DYNAMIC PROGRAMMING PART 2

GROUP 2

TABLE OF CONTENT

01

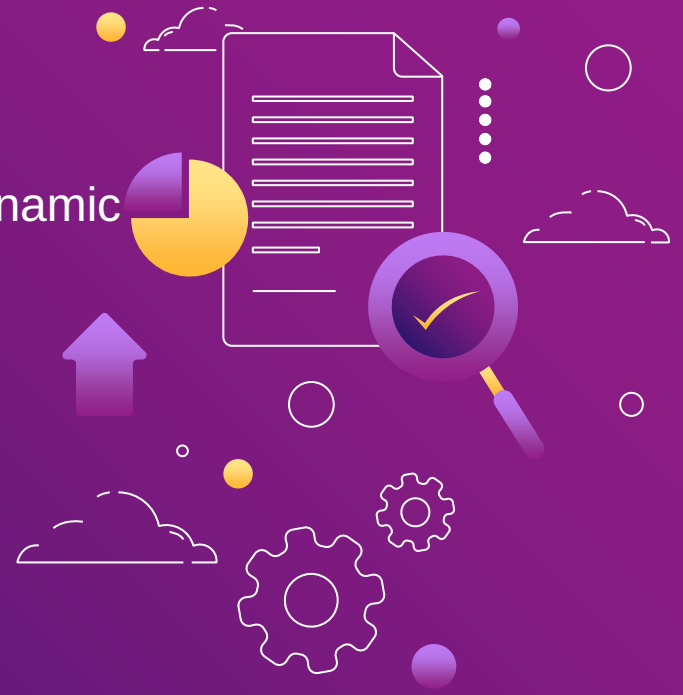
LONGEST INCREASING SUBSEQUENCE

REVIEW

Các bước để giải bài toán Dynamic Programing

1. Nhận dạng bài toán có thể giải bằng DP không
2. Tìm ra trạng thái với ít tham số nhất
3. Tìm ra mối quan hệ giữa các trạng thái
4. Sử dụng phương pháp top-down hoặc bottom up để giải

IDENTIFY DP PROBLEM



Cách để nhận dạng bài toán giải được bằng Dynamic Programing

1. Overlapping subproblems
2. Optimal structure

IDENTIFY DP PROBLEM

Cách để nhận dạng nhanh hơn ?

1. Bài toán tối ưu vd: Longest Common Subsequence, 0–1 Knapsack problem, optimal binary search tree, Longest increasing subsequences

2. Bài toán tổ hợp vd: coin change, Count Balanced Binary Trees of Height h, Count Distinct Subsequences,...

Link để tham khảo thêm

INTRODUCING PROBLEM

Tìm ra chiều dài lớn nhất của chuỗi con tăng (các phần tử của nó có độ lớn và vị trí tăng dần).
VD: Input: arr[] = {3, 10, 2, 1, 20}
Output: Length of LIS = 3
The longest increasing subsequence is {3, 10, 20}

Input: arr[] = {3, 2}
Output: Length of LIS = 1
The longest increasing subsequences are {3} and {2}

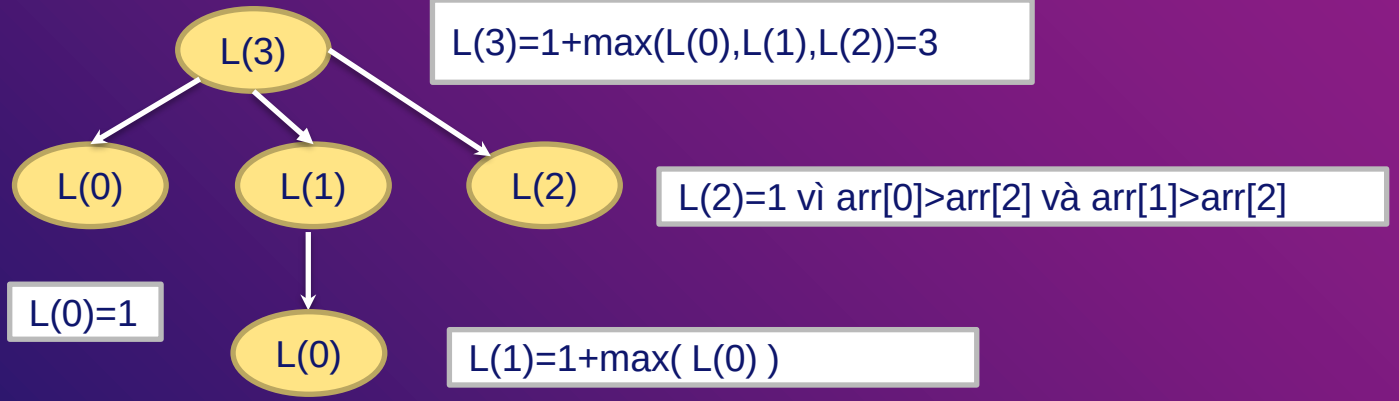
Input: arr[] = {50, 3, 10, 7, 40, 80}
Output: Length of LIS = 4
The longest increasing subsequence is {3, 7, 40, 80}

