例题 开心的金明

题目链接

https://nanti.jisuanke.com/t/284

所用知识

01背包

题目讲解

- 将价格和重要度的乘积作为价值
- 带入01背包模板

代码实现

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>

using namespace std;

// 复习一下结构体的使用方式
typedef struct arr {
    int w, data;
} arr;

int main() {
    int n, m, dp[30005] = {0}; // 没有要求将背包恰好装满 初始化为零即可
    arr num[30005];
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        cin >> num[i].w >> num[i].data;
        num[i].data *= num[i].w; // 将价格和重要度的乘积作为价值
    }

    // 01背包模板
    for (int i = 0; i < m; i++) {
        for (int j = n; j >= num[i].w; j--) {
            dp[j] = max(dp[j], dp[j - num[i].w] + num[i].data);
```

```
        }
    }
    cout << dp[n] << endl;
    return 0;
}
```

练习题 HDU1171

题目链接

http://acm.hdu.edu.cn/showproblem.php?pid=1171

所学知识

01背包

题目讲解

01背包变形

保证a组多 应求b组最大能放的，即变成对于b来说的01背包了，总价值为`sum_half = sum / 2`，即b组背包最大容量为sum_half

状态方程为

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

b组为`dp[sum_half]`，则a组为`sum - dp[sum_half]`

代码实现

```
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

const int MAX_N = 50010;
const int MAX_M = 300005;

int v[MAX_N] = {0};
int dp[MAX_M] = {0};

int main() {
    int n;
```

```
    while (cin >> n && n > 0) {
        int sum = 0, x, y, cnt = 0;
        memset(dp, 0, sizeof(dp));
        for (int i = 0; i < n; i++) {
            cin >> x >> y;
            sum += x * y;
            while (y--) v[cnt++] = x;
        }
        int sum_half = sum / 2;
        for (int i = 0; i < cnt; i++) {
            for (int j = sum_half; j >= v[i]; j--) {
                dp[j] = max(dp[j], dp[j - v[i]] + v[i]);
            }
        }
        cout << sum - dp[sum_half] << " " << dp[sum_half] <<
endl;
    }
    return 0;
}
```

例题 P1616 疯狂的采药

题目链接

https://www.luogu.org/problemnew/show/P1616

所用知识

完全背包

题目讲解

直接套模板

代码实现

```
#include <iostream>
#include <cmath>
#include <algorithm>
using namespace std;
int T, M, dp[100005], t[100005], w[100005];
int main() {
```

```
    cin >> T >> M;
    for (int i = 0; i < M; ++i) cin >> t[i] >> w[i];
    for (int i = 0; i < M; ++i) {
        for (int j = t[i]; j <= T; ++j) { // 因为无限次 无论dp[j-c[i]]选没选过c[i]都可以
            dp[j] = max(dp[j], dp[j - t[i]] +  w[i]);
        }
    }
    cout << dp[T] << endl;
    return 0;
}
```

练习题 HDU1114

题目链接

http://acm.hdu.edu.cn/showproblem.php?pid=1114

所学知识

完全背包

题目讲解

完全背包裸题，直接套模板

代码实现

```
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

const int INF = 0x3f3f3f3f;
const int MAX_N = 10010;
int dp[MAX_N] = {0};

int main() {
    int t;
    cin >> t;
    while (t--) {
        memset(dp, INF, sizeof(dp));
        dp[0] = 0;
```

```
        int n, w, m, e, f, V;
        cin >> e >> f;
        V = f - e;
        cin >> n;
        for (int i = 0; i < n; i++) {
            cin >> w >> m;
            for (int j = m; j <= V; j++) {
                dp[j] = min(dp[j], dp[j - m] + w);
            }
        }
        if (dp[V] < INF) {
            cout << "The minimum amount of money in the
piggy-bank is " << dp[V] << "." << endl;
        } else {
            cout << "This is impossible." << endl;
        }
    }
    return 0;
}
```

例题 HDU2191

题目链接

http://acm.hdu.edu.cn/showproblem.php?pid=2191

所学知识

多重背包、二进制优化、单调队列优化

题目讲解

模板题，直接套模板即可

分别使用三种方法解题：

1. 三层for，01背包 **`时间复杂度 O(n*m*cnt)`**
2. 二进制优化，01背包，直接套模板 **`时间复杂度 O(n*m*log(cnt))`**
3. 单调队列优化，01背包，直接套模板 **`时间复杂度 O(n*m)`**

代码实现一（二进制优化）

