cho các bài toán tối ưu nhưng không cần biết đầy đủ các phương án tối ưu, chỉ quan tâm đến kết quả tối ưu

Bảng phương án sẽ tốn bộ nhớ

Quy hoạch động (Dynamic programming)

Ví dụ: Tính số Fibonacci thứ N

Định nghĩa số Fibonacci F(n):

F(0)=0

F(1)=1

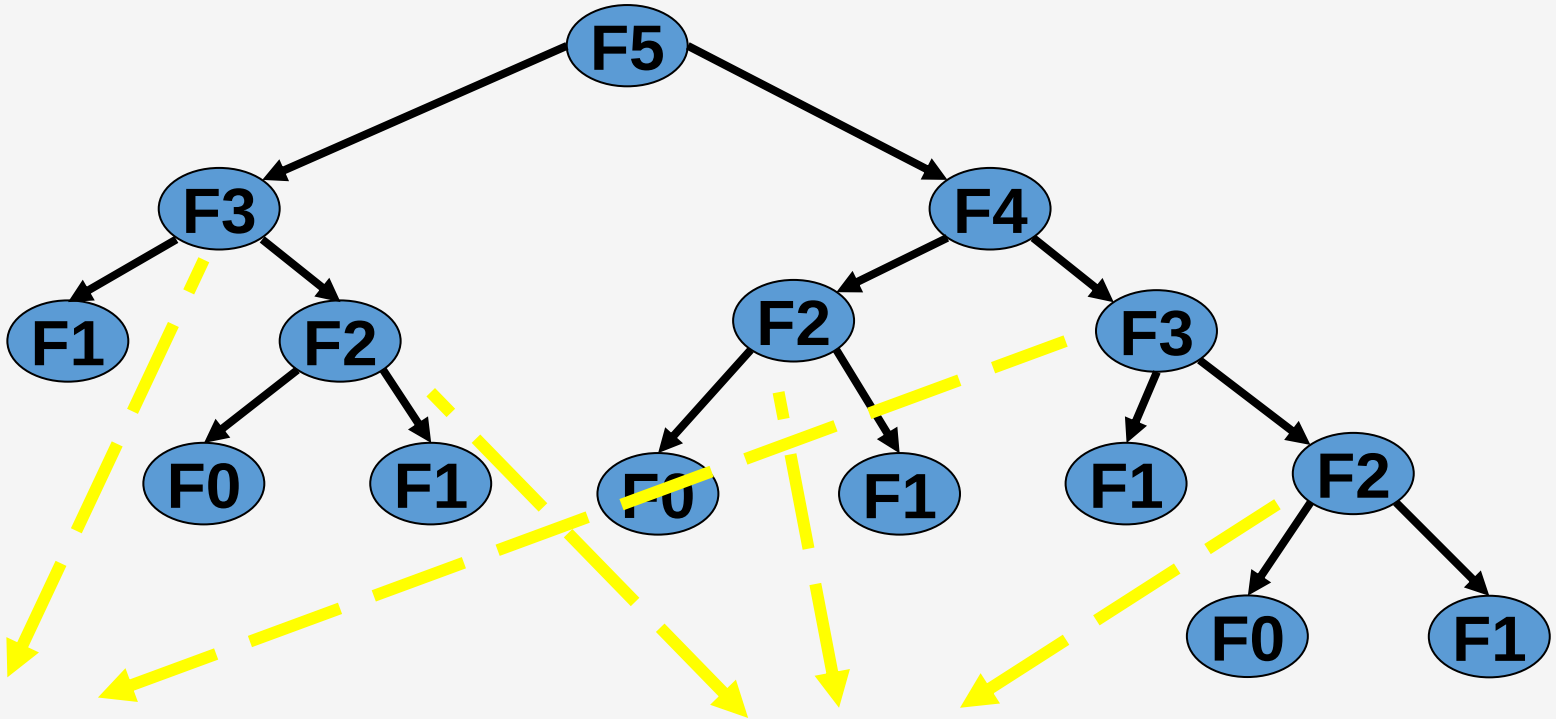
F(n)=F(n-2)+F(n-1) với n>1

Ví dụ:

F(2)=1, F(3)= 2, F(4) = 3 , F(5)=5, F(6)=8

Quy hoạch động (Dynamic programming)

Ví dụ: Tính số Fibonacci thứ N



2 lần tính F3

3 lần tính F2

Quy hoạch động (Dynamic programming)

Ví dụ: Tính số Fibonacci thứ N

Bước cơ sở: Tính F[0] =1, F[1] =1.

