

## 第四章-作业

### 1

在一条直线上有  $n$  堆石子，每堆有一定的数量，每次可以将两堆相邻的石子合并，合并后放在两堆的中间位置，合并的费用为两堆石子的总数。（30分）

求把所有石子合并成一堆的最小花费(定义  $dp[i][j]$  为第  $i$  堆石子到第  $j$  堆合并的最小花费)。

(1) 写出该问题的递推方程。（10分）

(2) 有 5 堆石子 ( $n = 5$ )，每堆石子大小分别为  $\langle 1, 3, 5, 2, 4 \rangle$ ，求出把所有石子合并成一堆的最小花费(要求写出运算矩阵)。（10分）

(3) 写出该问题的伪代码。（10分）

答:

设  $w[i]$  为第  $i$  堆石子的数量;

$R[i][j]$  为第  $i$  堆石子到第  $j$  堆石子的总数量，即  $R[i][j] = \sum_{r=i}^j w[r]$ 。

(1)

$$dp[i][j] = \begin{cases} 0, & \text{if } i = j; \\ \min_{i \leq k < j} \{dp[i][k] + dp[k+1][j] + R[i][j]\}, & \text{if } j > i. \end{cases}$$

(2)

$R[1][1] = 1$	$R[1][2] = 4$	$R[1][3] = 9$	$R[1][4] = 11$	$R[1][5] = 15$
$\backslash$	$R[2][2] = 3$	$R[2][3] = 8$	$R[2][4] = 10$	$R[2][5] = 14$
$\backslash$	$\backslash$	$R[3][3] = 5$	$R[3][4] = 7$	$R[3][5] = 11$
$\backslash$	$\backslash$	$\backslash$	$R[4][4] = 2$	$R[4][5] = 6$
$\backslash$	$\backslash$	$\backslash$	$\backslash$	$R[5][5] = 4$

$dp[1][1] = 0$	$dp[1][2] = 4$	$dp[1][3] = 13$	$dp[1][4] = 22$	$dp[1][5] = 34$
$\backslash$	$dp[2][2] = 0$	$dp[2][3] = 8$	$dp[2][4] = 17$	$dp[2][5] = 28$
$\backslash$	$\backslash$	$dp[3][3] = 0$	$dp[3][4] = 7$	$dp[3][5] = 17$
$\backslash$	$\backslash$	$\backslash$	$dp[4][4] = 0$	$dp[4][5] = 6$
$\backslash$	$\backslash$	$\backslash$	$\backslash$	$dp[5][5] = 0$

(3)

$stone\_merge(n, w[])$

```
1  for i <- 1 to n do
2      dp[i][i] <- 0
3      R[i][i] <- w[i]
4  for l <- 2 to n do
5      for i <- 1 to n - l + 1 do
6          j = i + l - 1
7          dp[i][j] = MAX_NUM
8          R[i][j] = R[i][j - 1] + w[j]
9          for k <- i to j - 1 do
10             q = dp[i][k] + dp[k + 1][j] + R[i][j]
11             if(q < dp[i][j]) then
12                 dp[i][j] = q
13  return dp[1][n]
```

c 语言代码: [1.c](#)

## 2

若 7 个关键字的概率如下所示，求其最优二叉搜索树的结构和代价，要求必须写出递推方程。（30分）

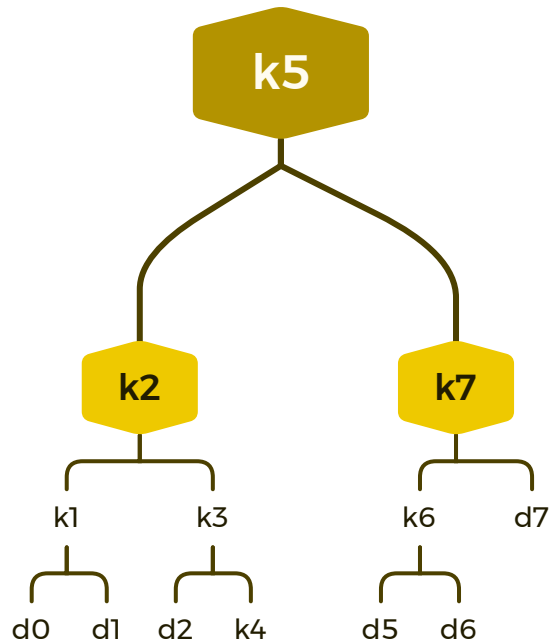
$i$	0	1	2	3	4	5	6	7
$p_i$		0.04	0.06	0.08	0.02	0.10	0.12	0.14
$q_j$	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05

答：

递推方程

$$E(i, j) = \begin{cases} q_{i-1} = q_j, & \text{if } j = i - 1; \\ \min_{i \leq k \leq j} \{E(i, k-1) + E(k+1, j) + W(i, j)\}, & \text{if } j > i. \end{cases}$$
$$W(i, j) = \sum_{l=i}^j p_l + \sum_{l=i-1}^j q_l$$

最优二叉搜索树的结构



代价为 3.12.