```
#include <iostream>
#include <cstdio>
```

```cpp

#define MAX_N 10000

using namespace std;

typedef struct array {
    int c, w;
} array;

int main() {
    int t;
    cin >> t;
    while (t--) {
        int dp[MAX_N + 5] = {0};
        int n, m, cnt = 0;
        array arr[MAX_N + 5];
        cin >> n >> m;
        for (int i = 0; i < m; i++) {
            int c, w, k;
            cin >> c >> w >> k;
            for (int j = 1; j <= k; j <= 1) {
                arr[cnt].c = j * c;
                arr[cnt++].w = j * w;
                k -= j;
            }
            if (k > 0) {
                arr[cnt].c = k * c;
                arr[cnt++].w = k * w;
            }
        }
        for (int i = 0; i < cnt; i++) {
            for (int j = n; j >= arr[i].c; j--) {
                dp[j] = max(dp[j], dp[j - arr[i].c] +
arr[i].w);
            }
        }
        cout << dp[n] << endl;
    }
    return 0;
}
```

代码实现二（单调队列优化）

https://blog.csdn.net/flyinghearts/article/details/5898183

```cpp

```

```cpp
#include <iostream>
#include <cstdio>

using namespace std;

#define MAX_V 100005
void pack(int *dp, int V, int v, int n, int w) {
    if (n == 0 || v == 0) return;
    if (n == 1) {
        for (int i = V; i >= v; --i) {
            dp[i] = max(dp[i], dp[i - v] + w);
        }
        return;
    }
    if (n * v >= V - v + 1) {
        for (int i = v; i <= V; ++i) {
            dp[i] = max(dp[i], dp[i - v] + w);
        }
        return;
    }

    int va[MAX_V], vb[MAX_V];
    for (int j = 0; j < v; ++j) {
        int pb = 0, pe = -1;
        int qb = 0, qe = -1;
        for (int k = j, i = 0; k <= V; k += v, ++i) {
            if (pe == pb + n) {
                if (va[pb] == vb[qb]) ++qb;
                ++pb;
            }
            int temp = dp[k] - i * w;
            va[++pe] = temp;
            while (qe >= qb && vb[qe] < temp) --qe;
            vb[++qe] = temp;
            dp[k] = vb[qb] + i * w;
        }
    }
    return ;
}

int main() {
    int t;
    cin >> t;
    while (t--) {
        int n, m, dp[MAX_V] = {0};
```

```
        cin >> n >> m;
        for (int i = 0; i < m; i++) {
            int c, w, k;
            cin >> c >> w >> k;
            pack(dp, n, c, k, w);
        }
        cout << dp[n] << endl;
    }
    return 0;
}
```

例题 金明的预算方案

题目链接

https://nanti.jisuanke.com/t/11589

所用知识

有依赖的背包问题、01背包

题目讲解

一个物品最多有两个附件，不买主件便不能买附件，即可以采用分组思想，可分为五种情况：

1. 不买主件也不买其附件
2. 只买主件
3. 买主件和附件A
4. 买主件和附件B
5. 买主件和附件A和附件B

最后再使用01背包求解即可。

代码实现

```
#include <iostream>
#include <cstdio>

using namespace std;

// cnt记录该主件和附件共有几种组合
```

```cpp
typedef struct data {
    int cnt;
    int c[4];
    int w[4];
} data;

int main() {
    data num[70];
    for (int i = 0; i < 70; i++) num[i].cnt = 0;
    int dp[32005] = {0};
    int n, m, a, b, c, ind = 0;
    cin >> n >> m;
    // 将主件和附件的几种组合情况分别储存
    for (int i = 0; i < m; i++) {
        cin >> a >> b >> c;
        if (c == 0) {
            num[i].cnt = 1;
            num[i].c[0] = a;
            num[i].w[0] = a * b;
            continue;
        }
        int temp = num[c-1].cnt;
        num[c-1].c[temp] = num[c-1].c[0] + a;
        num[c-1].w[temp] = num[c-1].w[0] + a * b;
        num[c-1].cnt++;
        temp++;
        if (temp == 3) { // temp等于3代表有一个主件两个配件，把最后一种情况存入分组
            num[c-1].c[temp] = num[c-1].c[1] + a;
            num[c-1].w[temp] = num[c-1].w[1] + a * b;
            num[c-1].cnt++;
        }
    }

    for (int i = 0; i < m; i++) {
        if (num[i].cnt == 0) continue;
        for (int l = n; l >= 0; l--) {
            for (int j = 0; j <= num[i].cnt - 1; j++) {
                if (l - num[i].c[j] < 0) continue;
                dp[l] = max(dp[l], dp[l - num[i].c[j]] + num[i].w[j]);
            }
        }
    }
    cout << dp[n] << endl;
```

