

CHIẾN LƯỢC QUI HOẠCH ĐỘNG

- Các đặc trưng cơ bản
- Các ví dụ minh họa

CÁC ĐẶC TRƯNG CƠ BẢN

- Giải một bài toán bằng cách giải các bài toán con kích thước nhỏ hơn
- Nghiệm của bài toán có được bằng cách kết hợp nghiệm của các bài toán con

CÁC ĐẶC TRƯNG CƠ BẢN

- Nghiệm (lời giải) bài toán ban đầu và các bài toán con được biểu diễn bởi một **hệ thức truy hồi**

CÁC ĐẶC TRƯNG CƠ BẢN

- Sử dụng chiến lược qui hoạch động để giải một bài toán được chia làm hai bước:
 - Mô hình hóa lời giải bài toán bằng một hệ thức truy hồi
 - Giải hệ thức truy hồi bằng một giải thuật hiệu quả (từ dưới lên)

CÁC ĐẶC TRƯNG CƠ BẢN

- Chiến lược qui hoạch động phát triển giải thuật giải bài toán bằng cách giải các bài toán con từ dưới lên (bottom up)
- Kết quả các bài toán con được lưu trữ trong một bảng và được sử dụng để giải bài toán đã cho

CÁC ĐẶC TRƯNG CƠ BẢN

- Lưu ý:
 - Nếu giải hệ thức truy hồi bằng đệ qui (top down) thì độ phức tạp có thể **rất lớn**
 - Chiến lược qui hoạch động giải bài toán từ dưới lên (bottom up) với **giải thuật hiệu quả**
 - Qui hoạch động thường được ứng dụng để giải các **bài toán tối ưu**

CÁC VÍ DỤ MINH HỌA

- Bài toán dãy các đồng xu
- Bài toán robot nhặt các đồng xu
- Bài toán cái túi
- Bài toán cây con chung dài nhất (tham khảo)

BÀI TOÁN DÃY ĐỒNG XU

- Cho một dãy n đồng xu có giá trị tương ứng là c_1, c_2, \dots, c_n , hãy nhặt một số các đồng xu sao cho tổng giá trị lớn nhất và không có 2 đồng xu nào kề nhau

BÀI TOÁN DÃY ĐỒNG XU

- Gọi $F(n)$ là lượng tiền (giá trị) lớn nhất có thể nhặt được, xét hai trường hợp
 - Nếu đồng xu cuối cùng được chọn thì giá trị lớn nhất là $c_n + F(n-2)$
 - Nếu đồng xu cuối cùng không được chọn thì giá trị lớn nhất là $F(n-1)$

BÀI TOÁN DÃY ĐỒNG XU

- Suy ra hệ thức truy hồi là

$$F(n) = \max\{c_n + F(n - 2), F(n - 1)\} \text{ với } n > 1,$$

$$F(0) = 0, F(1) = c_1$$

BÀI TOÁN DÃY ĐỒNG XU

ALGORITHM CoinRow($C[1..n]$)

1 $F[0] \leftarrow 0; F[1] \leftarrow C[1]$

2 **for** $i \leftarrow 2$ **to** n **do**

3 $F[i] \leftarrow \max(C[i] + F[i - 2], F[i - 1])$

4 **return** $F[n]$

BÀI TOÁN DÃY ĐỒNG XU

- Kích thước đầu vào là số n
- Thao tác **cơ bản** là tính max của 2 số
- Gọi thời gian thao tác cơ bản là c