INTRODUCING PROBLEM

Link

ANALYSING PROBLEM

Optimal structure: L(i) = 1 + max(L(j)) where 0 ≤ j < i and arr[j] < arr[i]; or
L(i) = 1, if no such j exists.
Overlapping subproblems: L(i)=1+max(L(0),..,L(i-1))
Input : arr[] = {3, 10, 2, 11}



SOLVING PROBLEM

BOTTOM-UP

Ý TƯỞNG

- Khởi tạo một mảng phụ dùng để lưu chiều dài chuỗi con ở mỗi vị trí
- Giá trị của các phần tử đều bằng 1 và số phần tử là n
- Tính toán chiều dài của mỗi chuỗi con tăng tại mỗi vị trí và lưu lại vào mảng phụ
- Duyệt qua mảng phụ để lấy giá trị lớn nhất

SOLVING PROBLEM

BOTTOM-UP CODE:

OPTIMAL STRUCTURE

```python
def lis(arr):
    n = len(arr)

    # khai báo mảng phụ lis lưu chiều dài của chuỗi con tăng
    # khởi tạo giá trị 1 cho các phần tử trong mảng
    lis = [1]*n

    # Tính toán chiều dài chuỗi con tăng tại mỗi index
    #Lưu lại vào trong mảng lis để có thể truy xuất
    for i in range(1, n):
        for j in range(0, i):
            if arr[i] > arr[j] and lis[i] < lis[j] + 1:
                lis[i] = lis[j]+1

    # Khởi tạo biến lưu chiều dài lớn nhất
    maximum = 0

    # Tìm ra giá trị lớn nhất
    for i in range(n):
        maximum = max(maximum, lis[i])

    return maximum
```

02

COIN CHANGE

INTRODUCING PROBLEM

Cho N là giá trị tiền cần trao đổi. Giả sử chúng ta có chỉ vài loại đồng xu kí hiệu là S={S1,..,Sm}. Cần tính số cách để đổi được số tiền N với tập đồng xu có giá trị S

VD: Input: N = 4, S = {1,2,3}

Output: 4

Vì có 4 cách để trả được 4 nghìn theo 4 cách từ tập hợp loại đồng xu trên gồm {1,1,1,1},{1,1,2},{2,2},{1,3}

Input: N = 10, S = {2, 5, 3, 6}

Output: 5

{2,2,2,2,2}, {2,2,3,3}, {2,2,6}, {2,3,5}, {5,5}

INTRODUCING PROBLEM

Link

TEAM DISCUSSION

Optimal structure của bài toán ?
Ý tưởng của bài toán dựa trên bottom-up?

ANALYSING PROBLEM

Optimal structure:
1) Giải pháp không chứa đồng xu cuối Sm
2) Giải pháp chứa ít nhất một lần chọn đồng xu cuối Sm.
Cho hàm count(S[], m, n) dùng để tính số cách, dựa vào việc phân tích trên thì nó được viết lại là count(S[], m-1, n) and count(S[], m, n-Sm).
Overlapping subproblems: Có

SOLVING PROBLEM

BOTTOM-UP

Ý TƯỞNG

+ Tạo một mảng một chiều có kích thước bằng số tiền N + 1, giá trị được lưu trữ trong mảng sẽ là tổng số lượng kết hợp cho các số tiền khác nhau
+ Lặp qua toàn bộ mảng và lấp đầy bằng công thức logic sau:

```
IF AMOUNT >= COIN THEN
COMBINATIONS[AMOUNT]+=COMBINATIONS[AMOUNT-COIN]
```

THE CODE OF COIN CHANGE-MAKING PROBLEM

```
def count(S, m, n):
    table = [0 for k in range(n+1)]
    table[0] = 1
    for i in range(0,m):
        for j in range(S[i],n+1):
            table[j] += table[j-S[i]]

    return table[n]
```

03

PROS AND CONS

PROS AND CONS

CONS

- 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

VS

PROS

- Tiết kiệm thời gian, cho lời giải chính xác
- Độ phức tạp thấp
- Giải quyết các vấn đề tối ưu

04

QUIZ