Dynamic Programming

Presented by Jim Yang

Dynamic Programming Overview

1. Dynamic Programming (动态规划) 是一种将问题分解为**子问题**的算法
    a. 类似于Recursion (递归)

Dynamic Programming Overview

2. 牺牲空间来获得时间上的优势 (Trade memory for time efficiency)

Dynamic Programming

1. Recursion
2. Memoization
3. Bottom-Up Approach

Fibonacci Number Problem

1, 1, 2, 3, 5 , 8, 13….

Fibonacci Number Problem (The Recursion Way)

```
def fib (n):
    assert (n > 0)
    if n == 1 or n == 2:
        return 1
    else:
        return fib (n - 1) + fib(n - 2)
```

Memoization (记录)

In computing, **memoization** or memoisation is an optimization technique used primarily to speed up computer programs by **storing the results of expensive function calls and returning the cached result** when the same inputs occur again.

Fibonacci Number Problem (the memoization way)

```
def fib (n, memo):
    assert (n > 0)
    if memo[n] != null:
        return memo[n]
    if n == 1 or n == 2:
        result = 1
    else:
        result = fib (n-1) + fib (n-2)
    memo[n] = result
    return result
```

Fibonacci Number Problem (Bottom-Up Approach)

```
def fib (n):
    assert (n > 0)
    if n == 1 or n == 2:
        return 2
    D = [0] * (n+1)
    D[1] = 1
    D[2] = 2
    for i from 3 to n:
        D[i] = D[i-1] + D[i-2]
    return D[n]
```

Dynamic Programming 三个步骤

1. Recursion(递归)

   用Recursion尝试将问题拆解成子问题

2. Memoization

   尝试用更多内存去减少递归次数

3. Bottom-Up Approach

   仅仅用内存解决问题

Fibonacci Number Problem (Stairsteps Problem)

There are n many stairsteps in a stair. Each time a person can either choose to go up 1 step, or 2 steps. Given a number n, return the total number of ways a person can finish walking up a stair.

i.e. 每次可以走一步，或者两步，那么n层阶梯需要几步完成?

Fibonacci Number Problem (Stairsteps Problem)

1. Recursion

   Let T(n) = Total Number of Ways to finish a staircase of length n

   How to decompose T(n) into T(n-1), T(n-2)..etc?

Fibonacci Number Problem (Stairsteps Problem)

2. Memoization

   Let D[n] = number of ways to finish n staircases.

   How can we write D[n] in terms of D[n-1] or D[n-2]?

Minimum Coin Problem (最少零钱问题)

Given a value V, if we want to make change for V cents, and we have infinite supply of each of C = { C1, C2, .. , Cm} valued coins, what is the minimum number of coins to make the change?

Input: C = {25, 10, 5}, V = 30

Output: Minimum 2 coins required. 1个25元硬币和1个5元硬币

Input: C = {9, 6, 5, 1}, V = 11

Output: Minimum 2 coins required. 1个6元硬币和1个5元硬币

Dynamic Programming 三个步骤

1. Recursion(递归)

   尝试将问题拆解成子问题

Minimum Coin Problem (最少零钱问题)

2. Memoization & Bottom-up Approach

Let D[n] = minimum number of ways we can hit the target

Can we write D[n] in terms of D[n1],D[n2]...where n1, n2 < n ?

Minimum Coin Problem (最少零钱问题)

```
def minCoins(coins, V):

    D = [0 for i in range(V + 1)]
    table[0] = 0

    for i in range(1, V + 1):
        table[i] = sys.maxsize

    for i in range(1, V + 1):
        for j in range(m):
            if (coins[j] <= i):
                sub_res = table[i - coins[j]]
                if (sub_res != sys.maxsize and
                    sub_res + 1 < table[i]):
                    table[i] = sub_res + 1
    return table[V]
```

Dynamic Programming 三个步骤

1. Recursion(递归)

   用Recursion尝试将问题拆解成 `1` 和 `n - 1`

2. Memoization

   尝试用更多内存去减少递归次数

3. Bottom-Up Approach

   仅仅用内存解决问题