目录

1 F-Dividing 题解

1.1 背包问题

1.1.1 题目包含的信息

- 背包容量
- 物品价值
- 物品重量

1.1.2 题目要求内容

在满足背包容量上限的情况下，某种最优解。

1.2 转化为多重背包问题

1.2.1 物品价值 — 硬币面值

1. 转换方式
   - 背包容量 — 硬币总价值/2
   - 物品价值 — 硬币的面值
   - 物品重量 — 硬币的面值
2. 状态 dp[i][j] = j 表示用 i 种硬币可以凑得总价值为 j dp[i][j] ! j 表示用 i 种硬币不可以凑得总价值为 j
3. 状态转移方程

```
dp[i][j] = max(dp[i][j-k[i]*w[i]]+k[i]*w[i],...,dp[i][j-w[i]]+w[i],dp[i-1][j]);
```

1.2.2 物品价值 — 是否可行

1. 转换方式
   - 背包容量 — 硬币总价值/2
   - 物品价值 — 是否可行
   - 物品重量 — 硬币的面值
2. 状态 dp[i][j] 表示使用前 i 个物品是否可以构成总价值 j 初始化: dp[0][0] = 1
3. 状态转移方程

```
dp[i][j] = max(dp[i][j-k[i]*w[i]],...,dp[i][j-w[i]],dp[i-1][j]);
```

1.3 多重背包问题转化为 01 背包问题

1.3.1 p 以下的数字可否由若干不同数字通过组合得到

1. 2^n - 1 n == 7 二进制: 111111 000001、000010、000100、001000、010000、100000
2. k + 2^n-1 < $2^{(n+1)}$-1 n == 7
   - 2^n-1 及以下的数字可以由以下数字表示 000001、000010、000100、001000、010000、100000
   - 大于 2^n-1 的数字 k 加上以上 7 个数字组成的数字可以表示任何大于 2^n-1 小于等于 k+2^n-1 的数字

1.3.2 代码实现

```
//num[i] 表示第i种硬币可以取多少次
//w[j] 表示转后为01背包后 只能取一次的硬币的价值与重量
int totlW = 0;
memset(w,0,sizeof w);
```

```
for (int i = 0 ;i< 6; i ++){
    for(int j=1; j<=num[i]; j<<=1){
        w[totlW++]=j*(i+1);
        num[i]-=j;
    }
    if(num[i]>0)
        w[totlW++]=num[i]*(i+1);
}
```