$$T(n) = (n-1)c = \Theta(n)$$

BÀI TOÁN DÃY ĐỒNG XU

Ví dụ: Cho các c_i khởi đầu như trong bảng

index	0	1	2	3	4	5	6
C		5	1	2	10	6	2
F	0	5					

$$F[0] = 0, F[1] = c_1 = 5$$

BÀI TOÁN DÃY ĐỒNG XU

$F[2] = \max\{1 + 0, 5\} = 5$

index	0	1	2	3	4	5	6
C		5	1	2	10	6	2
F	0	5	5				

$F[3] = \max\{2+5, 5\} = 7$

index	0	1	2	3	4	5	6
C		5	1	2	10	6	2
F	0	5	5	7			

$$F(n) = \max\{c_n + F(n - 2), F(n - 1)\}, n = 2, 3$$

BÀI TOÁN DÃY ĐỒNG XU

$F[4] = \max\{10+5, 7\} = 15$

index	0	1	2	3	4	5	6
C		5	1	2	10	6	2
F	0	5	5	7	15		

$F[5] = \max\{6+7, 15\} = 15$

index	0	1	2	3	4	5	6
C		5	1	2	10	6	2
F	0	5	5	7	15	15	

$$F(n) = \max\{c_n + F(n-2), F(n-1)\}, n = 4, 5$$

BÀI TOÁN DÃY ĐỒNG XU

index	0	1	2	3	4	5	6
C		5	1	2	10	6	2
F	0	5	5	7	15	15	17

$$F[6] = \max\{2+15, 15\}=17$$

$$F(n)=\max\{c_n+ F(n - 2), F(n - 1)\}, n=6$$

Kết quả $F(n)=17$

BÀI TOÁN DÃY ĐỒNG XU

- Tìm các đồng xu được chọn bằng cách theo vết quay lui (back-trace) của quá trình tính toán từ $F(n)$ về $F(2)$
- Xét $c_i + F(i - 2)$ và $F(i - 1)$, khi $i = n, n-1, \dots$,
 - Nếu $c_i + F(i - 2) > F(i - 1)$ thì xu thứ i với giá trị c_i là được chọn
 - Ngược lại tìm đồng xu được chọn trong khi tính $F(i - 1)$

BÀI TOÁN DÃY ĐỒNG XU

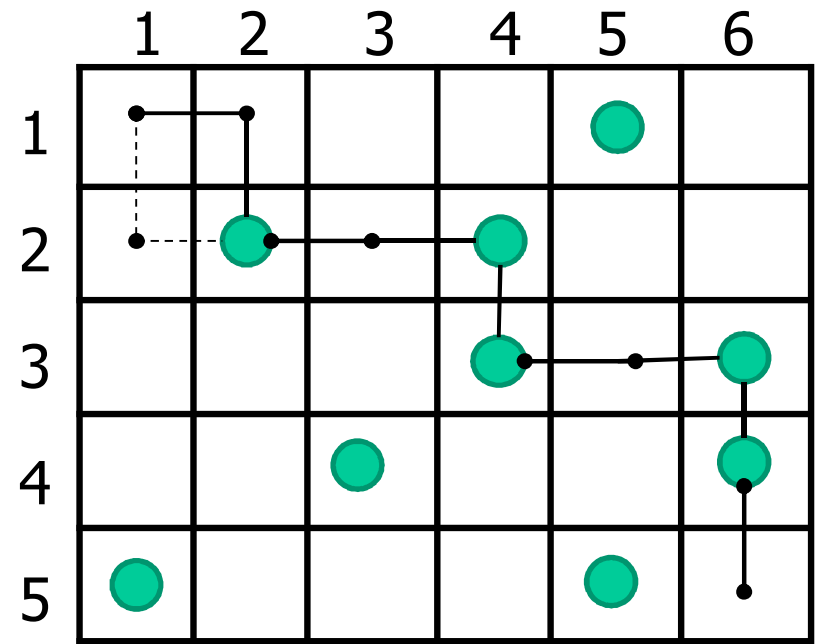
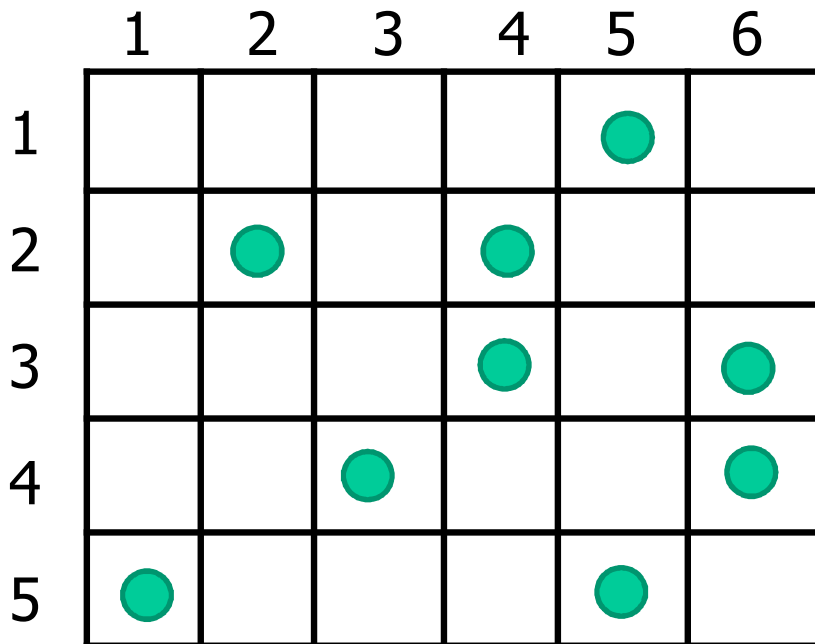
- Trong ví dụ trên do $F(6)=c_6+F(4)>F(5)$ nên xu thứ 6 với $c_6=2$ được chọn
- Đồng xu trước xu thứ 6 được chọn trong khi tính $F(4)$, do $F(4)=c_4+F(2)>F(3)$ nên xu thứ 4 với $c_4=10$ được chọn
- Đồng xu trước xu thứ 4 được chọn trong khi tính $F(2)$, do $F(2)=F(1)>c_2+F(0)$ nên xu thứ 1 với $c_1=5$ được chọn (trong $F(1)$)