```
    return 0;
}
```

例题 P1164 小A点菜

题目链接

https://www.luogu.org/problemnew/show/P1164

所学知识

01背包变型

题目讲解

只需将`dp[0] = 1 其他位置为0`，之后按01背包处理

代码实现

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>

using namespace std;

int main() {
    int n, m;
    int num[105] = {0}, dp[10005] = {0};
    dp[0] = 1; // 剩余的钱买完一道菜后刚好用完 所以剩余的这些钱只有一种买法
    cin >> n >> m;
    for (int i = 0; i < n; i++) cin >> num[i];
    for (int i = 0; i < n; i++) {
        for (int j = m; j >= num[i]; j--) {
            dp[j] += dp[j - num[i]]; // 现在的花费（方法数） += 不点这个菜的时候的花费（方法数）
        }
    }
    cout << dp[m] << endl;
    return 0;
}
```

HDU1203

http://acm.hdu.edu.cn/showproblem.php?pid=1203

题目大意

一共有n万美元，有m个学校，他要在m个学校中选择若干学校投递申请，每个学校有各自的申请费用和得到这个学校offer的概率，求他至少可以收到一个offer的概率

所学知识

01背包

题目讲解

要求至少可以收到一个offer的概率，可以先求收不到offer的最小概率，再用1减去其概率即可

动态转移方程

v该学校的费用 p得不到该学校offer的概率

`dp[i] = min(dp[i], dp[i] - v) * p`

代码实现

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

const int MAX_N = 10010;
double dp[MAX_N] = {0};
int v[MAX_N] = {0};
double w[MAX_N] = {0};

int main() {
    int n, m;
    while (scanf("%d %d", &n, &m) != EOF && (n || m)) {
        for (int i = 0; i <= n; i++) dp[i] = 1;
        for (int i = 0; i < m; i++) {
            cin >> v[i] >> w[i];
            w[i] = 1 - w[i];
        }
        for (int i = 0; i < m; i++) {
```

```
            for (int j = n; j >= v[i]; j--) {
                dp[j] = min(dp[j], dp[j - v[i]] * w[i]);
            }
        }
        printf("%.1lf%%\n", (1.0 - dp[n]) * 100.0);
    }
    return 0;
}
```

HDU1284

http://acm.hdu.edu.cn/showproblem.php?pid=1284

题目大意

在一个国家仅有1分，2分，3分硬币，将钱N兑换成硬币有多少种兑法

所学知识

完全背包

题目讲解

`dp[0] = 1`

代码实现

```
#include <iostream>
#include <cstdio>

using namespace std;

const int MAX_N = 40000;

int dp[MAX_N] = {0};

void init() {
    int num[5] = {1, 2, 3};
    dp[0] = 1;
    for (int i = 0; i < 3; i++) {
        for (int j = num[i]; j < MAX_N; j++) {
            dp[j] += dp[j - num[i]];
        }
    }
```

```
    return ;
}

int main() {
    init();
    int n;
    while (cin >> n) {
        cout << dp[n] << endl;
    }
    return 0;
}
```

HDU2159

题目链接

http://acm.hdu.edu.cn/showproblem.php?pid=2159

所学知识

完全背包

题目讲解

- 以忍耐度为背包，经验为dp值，在dp时记录杀怪的次数
- 再扫一边背包，看是否有经验值达到要求的

代码实现

```
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

const int MAX_N = 110;
const int INF = 0x3f3f3f3f;
int dp[MAX_N] = {0}, cnt[MAX_N] = {0};

int main() {
    int n, m, k, s, a, b;
    while (cin >> n >> m >> k >> s) {
        memset(dp, -INF - 1, sizeof(dp));
        memset(cnt, 0, sizeof(cnt));
```

```
        dp[0] = 0;
        for (int i = 0; i < k; i++) {
            cin >> a >> b;
            for (int j = b; j <= m; j++) {
                if (cnt[j - cnt[b]] + 1 > s) break;
                if (dp[j] < dp[j - b] + a) {
                    dp[j] = dp[j - b] + a;
                    cnt[j] = cnt[j - b] + 1;
                }
            }
        }
        int ind = 1;
        while (dp[ind] < n && ind <= m) ++ind;
        printf("%d\n", ind > m ? -1 : m - ind);
    }
    return 0;
}
```

P1057

题目大意

有n个人围成一圈（分别为1...n号），从1号开始传球，传m次，可以传给自己的左边或右边，最后在传回1号有多少种方法

所学知识

动态规划

题目讲解

状态转移方程

`dp[i][j] = dp[i + 1][j - 1] + dp[i - 1][j - 1]`

- `dp[i][j]`表示传 j 次传到 i 号手中的方法次数
- 因为成环，所以在 i 为 1 或 n 时特殊处理

代码实现

