目录

1 背包问题

1.1 应用场景

给定 n 种物品和一个背包。物品 i 的重量是 w_i，其价值为 v_i，背包的容量为 C。应该如何选择装入背包中的物品，使得装入背包中物品的总价值最大?

1.2 01 背包

1.2.1 01 背包特点

给定 n 种物品和一个背包 (**每个物品只能选取一个**)。物品 i 的重量是 w_i，其价值为 v_i，背包的容量为 C。应该如何选择装入背包中的物品，使得装入背包中物品的总价值最大?

1.2.2 状态

dp[i][j] 表示在只能从 1-i 个物品中选择物品并且背包容量大小为 j 的情况下，所能获得的最大价值。

1. 限制条件
    - 只能选择物品 1 ~ i
    - 选择物品的总重量不能超过 j

1.2.3 状态转移方程

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

1.2.4 实现

1. 初始化

```
int n,m; //n表示物品个数，m表示背包容量
for (int i = 0 ;i<= m ;i++)
    dp[0][i] = 0;
```

2. 01 背包

(a) 通过使用二维数组实现

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

(b) 通过使用一维数组实现

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

1.3 完全背包

1.3.1 完全背包特点

给定 n 种物品和一个背包 (**每个物品选取无限个**)。物品 i 的重量是 w_i，其价值为 v_i，背包的容量为 C。应该如何选择装入背包中的物品，使得装入背包中物品的总价值最大?

1.3.2 状态

dp[i][j] 表示在只能从 1-i 个物品中选择物品并且背包容量大小为 j 的情况下，所能获得的最大价值。

1. 限制条件

- 只能选择物品 1 ~ i
- 选择物品的总重量不能超过 j

1.3.3 状态转移方程

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

1.3.4 实现

1. 初始化

```
int n,m; //n表示物品个数, m表示背包容量
for (int i = 0 ;i<= m ;i++)
    dp[0][i] = 0;
```

2. 完全背包

   (a) 通过使用二维数组实现

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

   (b) 通过使用一维数组实现

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

1.4 多重背包

1.4.1 多重背包特点

给定 n 种物品和一个背包 (**每个物品选取有限个**)。物品 i 的重量是 w_i，其价值为 v_i, 每个物品可以取 k_i 个，背包的容量为 C。应该如何选择装入背包中的物品，使得装入背包中物品的总价值最大?

1.4.2 状态

dp[i][j] 表示在只能从 1-i 个物品中选择物品并且背包容量大小为 j 的情况下，所能获得的最大价值。

1. 限制条件

   - 只能选择物品 1 ~ i
   - 选择物品的总重量不能超过 j

1.4.3 状态转移方程

```
dp[i][j] = max(dp[i-1][j], dp[i-1][j-w[i]] + v[i], dp[i-1][j-w[i]*2] + v[i]*2,...,dp[
```

1.4.4 代码实现

1. 初始化

```
int n,m; //n表示物品个数, m表示背包容量
for (int i = 0 ;i<= m ;i++)
    dp[0][i] = 0;
```

2. 多重背包

   (a) 通过使用二维数组实现

```
for (int i = 1;i <= n ;i++){
    for (int j = 0 ;j<=m ;j++){
        dp[i][j] = dp[i-1][j];
```

```
            for (int z = 1 ; z<=k[i]; z++)
                if (j-w[i]*z>=0)
                    dp[i][j] = max(dp[i][j],dp[i-1][j-z*w[i]]+z*v[i]);
                else
                    break;
        }
    }
```

(b) 通过使用一维数组实现

```
    for (int i = 1;i <= n ;i++){
        for (int j = m ;j>=0 ;j--){
            for (int z = 1; z<=k[i] ;z++)
                if (j-w[i]*z>=0)
                    dp[j] = max(dp[j],dp[j-z*w[i]]+z*v[i]);
                else
                    break;
        }
    }
```

1.5 题目

1.5.1 HDU 2602

1.5.2 POJ 2063

1.5.3 POJ 1787

1.5.4 UVA 674

1.5.5 UVA 147

1.5.6 HDU 4508