$$\text{Tổng } c_1+c_4+c_6=5+10+2=17$$

BÀI TOÁN ROBOT NHẶT CÁC ĐỒNG XU

- Một số đồng xu được đặt trong một bảng kích thước $n \times m$, mỗi ô chỉ nhiều nhất một đồng xu. Một robot di chuyển từ ô $(1,1)$ đến ô (n,m) , mỗi bước di chuyển chỉ sẽ **đến ô bên phải hoặc ô phía dưới**. Khi robot đến một ô có đồng xu **nó luôn luôn nhặt đồng xu đó**. Thiết kế giải thuật cho robot nhặt **nhiều đồng xu nhất** và xác định đường đi của nó khi nhặt các đồng xu đó

BÀI TOÁN ROBOT NHẬT CÁC ĐỒNG XU



Hai đường đi **nhật 5 đồng xu**

BÀI TOÁN ROBOT NHẶT CÁC ĐỒNG XU

- Gọi $F(i, j)$ là số đồng xu lớn nhất mà robot nhặt được trên đường di chuyển đến $o(i, j)$
- Robot chỉ có thể đến $o(i, j)$ từ $o(i - 1, j)$ hoặc $o(i, j - 1)$
- Số đồng xu lớn nhất mà robot nhặt khi di chuyển đến $o(i - 1, j)$ hoặc $o(i, j - 1)$ tương ứng là $F(i - 1, j)$ hoặc $F(i, j - 1)$
- Vì vậy $F(i, j)$ là bằng $F(i - 1, j)$ hoặc $F(i, j - 1)$ cộng thêm 1 nếu trong $o(i, j)$ có một đồng xu

BÀI TOÁN ROBOT NHẬT CÁC ĐỒNG XU

- Hệ thức truy hồi cho $F(i, j)$ là

$$F(i, j) = \max\{F(i - 1, j), F(i, j - 1)\} + c_{ij} \quad // \quad c_{ij}=0, 1$$

với $1 \leq i \leq n, 1 \leq j \leq m$

và $F(0, j) = 0, 1 \leq j \leq m$ và $F(i, 0) = 0, 1 \leq i \leq n,$

trong đó $c_{ij}=1$ nếu $\hat{o}(i, j)$ có một đồng xu và $c_{ij}=0$ nếu ngược lại

BÀI TOÁN ROBOT NHẶT CÁC ĐỒNG XU

- Các $o(0, j)$ thuộc dòng thứ 0, không chứa đồng xu nào nên $F(0, j) = 0$
- Các $o(i, 0)$ thuộc cột thứ 0, không chứa đồng xu nào nên $F(i, 0) = 0$

