

# CS112.L21

Giảng viên: Thầy Nguyễn Thanh Sơn



# DYNAMIC PROGRAMMING

# Danh sách thành viên:

1. Lê Dương Khánh Việt 19522515
2. Nguyễn Tấn Tú 19522454
3. Trương Minh Tuấn 19522485
4. Nguyễn Thị Như Ý 19522555



# MỤC LỤC

---

Yêu cầu bài toán

---

Ưu điểm và nhược điểm

---

Kỹ thuật tối ưu hóa bài toán

---

Cách tiếp cận một bài toán

---

Ứng dụng

---

Bài tập



# QUY HOẠCH ĐỘNG

## 2 Yếu tố quan trọng tạo thành thuật toán quy hoạch động?

A. Phân rã, Tối ưu

B. Tối ưu, Sử dụng lại nghiệm

C. Phân rã, Sử dụng lại nghiệm

D. Không gian lưu trữ, các bước giải phải hữu hạn

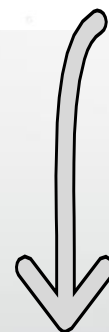
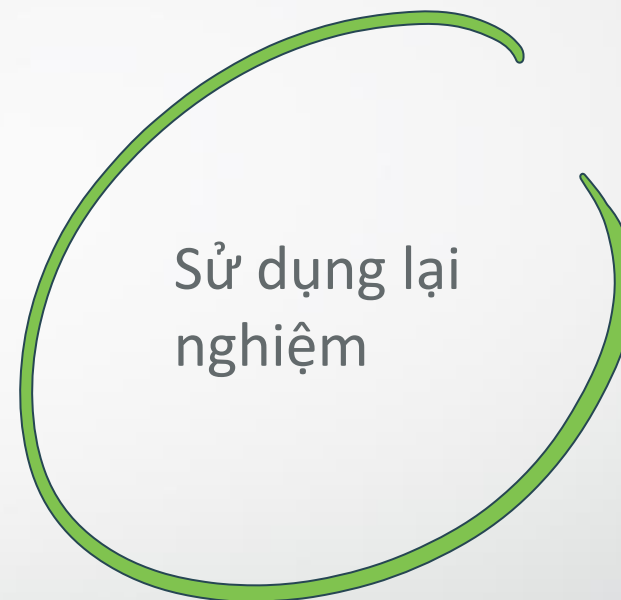
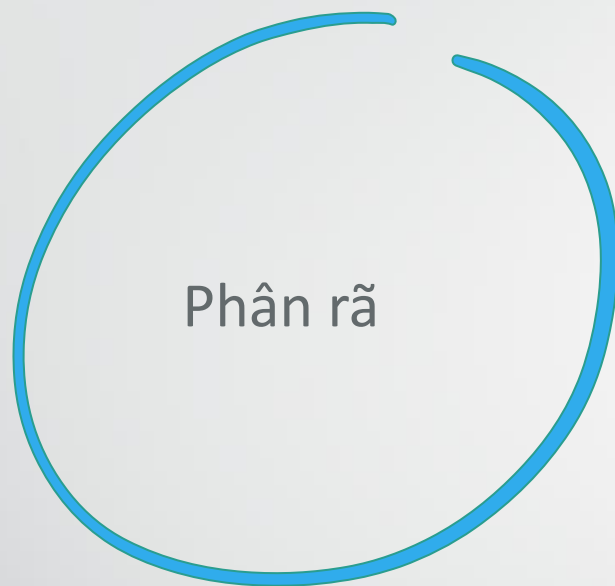
## 2 Yếu tố quan trọng tạo thành thuật toán quy hoạch động?