Công thức truy hồi: Lưu lời giải các bài toán con biết trước vào bảng phương án (ở đây là mảng một chiều F[100]). Sử dụng lời giải của bài toán con trước để tìm lời giải của bài toán con tiếp theo.

Trong bài toán này là F[i] = F[i-1] + F[i-2] với $n \geq 2$.

Bằng cách lưu trữ vào bảng phương án, ta chỉ cần giải mỗi bài toán con một lần.

Truy vết: Đưa ra nghiệm của bài toán bằng việc truy lại dấu vết phối hợp lời giải các bài toán con để có được nghiệm của bài toán lớn.

Quy hoạch động (Dynamic programming)

Ví dụ: Tính số Fibonacci thứ N

Procedure Fibonacci(N)

F[0] = 1

F[1] = 1

For i:=2 to N do

F[i]: = F[i-1] + F[i-2]

Độ phức tạp: O(N)

Quy hoạch động (Dynamic programming)

Các bài toán áp dụng quy hoạch động

1. Bài toán cái túi
2. Dãy con chung dài nhất
3. Dãy con liên tiếp có tổng lớn nhất
4. Dãy con tăng dài nhất
5. Dãy con có tổng bằng S
6. Xâu con đối xứng dài nhất
7. Tính tổ hợp

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Có N gói đồ vật, gói thứ i có khối lượng là w[i], có giá trị là v[i]

Cái túi chỉ có thể mang được khối lượng tối đa là W.

Xét trường hợp mỗi đồ vật chỉ có thể lấy nguyên vẹn hoặc không lấy.

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Ví dụ có 3 đồ vật:

+ Guitar: Giá trị $1500, Trọng lượng 1kg
+ Radio: Giá trị $3000, Trọng lượng 4kg
+ Laptop: Giá trị $2000, Trọng lượng 3kg

Túi có thể chứa được 4kg

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Thuật toán đơn giản (Duyệt vét cạn):

[Guitar, Radio, Laptop]

STT	Phương án	Trọng lượng	Giá trị	Max
1	Không chọn (000)	0kg	$0	$0
2	Laptop (001)	3kg	$2000	$2000
3	Radio (010)	4kg	$3000	$3000
4	Laptop + Radio (011)	~~7kg~~		
5	Guitar (100)	1kg	$1500	$3000
6	Guitar + Laptop (101)	4kg	$3500	**$3500**
7	Guitar + Radio (110)	~~5kg~~		
8	Guitar + Radio + Laptop (111)	~~8kg~~		

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

- 3 vật phẩm => 8 lần duyệt
- 4 vật phẩm => 16 lần duyệt

Số tổ hợp cần duyệt: 2^n Độ phức tạp tính toán: $O(2^n)$

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Thuật toán quy hoạch động

- Xây dựng bảng phương án

C	1kg	2kg	3kg	4kg
Guitar				
Radio				
Laptop				

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Thuật toán quy hoạch động

- Xây dựng bảng phương án

C	1kg	2kg	3kg	4kg
Guitar	$1500			
Radio				
Laptop				

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Thuật toán quy hoạch động

- Xây dựng bảng phương án

C	1kg	2kg	3kg	4kg
Guitar	$1500	$1500	$1500	$1500
Radio				
Laptop				

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Thuật toán quy hoạch động

- Xây dựng bảng phương án

C	1kg	2kg	3kg	4kg
Guitar	$1500	$1500	$1500	$1500
Radio	$1500	$1500	$1500	
Laptop				

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Thuật toán quy hoạch động

- Xây dựng bảng phương án

C	1kg	2kg	3kg	4kg
Guitar	$1500	$1500	$1500	$1500
Radio	$1500	$1500	$1500	$3000
Laptop				

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Thuật toán quy hoạch động

- Xây dựng bảng phương án

C	1kg	2kg	3kg	4kg
Guitar	$1500	$1500	$1500	$1500
Radio	$1500	$1500	$1500	$3000
Laptop	$1500	$1500	$2000	

