背包问题模板

01背包问题

其中n为物品个数，m为背包承受最大重量。v为物品i体积，w为物品价值。

```
#include<iostream>
#include<algorithm>

using namespace std;

const int V = 1010;

int n, m;
int dp[V];

int main() {
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int v, w;
        cin >> v >> w;
        for (int j = m; j >= v; j--)
            dp[j] = max(dp[j], dp[j - v] + w);
    }
    cout << dp[m] << endl;
    return 0;
}
```

完全背包问题

其中v为物品i体积，w为物品价值

```
#include<iostream>
#include<algorithm>

using namespace std;

const int V = 1010;

int n, m;
int dp[V];

int main() {
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int v, w;
        cin >> v >> w;
        for (int j = v; j <= m; j++)
            dp[j] = max(dp[j], dp[j - v] + w);
    }
    cout << dp[m] << endl;
    return 0;
}
```

多重背包问题

其中n为物品个数，m为背包承受最大重量。v为物品i体积，w为物品价值，s为物品个数

朴素算法

```
#include<iostream>
#include<algorithm>

using namespace std;

const int N = 110;

int n, m;
int dp[N];

int main() {
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int v, w, s;
        cin >> v >> w >> s;
        for (int j = m; j >= v; j--)
            for (int k = 0; k <= s && k*v <= j; k++)
                dp[j] = max(dp[j], dp[j - k * v] + k * w);
    }
    cout << dp[m] << endl;
    return 0;
}
```

二进制优化

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

const int V = 2020;

struct Good
{
    int v, w;
};

int n, m;
int dp[V];

int main() {
    vector<Good> goods;
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int v, w, s;
        cin >> v >> w >> s;
        for (int k = 1; k <= s; k <<= 1) {
            s -= k;
```

```
            goods.push_back({ k*v,k*w });
        }
        if (s > 0)
            goods.push_back({ s*v,s*w });
    }
    for (auto good : goods)
        for (int j = m; j >= good.v; j--)
            dp[j] = max(dp[j], dp[j - good.v] + good.w);
    cout << dp[m] << endl;
    return 0;
}
```

单调队列优化

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

const int V =  20010;

int n, m;
int dp[V], g[V], que[V];

int main() {
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int v, w, s;
        cin >> v >> w >> s;
        memcpy(g, dp, sizeof(dp));
        //枚举余数
        for (int j = 0; j < v; j++) {
            //单调队列
            int h = 0, t = -1;
            for (int k = j; k <= m; k += v) {
                dp[k] = g[k];
                if (h <= t && que[h] < k - s * v) h++;
                if (h <= t) dp[k] = max(dp[k], g[que[h]] + (k - que[h]) / v *
w);
                if (h <= t && g[que[t]] - (que[t] - j) / v * w <= g[k] - (k - j)
/ v * w) t--;
                que[++t] = k;
            }
        }
    }
    cout << dp[m];
    return 0;
}
```

混合背包问题

其中n为物品个数，m为背包承受最大重量。v为物品i体积，w为物品价值，`s = -1` 表示物品只能使用一次;`s = 0` 表示物品能使用无限次；`s > 0`表示物品可以使用 s 次

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

const int V = 1010;

struct  Good
{
    int v, w;
    bool type;
};

int n, m;
int dp[V];

int main() {
    vector<Good> goods;
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int v, w, s;
        cin >> v >> w >> s;
        if (s == -1) {
            goods.push_back({ v,w,true });
        }
        else if (s > 0) {
            for (int k = 1; k <= s; k <<= 1) {
                s -= k;
                goods.push_back({ k*v,k*w,true });
            }
            if (s > 0)goods.push_back({ s*v,s*w,true });
        }
        else {
            goods.push_back({ v,w,false });
        }
    }
    for (auto good : goods) {
        if (good.type)
            for (int j = m; j >= good.v; j--)
                dp[j] = max(dp[j], dp[j - good.v] + good.w);
        else
            for (int j = good.v; j <= m; j++)
                dp[j] = max(dp[j], dp[j - good.v] + good.w);
    }
    cout << dp[m];
    return 0;
}
```

二维费用背包问题

其中N为物品个数，C为背包承受最大体积，M为背包承受最大重量。c为物品i体积，m为物品重量,w为物品价值。求在总体积不超过背包容量，总重量不超过背包可承受的最大重量下，价值总和最大。物品只用一次。

```
#include<iostream>
#include<algorithm>

using namespace std;

const int V = 110;

int N, C, M;
int dp[V][V];

int main() {
    cin >> N >> C >> M;
    for (int i = 0; i < N; i++) {
        int c, m, w;
        cin >> c >> m >> w;
        for (int j = C; j >= c; j--)
            for (int k = M; k >= m; k--)
                dp[j][k] = max(dp[j][k], dp[j - c][k - m] + w);
    }
    cout << dp[C][M];
    return 0;
}
```

分组背包问题

其中n为物品个数，m为背包承受最大重量。v为物品i体积，w为物品价值每组物品有若干个，同一组内的物品最多只能选一个。每件物品的体积是 vij，价值是 wij，其中 i 是组号，j 是组内编号。