### 3

编程题：兑换零钱问题（40分）

题目描述：

给定不同面额的硬币 *coins* 和一个总金额 *amount*。编写一个函数来计算可以凑成总金额所需的最少的硬币个数。如果没有任何一种硬币组合能组成总金额，返回 -1。（提示：你可以认为每种硬币的数量是无限的）。

示例 1:

输入: `coins = [1, 2, 5]`, `amount = 11`

输出: 3

解释:  $11 = 5 + 5 + 1$

示例 2:

输入: `coins = [2]`, `amount = 3`

输出: -1

运用动态规划的思想作答，请写出分析过程和状态转移方程，并用一种语言（最好是 *C++* 或 *JAVA*）实现你的思路，并保证代码能正确运行，复杂度尽可能低。

原题：[322. 零钱兑换 - 力扣 \(LeetCode\)](#)

答：

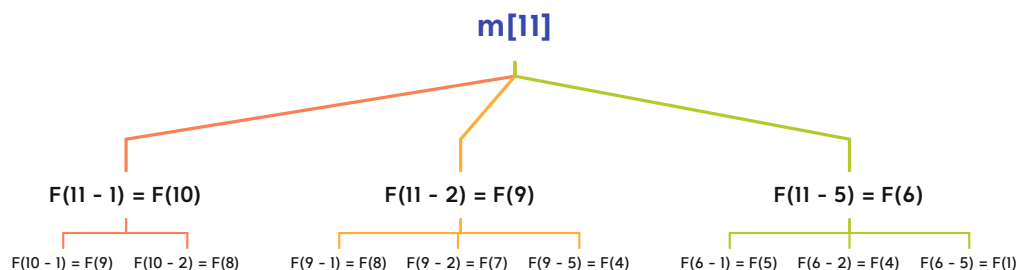
1. 分析优化解的结构

设凑成总金额  $i$  所需的最少的硬币个数为  $dp[i]$ ，设选择的最后一枚硬币为第  $j$  枚硬币，其面值为  $c_j$ 。

若  $dp[i - c_j]$  存在，则求  $dp[i - c_j]$  是求  $dp[i]$  的子问题。对应于子问题  $dp[i - c_j]$  的解是子问题  $dp[i - c_j]$  的优化解。

证明：反证法略

子问题的重叠性：上示例 1



## 2. 递归地定义最优值的代价

$$dp[i] = \begin{cases} 0, & \text{if } i = 0; \\ \min_{0 \leq k \leq \text{coinsSize}-1} \{dp[i], dp[i - c_j] + 1\}, & \text{if } i > c_j. \end{cases}$$

## 3. 自底向上地计算优化解的代价保存之，并获取构造最优值的信息 & 根据构造最优值的信息构造优化解

一维数组，由小到大计算即可。

## 4. 源代码

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int min(int a, int b)
4  {
5      return (a < b) ? a : b;
6  }
7
8  int coinChange(int *coins, int coinsSize, int amount)
9  {
10     int MAX_NUM = amount + 1;
11     if (coins == NULL || coinsSize == 0 || amount < 0) return -1;
12     if (amount == 0) return 0;
13     if (coinsSize == 1 && (amount % coins[0])) return -1;
14
15     int dp[amount + 1];
16
17     for (int i = 0; i <= amount; i++)
18         dp[i] = MAX_NUM;
19     dp[0] = 0;
20
21     for (int i = 0; i <= amount; i++)
22         for (int j = 0; j < coinsSize; j++)
23         {
24             if (i < coins[j]) continue;
25             dp[i] = min(dp[i], dp[i - coins[j]] + 1);
26         }
27 }
```

```

28     return (dp[amount] == MAX_NUM) ? -1 : dp[amount];
29 }
30
31 int main()
32 {
33     int coins1[4] = {0, 1, 2, 5};
34     int amount1 = 11;
35     int coins2[2] = {2};
36     int amount2 = 3;
37     int coins3[9] = {328, 122, 26, 397, 252, 455, 250, 252};
38     int amount3 = 7121;
39     int coins4[9] = {430, 360, 440, 204, 206, 194, 150, 443};
40     int amount4 = 3580;
41     int coins5[9] = {259, 78, 94, 130, 493, 4, 168, 149};
42     int amount5 = 4769;
43     int coins6[9] = {244, 125, 459, 120, 316, 68, 357, 320};
44     int amount6 = 9793;
45     int coins7[2] = {2147483647}; //超大 coins 面额打击, 大于 amount 的
coins 直接不考虑
46     int amount7 = 2;
47
48     printf("%d\n", coinChange(coins1, 3, amount1));
49     printf("%d\n", coinChange(coins2, 1, amount2));
50     printf("%d\n", coinChange(coins3, 8, amount3));
51     printf("%d\n", coinChange(coins4, 8, amount4));
52     printf("%d\n", coinChange(coins5, 8, amount5));
53     printf("%d\n", coinChange(coins6, 8, amount6));
54     printf("%d\n", coinChange(coins7, 1, amount7));
55     return 0;
56 }

```

源文件: [3.c](#)