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Thuật toán quy hoạch động

- Xây dựng bảng phương án

C	1kg	2kg	3kg	4kg
Guitar	$1500	$1500	$1500	$1500
Radio	$1500	$1500	$1500	$3000
Laptop	$1500	$1500	$2000	**$3500**

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Thuật toán quy hoạch động

- Xây dựng bảng phương án

C	1kg	2kg	3kg	4kg
Guitar	$1500	$1500	$1500	$1500
Radio	$1500	$1500	$1500	$3000
Laptop	$1500	$1500	$2000	**$3500**

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

Công thức cho các ô trong ma trận C[N][W]

C[i][j] = max(C[i-1][j], giá trị của đồ vật hiện tại + giá trị của chỗ trống còn lại) = max (C[i-1][j], v_i + C[i-1][j-w_i])

C	1kg	2kg	3kg	4kg
Guitar	$1500	$1500	$1500	$1500
Radio	$1500	$1500	$1500	$3000
Laptop	$1500	$1500	$2000	**$3500**

Điều gì xảy ra khi thêm vật phẩm iPhone 1kg, giá $2000 vào túi?

Quy hoạch động (Dynamic programming)

1. Bài toán cái túi

{Khởi tạo}: For j = 0 to W do C[0,j] :=0 ;

{Lặp:} For i = 1 to N do

  For j = 1 to W do

    Begin

        C[i,j] := max(C[i-1, j], v[i] + C[i-1,j-w[i]]);

End;

Return C(N, W)

2. Bài toán dãy con chung dài nhất

Cho hai dãy $X = (x_1,x_2,...,x_m)$ và $Y = (y_1,y_2,...,y_n)$ gồm các ký tự.

Cần tìm dãy con chung dài nhất của hai dãy X và Y.

Ví dụ: Khi truy cập một trang từ điển Dictionary.com và tìm kiếm từ khóa "FISH". Do vô tình gõ sai chính tả từ khóa thành "HISH". Từ điển vẫn có thể trả về kết quả gợi ý: "Fish" hoặc "Vista".

Nếu sử dụng quy hoạch động thì giá trị các ô là gì?

Các bài toán nhỏ được chia như thế nào?

Các chiều trong bảng phương án là gì?

2. Bài toán dãy con chung dài nhất

Xây dựng bảng phương án

	H	I	S	H
F				
I				
S				
H				

2. Bài toán dãy con chung dài nhất

Xây dựng bảng phương án

	H	I	S	H
F	0	0		
I				
S			2	0
H				**3**

2. Bài toán dãy con chung dài nhất

Xây dựng bảng phương án

	H	I	S	H
F	0	0	0	0
I	0	1	0	0
S	0	0	2	0
H	0	0	0	**3**

Công thức: C[i][j] = C[i-1][j-1] + 1 Nếu WordA[i] == WordB[j]

= 0 Nếu ngược lại

2. Bài toán dãy con chung dài nhất

Áp dụng cho từ HISH và VISTA

	V	I	S	T	A
H	0	0	0	0	0
I	0	1	0	0	0
S	0	0	**2**	0	0
H	0	0	0	0	0

Giá trị cần tìm là giá trị lớn nhất ghi nhận được, không phải giá trị cuối cùng

2. Bài toán dãy con chung dài nhất

Nếu từ khóa tìm kiếm được gõ nhầm thành "FOSH" thì từ cần tìm là FISH hay FORT

	F	O	S	H
F	1	0	0	0
O	0	**2**	0	0
R	0	0	0	0
T	0	0	0	0

	F	O	S	H
F	1	0	0	0
I	0	0	0	0
S	0	0	1	0
H	0	0	0	**2**

Mặc dù độ dài xâu con chung liên tiếp cùng là 2 nhưng có thể nhận thấy FISH liên quan hơn so với FORT

2. Bài toán dãy con chung dài nhất

Xây dựng bảng phương án xét xâu tuần tự nhưng không liên tiếp

	F	O	S	H
F	1	1		
I	1			
S		1	2	2
H				

2. Bài toán dãy con chung dài nhất

	F	O	S	H
F	1	1	1	1
O	1	2	2	2
R	1	2	2	2
T	1	2	2	**2**

	F	O	S	H
F	1	1	1	1
I	1	1	1	1
S	1	1	2	2
H	1	1	2	**3**

Công thức: C[i][j] = C[i-1][j-1] + 1, nếu WordA[i] == WordB[j]

C[i][j] = max(C[i-1][j], C[i][j-1]), nếu ngược lại

3. Bài toán dãy con liên tiếp tổng lớn nhất

Cho dãy A dưới dạng mảng A[1 … n] các số nguyên bao gồm cả số âm và số dương.

