背包問題

11/07/2019

Road map

- 數學歸納法
- Subset
- Backpack II
- Backpack V

數學歸納法

- N=1 是否成立
- If N = n-1 成立， N = n 成立， n>=2

- 滿足以上兩條則 N >= 1都成立

問題空間

- F(n)

- F: 問題的定義
- n: 表徵問題尺寸的參數

- 可以是多維的，比如 F(i, j)

從動態規劃的觀點解決問題 F(N)

- F(1) 可以解出來
- F(n) 的答案可以通過 F(n - 1)的答案轉化而來

- 如果以上兩條成立，則可以解出F(N) 對任意 N>=1

從大到小的解法（遞歸）好寫

```
def main():
        return dfs(N)


def dfs(n):
        # 遞歸出口
        if n == 1:
                ans = 手動算出
                return ans

        # 遞歸運算
        ans = 某種運算(dfs(n-1))
        return ans
```

從小到大的解法（循環）運算更快

```
def main():
        # 定義解空間 dp[n] 表示大小為n的問題的解
        dp = [0 for i in range(N)]

        # N=1 可解
        dp[0] = 手動算出

        # 答案轉化
        for n in range(1, N):
                dp[n] = 某種運算(dp[n-1])

        # 返回
        return dp[N-1]
```

sub set

- Leet 78
- lint code：https://www.lintcode.com/problem/subsets/description?_from=ladder&&fromId=1

Backpack II

- Lint code
- https://www.lintcode.com/problem/backpack-ii/description

背包問題

有 `n` 个物品和一个大小为 `m` 的背包. 给定数组 `A` 表示每个物品的大小和数组 `V` 表示每个物品的价值.

问最多能装入背包的总价值是多大?

1. `A[i], V[i], n, m` 均为整数
2. 你不能将物品进行切分
3. 你所挑选的要装入背包的物品的总大小不能超过 `m`
4. 每个物品只能取一次

解空間

- dp[y][i]表達以下問題的解
- 在考慮前i件物品的前提之下
- 一個大小為y的包能裝下的物品
- 價值的最大值

基礎問題的解

- 基礎狀況長什麼樣子
  - 背包什麼也裝不下：y=0
  - 沒有物體：i=0
- 基礎狀況的答案
  - dp[y][0] = 0 注定
  - dp[0][i] = 0 注定

可以通過初始化dp[][]為全零來實現

問題答案的轉換

- 每新加入一個物體（第i個），會面臨兩個選擇。根據不同的選擇，有不同的轉換規則。
  - 裝入（如果包夠大的話）
    - 包被分爲兩部分，價值相加
      - 包含該物體的部分，價值已知 V[i]
      - 不包含該物體的部分，可以看作一個小一點的包，這部分能得到的最大價值可以參考之前的答案 dp[y-A[i]][i-1]
  - 不裝入
    - 那麼和沒這個物體沒區別 dp[y][i-1]
- 然而因爲要的是“最大價值”，所以選擇價值最大的那個

```
dp[y][i] = max([
    V[i] + dp[y-A[i]][i-1],
    dp[y][i-1]
])
```

Backpack V

- Lint code
- https://www.lintcode.com/problem/backpack-v/description