BÀI TOÁN ROBOT NHẶT CÁC ĐỒNG XU










ALGORITHM RobotCoinCollection($C[1..n, 1..m]$)

```
1   $F[1, 1] \leftarrow C[1, 1]$ 
2  for  $j \leftarrow 2$  to  $m$  do
3     $F[1, j] \leftarrow F[1, j - 1] + C[1, j]$ 
4  for  $i \leftarrow 2$  to  $n$  do
5     $F[i, 1] \leftarrow F[i - 1, 1] + C[i, 1]$ 
6    for  $j \leftarrow 2$  to  $m$  do
7       $F[i, j] \leftarrow \max(F[i - 1, j], F[i, j - 1]) + C[i, j]$ 
8  return  $F[n, m]$ 
```


BÀI TOÁN ROBOT NHẶT CÁC ĐỒNG XU

- Đường đi của robot có thể xác định bằng quay lui
 - Nếu $F(i - 1, j) > F(i, j - 1)$, đường đi đến $o(i, j)$ phải đi qua $o(i - 1, j)$
 - Nếu $F(i - 1, j) < F(i, j - 1)$, đường đi đến $o(i, j)$ phải đi qua $o(i, j-1)$
 - Nếu $F(i - 1, j) = F(i, j - 1)$, đường đi đến $o(i, j)$ có thể từ $o(i-1, j)$ hoặc $o(i, j-1)$

BÀI TOÁN ROBOT NHẶT CÁC ĐỒNG XU

	1	2	3	4	5	6
1						
2						
3						
4						
5						

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

Kết quả các $F(i, j)$

Từ ô(5, 6) ngược về ô(4, 6), ô(3,6), ..., ô(2,3), ô(2,2), ô(1,2), ô(1,1)

BÀI TOÁN ROBOT NHẬT CÁC ĐỒNG XU

- Độ phức tạp của giải thuật
 - Kích thước đầu vào là n và m
 - Thao tác cơ bản là tìm max ở dòng 7
 - Gọi thời gian thao tác cơ bản là c
 - $T(n) = (n-1)(m-1)c = \Theta(nm)$
 - Thời gian tìm đường đi khi đã có bảng kết quả là $\Theta(n+m)$

BÀI TOÁN CẢI TÚI

- Cho n đồ vật có trọng lượng là w_1, \dots, w_n và giá trị tương ứng là v_1, \dots, v_n và một cái túi có thể chứa được một **trọng lượng là W** , tìm một tập con các đồ vật **có tổng giá trị lớn nhất mà có thể đặt được vào trong túi** (giả thiết các trọng lượng của các vật là số nguyên, giá trị của chúng là số thực)

BÀI TOÁN CẢI TÚI

- Gọi $F(i, j)$ là giá trị lớn nhất của các tập con của i vật đầu tiên có trọng lượng w_1, w_2, \dots, w_i , giá trị v_1, v_2, \dots, v_i có thể đặt được trong túi trọng lượng tối đa là j

BÀI TOÁN CÁI TÚI

- Chia tất cả các tập con của i vật thành hai loại, chứa vật thứ i và không chứa vật thứ i , khi đó
 - $F(i,j) = F(i-1,j)$, nếu tập con được chọn không chứa vật thứ i
 - $F(i,j) = v_i + F(i-1, j-w_i)$, nếu tập con được chọn chứa vật thứ i , trong đó $F(i-1, j-w_i)$ là giá trị lớn nhất của tập con của $i-1$ vật không chứa vật thứ i , nếu trọng lượng w_i của vật thứ i lớn hơn j thì $F(i,j) = F(i-1, j)$

BÀI TOÁN CÁI TÚI

- Hệ thức truy hồi cho $F(i, j)$

$$F(i, j) = \begin{cases} \max\{F(i-1, j), v_i + F(i-1, j-w_i)\}, & \text{nếu } j-w_i \geq 0 \\ F(i-1, j) & \text{nếu } j-w_i < 0 \end{cases}$$

trong đó

$F(0, j) = 0$ với mọi $j \geq 0$ và $F(i, 0) = 0$ với mọi $i \geq 0$

- Bài toán ban đầu được giải quyết khi cho $i=n$ và $j=W$

BÀI TOÁN CẢI TÚI

Bảng tính toán các giá trị $F[i, j]$

		0	$j-w_i$	j	W
w_i, v_i	0	0	0	0	0
	$i-1$	0	$F(i-1, j-w_i)$	$F(i-1, j)$	
	i	0		$F(i, j)$	
	n	0			goal