Hãy tìm dãy con các phần tử liên tiếp của dãy A có tổng lớn nhất

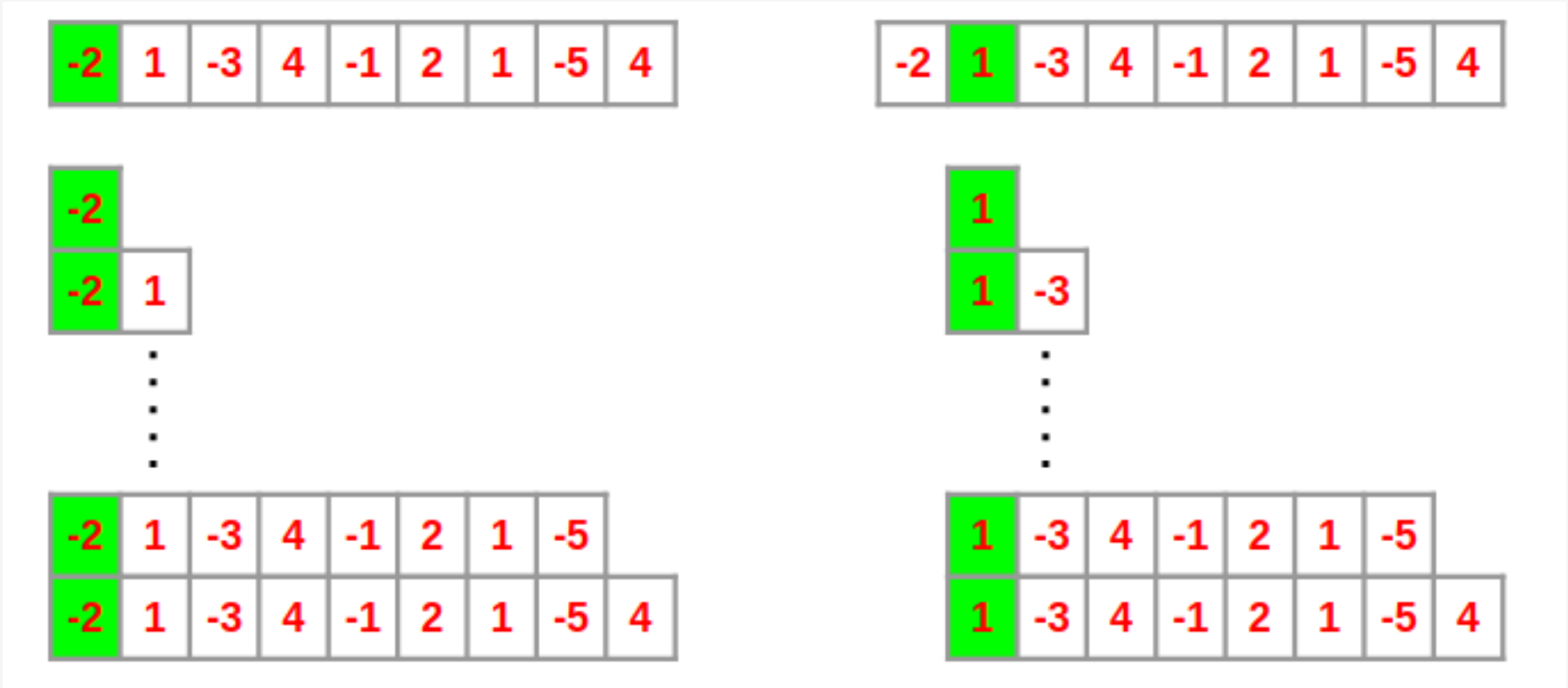
Kết quả: In ra tổng lớn nhất và dãy con tương ứng.