```
#include<iostream>
#include<algorithm>

using namespace std;

const int V = 110;

int n, m;
int dp[V], v[V],w[V];

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

```
    cout << dp[m];
    return 0;
}
```

有依赖问题的背包

其中n为物品个数，m为背包承受最大重量。v为物品i体积，w为物品价值，p为其依赖的父节点的编号。`p = -1`表示为根节点。h存储节点编号，e存储其指向的节点，ne指向其下一个指向的节点，是邻接表的使用。每件物品只能使用一次。

```
#include<iostream>
#include<algorithm>

using namespace std;

const int N = 110;

int n, m;
int h[N], e[N], ne[N], index;
int v[N], w[N];
int dp[N][N];
//(x-->y)
void add(int x, int y) {
    e[++index] = y, ne[index] = h[x], h[x] = index;
}
void dfs(int u) {
    for (int i = h[u]; i; i = ne[i]) {
        int son = e[i];
        dfs(son);
        for (int j = m - v[u]; j >= 0; j--) {
            for (int k = 0; k <= j; k++)//分组选择其中一个
                dp[u][j] = max(dp[u][j], dp[u][j - k] + dp[son][k]);
        }
    }
    for (int j = m; j >= v[u]; j--) dp[u][j] = dp[u][j - v[u]] + w[u];
    for (int j = 0; j < v[u]; j++) dp[u][j] = 0;
}
int main() {
    cin >> n >> m;
    int root, p;
    for (int i = 1; i <= n; i++) {
        cin >> v[i] >> w[i] >> p;
        if (p == -1) root = i;
        else add(p, i);
    }
    dfs(root);
    cout << dp[root][m] << endl;
    return 0;
}
```

背包问题求方案数

其中n为物品个数，m为背包承受最大重量。v为物品i体积，w为物品价值。求最优选法的方案数，答案模1000000007的结果。每件物品只能使用一次。

方法1：恰好

```
#include<iostream>
#include<algorithm>

using namespace std;

const int N = 1010, mod = 1e9 + 7, INF = 1e9;

int n, m;
int dp[N], g[N];

int main() {
    cin >> n >> m;
    for (int i = 1; i <= m; i++) dp[i] = -INF;
    g[0] = 1;//什么都不选为一种
    for (int i = 0; i < n; i++) {
        int v, w;
        cin >> v >> w;
        for (int j = m; j >= v; j--) {
            int t = max(dp[j], dp[j - v] + w);
            int s = 0;
            if (t == dp[j]) s += g[j];
            if (t == dp[j - v] + w) s += g[j - v];
            if (s >= mod) s -= mod;
            g[j] = s;
            dp[j] = t;
        }
    }
    int maxw = 0;
    for (int i = 0; i <= m; i++) maxw = max(maxw, dp[i]);
    int res = 0;
    for (int i = 0; i <= m; i++) {
        if (dp[i] == maxw) {
            res += g[i];
            if (res >= mod) res -= mod;
        }
    }
    cout << res;
    return 0;
}
```

方法2：`g[m]直接为最优解`

```
#include<iostream>
#include<algorithm>

using namespace std;

const int N = 1010, mod = 1e9 + 7;

int n, m;
```

```
int dp[N], g[N];

int main() {
    cin >> n >> m;
    for (int i = 0; i <= m; i++) g[i] = 1;//什么都不选为一种
    for (int i = 0; i < n; i++) {
        int v, w;
        cin >> v >> w;
        for (int j = m; j >= v; j--) {
            int t = max(dp[j], dp[j - v] + w);
            int s = 0;
            if (t == dp[j]) s += g[j];
            if (t == dp[j - v] + w) s += g[j - v];
            if (s >= mod) s -= mod;
            g[j] = s;
            dp[j] = t;
        }
    }
    cout << g[m];
    return 0;
}
```

求背包问题具体方案

01背包问题，求字典序最小具体方案。

```
#include<iostream>
#include<algorithm>

using namespace std;

const int N = 1010;

int n, m;
int dp[N][N], v[N], w[N];

int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++) cin >> v[i] >> w[i];
    for (int i = n; i > 0; i--) {
        for (int j = m; j >= 0; j--) {
            dp[i][j] = dp[i + 1][j];
            if (j >= v[i])
                dp[i][j] = max(dp[i][j], dp[i + 1][j - v[i]] + w[i]);
        }
    }
    int op = m;
    for (int i = 1; i <= n; i++) {
        if (op - v[i] >= 0 && dp[i + 1][op - v[i]] + w[i] == dp[i][op])
{
            cout << i << " ";
            op -= v[i];
        }
    }
    return 0;
```

```
}
```

参考资料：

- ACWing以上的题目均可以在此网站“题库分类:背包九讲”中找到。