10.动态规划

动态规划

1. 确定dp数组及其下标含义
2. 确定递推公式
3. dp数组初始化
4. 确定遍历顺序
5. 举例推导dp数组
   - 递归改动态规划，有几个可变参数决定你维护的dp数组是几维数组
1. 状态表示 集合（所有选法、条件）
2. 状态计算 集合划分

一些文章

- link
- AcWing算法基础课

分类

记忆化搜索（**答案一定要是return值**）

```
int dp[N][N][N] ;
int dfs(int a, int b, int c){
    if(越界/不符合条件) return ...;

    if(dp[a][b][c]已经算过) return dp[a][b][c];

    int ans = 0;

    for(...){
        //或者是在这里if(越界/不符合条件) return ...;
        ans += dfs(...) ;
    }

    //记录
    dp[a][b][c] = ans;
    return ans;
}
```

- 576. 出界的路径数
- 剑指 Offer II 112. 最长递增路径

1. 背包dp

！以下问题是背包最大价值是多少，不一定装满

1-0背包问题分析

- 常见背包问题：
  - 组合问题：dp[i] += dp[i - num]
  - True/False问题：dp[i] = dp[i] or dp[i-num]
  - 最大最小问题：dp[i] = min(dp[i], dp[i-num]+1)或者dp[i] = max(dp[i], dp[i-num]+1)

1-1 01背包(每件物品最多只用一次，0和1两种情况)

n个物品（体积vi，价值wi），容量是v的背包，每个物品只能用一次

总体积小于等于v，总价值最大

```
//二维
dp[n][m] 放入1-n件物品，重量不超过m的最大价值

//初始化（不需要）
dp[0][m]都是0，因为装入0件物品


//放入1-i件物品
for(int i = 1; i <= n; i++){
    //背包容量为j
    for(int j = 0; j <= m; j++){
        dp[i][j] = dp[i-1][j]
        if(j >= weight[i]) dp[i][j] = max(dp[i][j], dp[i-1][j-weight[i]) + value
    }
}
```

```
//一维，滚动数组
dp[n]

//放入1-i件物品
for(int i = 1; i <= n; i++){
    //背包容量为j
    //从后向前遍历，相当于之前的数还没改变，还是上一层物品的值
    //为忽略判断条件，要满足j >= v[i]
    for(int j = m; j >= v[i]; j--){
        dp[j] = max(dp[j], dp[j - weight[i]] + value[i]);
    }
}
```

1-2 完全背包(每件物品有无限个)

- 279. 完全平方数
- 322. 零钱兑换（求max）
- 518. 零钱兑换 II（求和，+）

```
//二维 朴素做法
for(int i = 1; i <= n; i++){
    for(int j = 0; j <= m; j++){
        //k为0，相当于不放物品第i件时的最大值，就是dp[i-1][j]
```

```
        for(int k = 0; k * weight[i] <= j; k++){
            //max第二项包括了dp[i-1][j] 的情况，故第一个为dp[i][j]
            dp[i][j] = max(dp[i][j], dp[i-1][j-k * weight[i] + value[i] * k);
        }
    }
}

```

优化思路：

$$f[i,j] = f[i-1, j - v[i]*k] + w[i]*k$$

f[i,j-v]+w

$$f[i, j] = Max(f[i-1,j], f[i-1, j-v]+w, f[i-1,j-2v]+2w, f[i-1,j-3v]+3w,....)$$

$$f[i,j-v] = Max(\quad f[i-1,j-v], \quad f[i-1,j-2v]+w, \quad f[i-1,j-3v]+2w,.....)$$

比较：

$$f[i,j] = Max(f[i-1],j], f[i-1,j-v]+w)$$

$$f[i,j] = Max(f[i-1,j], f[i,j-v]+w)$$

```
//优化
for(int i = 1; i <= n; i++){
    for(int j = 0; j <= m; j++){
        dp[i][j] = dp[i-1][j];
        if(j >= weight[i]) dp[i][j] = max(dp[i][j], dp[i][j - weight[i]) + value
        }
    }
}
```

```
//一维

for(int i = 1; i <= n; i++){
    for(int j = v[i]; j <= m; j++){
        dp[j] = max(dp[j], dp[j - weight[i]] + value[i]);
    }
}
```

01背包从大到小循环

完全背包从小到达循环

1-3 多重背包(每件物品有si个)

```
//二维 朴素做法
for(int i = 1; i <= n; i++){
    for(int j = 0; j <= m; j++){
        //k为0，相当于不放物品第i件时的最大值，就是dp[i-1][j]
        for(int k = 0; k * weight[i] <= j && k <= s[i]; k++){
            //max第二项包括了dp[i-1][j] 的情况，故第一个为dp[i][j]
            dp[i][j] = max(dp[i][j], dp[i-1][j-k * weight[i]) + value[i] * k;
```

```
        }
    }
}