```
#include <iostream>
#include <cstdio>

using namespace std;
```

```cpp

int main() {
    int n, m;
    int dp[35][35] = {0};
    dp[1][0] = 1;
    cin >> n >> m;
    for (int i = 1; i <= m; i++) {
        dp[1][i] = dp[n][i - 1] + dp[2][i - 1];
        for (int j = 2; j <= n - 1; j++) {
            dp[j][i] = dp[j + 1][i - 1] + dp[j - 1][i - 1];
        }
        dp[n][i] = dp[1][i - 1] + dp[n - 1][i - 1];
    }
    cout << dp[1][m] << endl;
    return 0;
}
```

P1002

题目大意

一个过河卒从 A 点(0,0)，走到 B 点(n,m)(n, m为不超过20的整数)，每次过河卒只能向下或向右走一步，不能走到马的位置或马所能走到的位置，求能走到 B 点的路径条数

所学知识

动态规划，方向数组

题目讲解

状态转移方程

`dp[i][j] = dp[i - 1][j] + dp[i][j - 1]`

`dp[i][j]`表示从（0，0）点到（i，j）点的路径条数

- dir数组表示马能走的8个方向
- `map_arr[y][x]` ，x 表示横坐标，y 表示纵坐标。dp数组同理

代码实现

```cpp
#include <iostream>
#include <cstdio>

```

```cpp
using namespace std;

int map_arr[25][25] = {0};
int b_x = 0, b_y = 0, m_x = 0, m_y = 0;
int dir[8][2] = {-1, -2, -2, -1, -1, 2, -2, 1, 1, -2, 2, -1, 1, 2, 2, 1};
long long int dp[25][25] = {0};

// 初始化马的位置和马能走到的位置
void init_map_arr() {
    map_arr[m_y][m_x] = 1;
    for (int i = 0; i < 8; i++) {
        int x = m_x + dir[i][0], y = m_y + dir[i][1];
        if (x < 1 || x > 21 || y < 1 || y > 21) continue;
        map_arr[y][x] = 1;
    }
    return ;
}

int main() {
    cin >> b_x >> b_y >> m_x >> m_y;
    ++b_x, ++b_y, ++m_x, ++m_y;
    init_map_arr();
    dp[1][1] = 1;
    for (int i = 1; i <= b_y; i++) {
        for (int j = 1; j <= b_x; j++) {
            if (i == 1 && j == 1) continue;
            if (map_arr[i][j]) continue;
            dp[i][j] = dp[i - 1][j] + dp[i][j - 1];
        }
    }
    cout << dp[b_y][b_x] << endl;
    return 0;
}
```

P1020 导弹拦截

所学知识

最长上升序列

问题解析

- 计算这套系统最多能拦截多少导弹?
  - 直接使用模板跑出最长不上升子序列长度
- 拦截所有导弹最少要配备多少套这种导弹拦截系统?
  - 通过Dilworth定理可以知道**最少的下降序列个数就等于整个序列最长上升子序列的长度**
  - 使用模板跑出最长上升子序列长度

代码

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;

#define MAX_N 100010
int len[MAX_N] = {0};
int arr[MAX_N] = {0};

void init(int x) {
    memset(len, x, sizeof(len));
    len[1] = arr[0];
    return ;
}

bool check(int flag, int x, int y) {
    return (flag ? x >= y : x < y);
}

int find(int *num, int n, int key, int flag) {
    int l = 0, r = n, mid;
    while (l < r) {
        mid = (l + r) / 2;
        if (check(flag, num[mid], key)) r = mid;
        else l = mid + 1;
    }
    return l;
}

int main() {
    int n = 0, ans = 1, sum = 1;
    while (scanf("%d", &arr[n]) != EOF) ++n;
    init(0x3f);
    for (int i = 1; i < n; i++) {
        if (len[ans] >= arr[i]) len[++ans] = arr[i];
```

```
        else {
            len[find(len, ans + 1, arr[i], 0)] = arr[i];
        }
    }
    init(-1);
    for (int i = 1; i < n; i++) {
        if (len[sum] < arr[i]) len[++sum] = arr[i];
        else {
            len[find(len, sum + 1, arr[i], 1)] = arr[i];
        }
    }
    cout << ans << endl << sum << endl;
    return 0;
}
```