背包问题的问法变化

$first$ 输出方案的方案：

> 01背包问题，这里只要求找到一个方案即可；

- 思路
  - 1.开一个数组来记录状态的转移情况。
  - 2.当输出方案的时候，去检查该状态和哪个状态相等。从而确定该状态的转移情况。
- 发散问题：
  - 有没有可能某一步转移时，max的两个比较值是相等的?
    - 举有可能，个例子即可。但是无论哪一种都有办法追溯这一个方案的价值是哪些物品的选择贡献的。
  - 如果是其它类型的背包问题该怎么处理该问题?
    - 也是类比这一个，但是无论哪一种，都要把所有的子问题解给保存下来。以及所有的花费开销保存下来
    - 统一采取，反推的方法。即使出现重复，也是只是说明出现了两种虽然方式不一样，但是。
  - 追溯最优解的选择情况是否可行。
    - 计算得到的子问题的解是真实的。意味着，肯定存在一个选法，指向该结果。本质是，多个物品对结果的贡献。
    - 从低的状态开始看。显然问题的解可以追溯出有一个解的具体方案。
    - 假设现在考察i个状态的子问题的解，现在通过比较迁移的多种状态，可以直到第i个是否选了。从而问题转化为，更小规模的子问题的求解具体方案问题。，若是该迁移的方案可以溯源，那么说明此时考察的方案也可一溯源（考求出一个可行的具体方案。）递推下去得，所有解都可以追溯到一个可行得方案。
    - 只要最终找出来一组方案，它们的贡献就是等于问题的解。一步一步做减法，最终必然等于0；完成追溯。

简单实现如下：

```
#include <bits/stdc++.h>
using namespace std;

int n, m;
const int maxn = 110;
int w[maxn], v[maxn];
int f[maxn][maxn];

int main() //还是要记录之前的选择的果。如果关注过程
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> w[i] >> v[i];
    for (int i = 1; i <= n; i++)
        for (int j = m; j >= 0; j--)
```

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

Second 输出最小字典序的方案：

- 简介

  > 求字典序最小的最优方案。

```
#include <iostream>
#include <algorithm>
using namespace std;

const int maxn = 1010;
int n, m;
int w[maxn], v[maxn];
int f[maxn][maxn];
int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> v[i] >> w[i];
    for (int i = n; i >= 1; i--)
        for (int j = m; j >= 1; j--)
        {
            f[i][j] = f[i + 1][j];
            if (j >= v[i])
                f[i][j] = max(f[i][j], f[i + 1][j - v[i]] + w[i]);
        }
    int i = 1, j = m;
    while (i <= n)
    {
        if (j >= v[i] && f[i][j] == f[i + 1][j - v[i]] + w[i])
        {
            cout << i << ' ';
            j -= v[i];
        }
        i++;
    }
}
```

question

- 虽然已经求出了最优解但是如何保证输出字典序最小
- 这个逆序的迁移和一般正序的差别在哪?
- 几个该明确的点
  - 首先，如果发现第一个可行，i存在一个以它为首，向后发展的最优方案。
  - 如果不可行，说明不存在选择了该点的最优方案。

---

*three*求方案个数

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

int n, m, v, w;
const int maxn = 1010, mod = 109 + 7;
ll f[maxn];
int main()
{
    cin >> n >> m;
    memset(f, 1, sizeof(f));
    for (int i = 1; i <= n; i++)
    {
        cin >> w >> v;
        for (int j = m; j >= w; j--)
            f[j] += f[j - w], f[j] %= mod;
    }
    cout << f[m] << '\n';
}
```

最优方案个数问题

- 一段错误的代码

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

int n, m;
const int maxn = 1010, mod = 1e9 + 7;
ll f[maxn][maxn], s[maxn][maxn];
int w[maxn], v[maxn];

int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> v[i] >> w[i];
    memset(s, 1LL, sizeof(s));//
    for (int i = 1; i <= n; i++)
```

```
        for (int j = m; j >= 0; j--)
        {
            f[i][j] = f[i - 1][j];
            if (j >= v[i])
                f[i][j] = max(f[i][j], f[i - 1][j - v[i]] + w[i]);
            if (f[i - 1][j] == f[i][j])
                s[i][j] = s[i - 1][j];
            if (j >= v[i] && f[i][j] == f[i - 1][j - v[i]] + w[i])
                s[i][j] += s[i - 1][j - v[i]];
            s[i][j] %= mod;
        }
    cout << s[n][m] << '\n';
}
```

- 函数的使用出现问题；memset。细节把握不够，对函数认识不清晰，吃在出事情。

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

int n, m;
const int maxn = 1010, mod = 1e9 + 7;
ll f[maxn][maxn], s[maxn][maxn];
int w[maxn], v[maxn];

int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> v[i] >> w[i];
    for (int i = 0; i <= m; i++)
        s[0][i] = 1;
        for (int i = 1; i <= n; i++)
            for (int j = m; j >= 0; j--)
            {
                f[i][j] = f[i - 1][j];
                if (j >= v[i])
                    f[i][j] = max(f[i][j], f[i - 1][j - v[i]] + w[i]);
                if (f[i - 1][j] == f[i][j])
                    s[i][j] = s[i - 1][j];
                if (j >= v[i] && f[i][j] == f[i - 1][j - v[i]] + w[i])
                    s[i][j] += s[i - 1][j - v[i]];
                s[i][j] %= mod;
            }
    cout << s[n][m] << '\n';
}
```

简化版一维数组;

```
#include <bits/stdc++.h>
using namespace std;

const int maxn = 1010;
int n, m, f[maxn], sum[maxn];
int mod = 1e9 + 7;

int main()
{
    cin >> n >> m;
    for (int i = 0; i <= m; i++)
        sum[i] = 1;
    for (int i = 1; i <= n; i++)
    {
        int v, w;
        cin >> v >> w;
        for (int j = m; j >=v ; j--)
        {
            int value = f[j - v] + w;
            if(f[j]<value)
            {
                f[j] = value;
                sum[j] = sum[j - v];
            }
            else if(f[j]==value)
                sum[j] = (sum[j] + sum[j - v]) % mod;
        }
    }
    cout << sum[m] << '\n';
}
```

- 综合上一系列背包问题的结论：
    - 1.每一个子问题的解必然处直接或者间接的比较了所有的可行方案。
    - 2.比较方案，不存在重复，每一次都是等价于比较两种不一样的集合。