Ví dụ

-2	1	-3	4	-1	2	1	-5	4

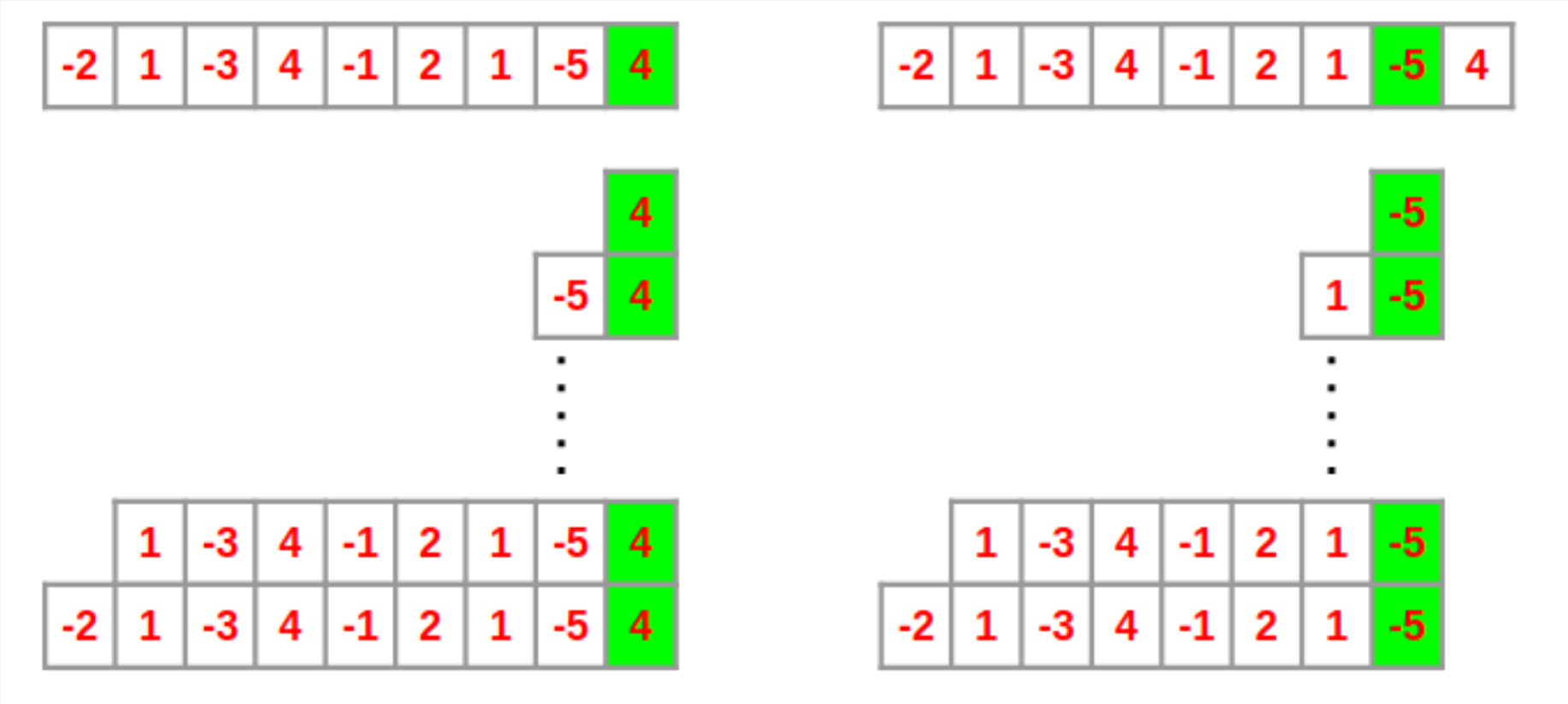
3. Bài toán dãy con liên tiếp tổng lớn nhất