A. Phân rã, Tối ưu

B. Tối ưu, Sử dụng lại nghiệm

C. Phân rã, Sử dụng lại nghiệm

D. Không gian lưu trữ, các bước giải phải hữu hạn



DYNAMIC PROGRAMMING






# YÊU CẦU BÀI TOÁN

## *Yêu cầu bài toán*

- Bài toán con gối nhau
- Cấu trúc con tối ưu
- Không gian lưu trữ
- Quá trình giải là các bước hữu hạn

```
fib(n):  
    F[0] <- 0; F[1] <- 1  
    for i <- 2 to n  
        F[i] <- F[i-1] + f[i-2]  
    return F[n]
```

Bài toán con gối nhau?  
Cấu trúc con tối ưu?



# ƯU ĐIỂM & NHƯỢC ĐIỂM

Tiết kiệm thời gian,  
cho lời giải chính xác

Độ phức tạp thấp

Giải quyết các vấn đề  
tối ưu

Ưu điểm

DYNAMIC

Nhược điểm

Số lượng bài toán con  
được chia ra có thể rất lớn

Không có cách giải tổng  
quát nào cho thuật toán

Sự kết hợp các bài toán  
con chưa chắc đã cho kết  
quả của bài toán lớn



# KỸ THUẬT TỐI ƯU

Đổi  
biến

Chia  
để trị

Bao lỗi  
đường  
thẳng

Kỹ thuật  
tối ưu

Slope  
trick

Stack



# Cách tiếp cận một bài toán




Bài toán được chia thành các bài toán con, các bài toán con này được giải và lời giải được ghi nhớ để phòng trường hợp cần dùng lại chúng. Đây là đệ quy và lưu trữ được kết hợp với nhau.

Top-down

Bottom-Up

Tất cả các bài toán con có thể cần đến đều được giải trước, sau đó được dùng để xây dựng lời giải cho các bài toán lớn hơn

A silver microphone is the central focus. It has a black band around its middle with the text "PHONE" and "IMP. 600 Ω". A pair of black-rimmed glasses is perched on top of the microphone's grille. To the left of the microphone, a hand-drawn notepad with lines is attached to the stand. To the right, a hand-drawn arm with two fingers pointing up is attached to the stand. The background is a solid light blue.

# ỨNG DỤNG của thuật toán quy hoạch động cho tính khoảng cách

## ĐIỀU KHIỂN MÀN HÌNH TỪ XA





SỬA LỖI CHÍNH TẢ



## PHÁT HIỆN ĐẠO VẤN



# Bài toán ôn tập: Đồng xu

Giả sử chúng ta có  $n$  loại đồng xu nặng lần lượt là **W1**, **W2**, **W3**, ..., **Wn** và bài toán đặt ra là tìm số lượng đồng xu nhỏ nhất để tổng khối lượng của chúng là một giá trị  $S$ . Số lượng đồng xu là không giới hạn

## Với bài toán này :

- Input :  $n, S, W[]$
- Áp dụng giải thuật Dynamic Programming:
  - + Các bài toán con  $dp(P)$  gộp lên nhau với  $P$  là khối lượng các đồng xu và  $P \leq S$
  - +  $dp(P) = k$ , với  $k$  là số lượng đồng xu nhỏ nhất để khối lượng của chúng bằng  $P$
- Yêu cầu : mỗi bài toán con  $dp(P)$  phải được tối ưu. Trong bài này là mỗi bài toán con phải nhỏ nhất



Với input:  $n = 3$ ,  $S = 11$ ,  $W = [1, 3, 5]$ .  
Bài toán con  $dp(P)$ ,  $P \leq 11$

- Với bài toán con 0, ta có  $dp(0) = 0$
- Với bài toán con 1, ta có thể thêm đồng xu (nặng 1) vào 0 đồng xu nào cả.  
Vậy  $dp(1) = dp(0) + 1 = 1$
- Với bài toán con 2, ta có thể thêm đồng xu (nặng 1) vào 1 đồng xu ở bài toán con 1.  
Vậy  $dp(2) = dp(1) + 1 = 2$
- Với bài toán con 3, ta có thể thêm đồng xu (nặng 1) vào 2 đồng xu ở bài toán con 2 **hoặc** thêm đồng xu (nặng 3) vào 0 đồng xu.