BÀI TOÁN CẢI TÚI

ALGORITHM knapsack($v[1..n]$, $w[1..n]$, W)

```
1 for  $i \leftarrow 0$  to  $n$  do  $F(i, 0) \leftarrow 0$ 
2 for  $j \leftarrow 1$  to  $W$  do  $F[0, j] \leftarrow 0$ 
3 for  $i \leftarrow 1$  to  $n$  do
4   for  $j \leftarrow 1$  to  $W$  do
5     if  $j - w_i \geq 0$ 
6       then  $F[i, j] \leftarrow \max(F[i-1, j], v_i + F[i-1, j-w_i])$ 
7       else  $F[i, j] \leftarrow F[i-1, j]$ 
8 return  $F[n, W]$ 
```

BÀI TOÁN CẢI TÚI

- Độ phức tạp của giải thuật
 - Kích thước đầu vào là n và W
 - Thao tác cơ bản là so sánh ở dòng 5
 - Gọi thời gian thao tác cơ bản là c
 - $T(n) = (n-1)(W-1)c = \Theta(nW)$

BÀI TOÁN CẢI TÚI

Vật	trọng số	giá trị
1	2	\$12
2	1	\$10
3	3	\$20
4	2	\$15

TL túi $W = 5$

BÀI TOÁN CẢI TÚI

Bảng tính toán giá trị $F(i, j)$ theo trọng số j của túi

$i \backslash j$		0	1	2	3	4	5
	0	0	0	0	0	0	0
$w_1 = 2, v_1 = 12$	1	0	0	12	12	12	12
$w_2 = 1, v_2 = 10$	2	0	10	12	22	22	22
$w_3 = 3, v_3 = 20$	3	0	10	12	22	30	32
$w_4 = 2, v_4 = 15$	4	0	10	15	25	30	37

BÀI TOÁN CẢI TÚI

- Xác định tập các vật được chọn như sau:
 - Do $F(4, 5) > F(3, 5)$ nên vật thứ 4 được chọn (nếu $F(i, j) > F(i-1, j)$ thì chọn vật i , khi đó $F(i, j) = v_i + F(i-1, j-w_i)$)
 - Do $F(3, 5-2) = F(3, 3) = F(2, 3)$ nên vật thứ 3 không được chọn (do $F(4, 5) = v_4 + F(4-1, 5-w_4) = v_4 + F(3, 5-2)$)
 - Vì $F(2, 3) = v_2 + F(1, 3-1) > F(1, 3)$ nên vật thứ 2 được chọn
 - Do $F(1, 3-1) = F(1, 2) > F(0, 2)$ nên vật thứ 1 được chọn
- Vậy các vật 1, 2, 4 được chọn với tổng trọng lượng $w_1 + w_2 + w_4 = 2 + 1 + 2 = 5$ và tổng giá trị $v_1 + v_2 + v_4 = 12 + 10 + 15 = 37$

BÀI TOÁN XÂU CON CHUNG DÀI NHẤT

- Cho hai chuỗi $a=(a_1, a_2, \dots, a_m)$ và $b=(b_1, b_2, \dots, b_n)$, tìm chuỗi con chung $c=(c_1, c_2, \dots, c_k)$ của a và b sao cho độ dài của c là lớn nhất
- **Hướng dẫn:** Tìm và đọc giải thuật qui hoạch động trong tài liệu "Introduction to Algorithms của Thomas H. Cormen, Charles E. Leiserson, Ronald D. Rivest"

BÀI TẬP VỀ NHÀ

- Đọc chương 8: Dynamic Programming sách Levitin
- Làm bài tập về nhà đã cho trong DS bài tập
- Kết hợp biến đổi và chia để trị để tính số Fibonacci $f(n) = f(n-1) + f(n-2)$, $\forall n \geq 2$ và $f(0) = 1$, $f(1) = 1$
- Hiện thực các giải thuật đã học và vẽ đồ thị độ phức tạp