Duyệt vét cạn



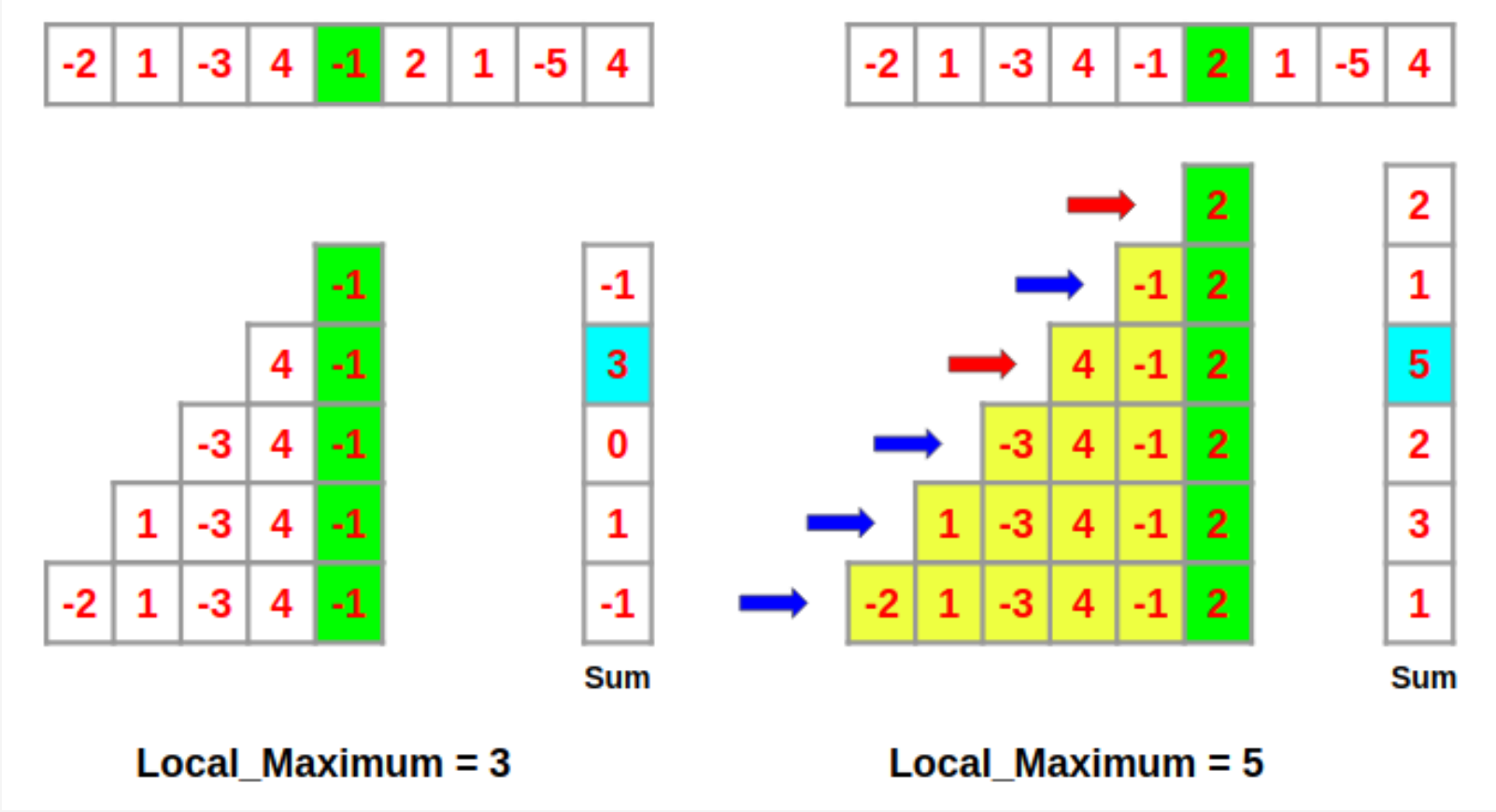
3. Bài toán dãy con liên tiếp tổng lớn nhất

Duyệt vét cạn



3. Bài toán dãy con liên tiếp tổng lớn nhất

Duyệt vét cạn



3. Bài toán dãy con liên tiếp tổng lớn nhất

Bảng phương án

-2	1	-2	4	3	5	**6**	1	5

Công thức: Local_maximum[i] = Max(A[i], A[i] + Local_maximum[i-1])

4. Bài toán dãy con tăng dài nhất

Cho dãy số A có N phần tử, bài toán yêu cầu tìm dãy con dài nhất của dãy A sao cho phần tử sau của dãy con luôn lớn hơn phần tử trước. Dãy con của một dãy số là dãy có được sau khi loại bớt một số phần tử, các phần tử khác giữ nguyên vị trí.

Gọi F(i) là độ dài dãy con tăng dài nhất kết thúc ở A(i), ta có công thức tính:

$F(1)=1$

$F(i)=\max\{F(i), F(j)+1\}$ Với i,j thỏa mãn $1 \le j < i$ và $A(j) < A(i)$

Kết quả bài toán là **max{F}**

4. Bài toán dãy con tăng dài nhất

Ví dụ: 10, 22, 9, 33, 21, 50, 41, 60

Bảng phương án

1	2	1	3	2	4	4	**5**

5. Bài toán dãy con có tổng bằng S

Bài toán:

Cho dãy $A_1,A_2,...,A_N$. Xác định có hay không một dãy con của dãy đó có tổng bằng S.