```

优化：
二进制->十进制的启发

s=200打包成1，2，4，8，16，32，64，73 = 127 + 73

所以0-127能凑出来，73-200能凑出来

```
拆分
for(int i = 1; i <= n; i++){
    for(int j = m; j >= v[i]; j--){
        f[j] = max(dp[j], dp[j - weight[i]] + value[i]);
    }
}

```

1-4 分组背包(物品有N组，每一组物品有若干个，每一组只能选一个)

只从前i组物品各选一件，并且总体积不大于j

```
for(int i = 1; i <= n; i++){
    for(int j = m; j >= 0; j--){
        //s[i]是第i组有几个物品
        for(int k = 0; k < s[i]; k++){
            if(v[i][k] <= j){
                f[j] = max(f[j], f[j - v[i][k]] + w[i][k]);
            }
        }
    }

}

```

2. 线性dp

单串：

最长上升子序列（LIS）

- 300. 最长递增子序列（Ologn的算法同样重要，贪心二分）
- 最大子数组和系列?

打家劫舍

股票问题

- 题目
  - 121. 买卖股票的最佳时机 限定交易次数 k=1
  - 122. 买卖股票的最佳时机 II 交易次数无限制

- 123. 买卖股票的最佳时机 III 限定交易次数 k=2
- 188. 买卖股票的最佳时机 IV 限定交易次数 k=?
- 309. 最佳买卖股票时机含冷冻期 含有交易冷冻期
- 714. 买卖股票的最佳时机含手续费 每次交易含手续费

- 讲解：股票问题总结

双串

- 最长公共子序列（LCS）
- 编辑距离
- 题目

a. 1143. 最长公共子序列

3. 区间dp

- 模板

```
for(int len = 2; len <= n; ++len){
    for(int i = 0; i <= n - len; ++i){
        int j = i + len - 1;
            for(int k = i + 1; k < j; k++){
                    p[i][j] = ...;
            }
    }
}
```

- 题目

a. 312. 戳气球

b. 375. 猜数字大小 II

4. 计数dp

- 求个数
- 题目

a. AcWing 900. 整数划分

b. 62. 不同路径

c. 63. 不同路径 II

d. 96. 不同的二叉搜索树

5. 数位dp

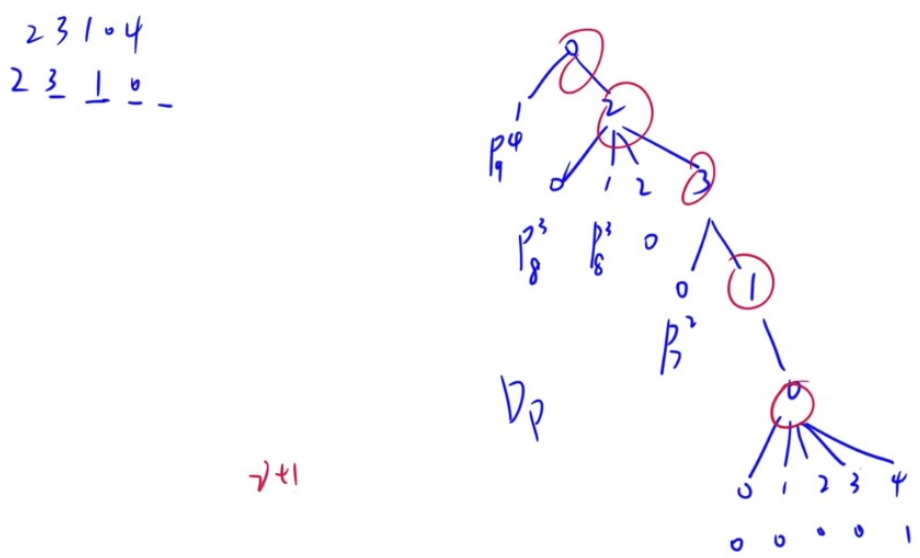
- 计算1-n中某个数字出现的个数
- 如果计算a-b用前缀和
- 题目

a. 233. 数字 1 的个数

b. 剑指 Offer 43. 1～n 整数中 1 出现的次数

c. 902. 最大为 N 的数字组合

- 类型要求：计算每一位都不相同的数字个数
  - 基础：计算n位数字，有多少这样的数（357）
  - 进阶：给一个数，不超过这个数合法数字的个数（1012、2376）
  - ps1：P94就是9 * 8 * 7 * 6



- 题目：

a. 357. 统计各位数字都不同的数字个数

b. 1012. 至少有 1 位重复的数字

c. 2376. 统计特殊整数

6. 状压dp

- 用01序列表示状态

7. 树形dp

- 题目

a. 337. 打家劫舍 III

一些新的理解

- dp方程定义、状态转移、初始化、遍历顺序、举例验证
- 线性dp：从前到后
- 区间dp：从短到长

```
//仅与前一个有关
for(len){
    for(i){
        int l = i, r = i + len;
        dp...
    }
}
//与前n个有关
for(len){
    for(i){
        int l = i, r = i + len;
        for(k: l~r){
```

```
            dp...
        }
    }
}
```

- 状压dp：用一个32位二进制数表示（是否使用选取/使用的）状态