Do yêu cầu của bài toán là ở mỗi bài toán con phải tối ưu, nghĩa là các bài toán con trong bài này phải nhỏ nhất do đó:

$$\Rightarrow dp(3) = \min(dp(2) + 1, dp(0) + 1) = \min(3, 1) = 1$$

Tiếp tục làm cho đến khi  $P = S = 11$



# Về phần cài đặt code

- Mảng **dp[0...S]** sẽ lưu kết quả cho từng bài toán con
- Do đó **dp[P] = k**, nghĩa là cần ít nhất **k** đồng xu để khối lượng toàn bộ đồng xu đó bằng **P**

Code :

```
1  n, S = map(int, input().split())
2  w = list(map(int, input().split()))
3  dp = [0] * (S + 1)
4  dp[0] = 0
5
6  for P in range(1, S + 1):
7      dp[P] = min(dp[P - x] for x in w if x <= P) + 1
8
9  print(dp)
10 print(dp[S])
```

3 11

1 3 5

[0, 1, 2, 1, 2, 1, 2, 3, 2, 3, 2, 3]

3

# Bài toán ôn tập: Chuỗi chung dài nhất

- Cho chuỗi A độ dài m, chuỗi B độ dài n. Hãy tìm độ dài chuỗi con chung dài nhất của hai chuỗi, chú ý là chuỗi con chung có thể không liên tiếp.

- Ví dụ: A = ADBCC

B = ABCD

Chuỗi con chung dài nhất của hai chuỗi là : ABC

- Đáp án là : 3

# Với bài toán này:

- Input: A, B
- Mỗi bài toán con là  $dp(i,j)$  với  $i$  là  $i$  kí tự đầu tiên của xâu A và  $j$  là  $j$  kí tự đầu tiên của xâu B. Với  $i \leq \text{độ dài xâu A}$ ,  $j \leq \text{độ dài xâu B}$ .
- Mỗi bài toán con  $dp(i,j)$  đều phụ thuộc vào các bài toán con trước đó  $dp(i-1, j)$ ,  $dp(i, j-1)$ ,  $dp(i-1, j-1)$
- Do đó yêu cầu của bài toán là: Mỗi bài toán con  $dp(i,j)$  phải được tối ưu sao cho  $dp(i,j) = \text{độ dài xâu con chung dài nhất}$

#		B			A	B	C	D
#	A	ij		0	1	2	3	4
#	-----+-----							
#		0		0	0	0	0	0
#	A	1		0	1	1	1	1
#	D	2		0	1	1	1	2
#	B	3		0	1	2	2	2
#	C	4		0	1	2	3	3
#	C	5		0	1	2	3	3

## Code:

```
1 A, B = input().split()
2 dp = [[0] * (len(B) + 1) for _ in range(len(A) + 1)]
3 for i, x1 in enumerate(A, 1):
4     for j, x2 in enumerate(B, 1):
5         if x1 == x2:
6             dp[i][j] = dp[i - 1][j - 1] + 1
7         else:
8             dp[i][j] = max(dp[i][j - 1], dp[i - 1][j])
9
10 print(*dp)
11 print(dp[-1][-1])
```

Bấm để thêm nội dung

adbcc abcd

```
[0, 0, 0, 0, 0] [0, 1, 1, 1, 1] [0, 1, 1, 1, 2] [0, 1, 2, 2, 2] [0, 1, 2, 3, 3] [0, 1, 2, 3, 3]
3
```

<https://vnoi.info/wiki/algo/dp/Mot-so-ky-thuat-toi-uu-hoa-thuat-toan-Quy-Hoach-Dong.md>



□ Link bài tập về nhà:

[www.hackerrank.com/on-tap-dynamic-programming](http://www.hackerrank.com/on-tap-dynamic-programming)

Cảm ơn Thầy và các bạn đã chú ý  
lắng nghe