Giải pháp:

Sử dụng bảng phương án C là một ma trận nhị phân

Đặt C[i,j]=1 nếu có thể tạo ra tổng j từ một dãy con của dãy gồm các phần tử A1,A2,...,Ai. Với 0 <= j <= S

Ngược lại thì C[i,j]=0. Nếu C[n,S]=1 thì đáp án của bài toán trên là "có".

5. Bài toán dãy con có tổng bằng S

Ta có thể tính C[i,j] theo công thức:

C[i,j]=1 nếu C[i−1,j]=1 hoặc C[i−1,j−a[i]]=1.

Với bài toán con cơ sở C[i][0] = 1

5. Bài toán dãy con có tổng bằng S

Ví dụ A = 1, 2, 3, 4, 5 S = 6

	1	2	3	4	5	6
1	1	0	0	0	0	0
2	1	1	1	0	0	0
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	1	1	1	1	1	1

C[i,j]=1 nếu C[i−1,j]=1 hoặc C[i−1,j−a[i]]=1

5. Bài toán dãy con có tổng bằng S

Ta có thể tính C[i,j] theo công thức:

C[i,j]=1 nếu C[i−1,j]=1 hoặc C[i−1,j−a[i]]=1.

Với bài toán con cơ sở C[i][0] = 1

Nhận xét:

Để tính dòng thứ i, ta chỉ cần dòng i−1.

Bảng phương án khi đó chỉ cần 1 mảng 1 chiều C[0..S]

5. Bài toán dãy con có tổng bằng S

```
C[0]:=1;


for i := 1 to n do
    for j := S downto a[i] do
        if (C[j]=0) and (C[j-a[i]]=1) then
            C[t]:=1;
```

6. Bài toán xâu con đối xứng dài nhất

Bài toán: Cho một xâu S, độ dài không quá 1000 kí tự. Tìm xâu đối xứng dài nhất là xâu con của S (Xâu con là một dãy các kí tự liên tiếp).

Giải pháp: Dùng ma trận F[i, j] có ý nghĩa: F[i, j] = true/false nếu đoạn gồm các kí tự từ i đến j của S có/không là xâu đối xứng.

Công thức:

1. F[i, i] = True: xâu 1 ký tự luôn đối xứng.
2. F[i, i + 1] = True nếu $S_i = S_{i+1}$. (Xâu 2 ký tự đối xứng)
3. F[i, j] = F[i+1, j-1] nếu $S_i = S_j$.
4. F[i, j] = False nếu $S_i \neq S_j$.

6. Bài toán xâu con đối xứng dài nhất

Ví dụ xâu: abcbadd

	a	b	c	b	a	d	d
a	1	0	0	0	1	0	0
b		1	0	1	0	0	0
c			1	0	0	0	0
b				1	0	0	0
a					1	0	0
d						1	1
d							1

6. Bài toán xâu con đối xứng dài nhất

```
for i := 1 to n do
    F[i, i] := True;
for i := 1 to n - 1 do
    if s[i] == s[i+1] F[i, i+1] := True;
for k := 3 to n do
    for i := 0 to (n-k + 1) do
            j := i + k;
            F[i, j] := ( F[i+1, j-1] ) and (s[i] = s[j] );
```

Kết quả: Max(k) thỏa mãn F[i,j]=True. Độ phức tạp là $O(N^2)$.

7. Bài toán tính tổ hợp

Tính C(n,k) với n đến 1000.

Phương pháp quy hoạch động: Áp dụng công thức

$$C_n^0 = C_n^n = 1$$
$$C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$$

Chú ý: Vì giá trị sẽ rất lớn nên cần chia dư sau mỗi bước tính.

7. Bài toán tính tổ hợp

```
for (i = 0; i <= 1000; i++)    {
        for (j = 0; j <= i; j++)    {
            if (j == 0 || j == i)
                C[i][j] = 1;
            else
                C[i][j] = (C[i-1][j-1] + C[i-1][j]) % MOD;
        }
}
```