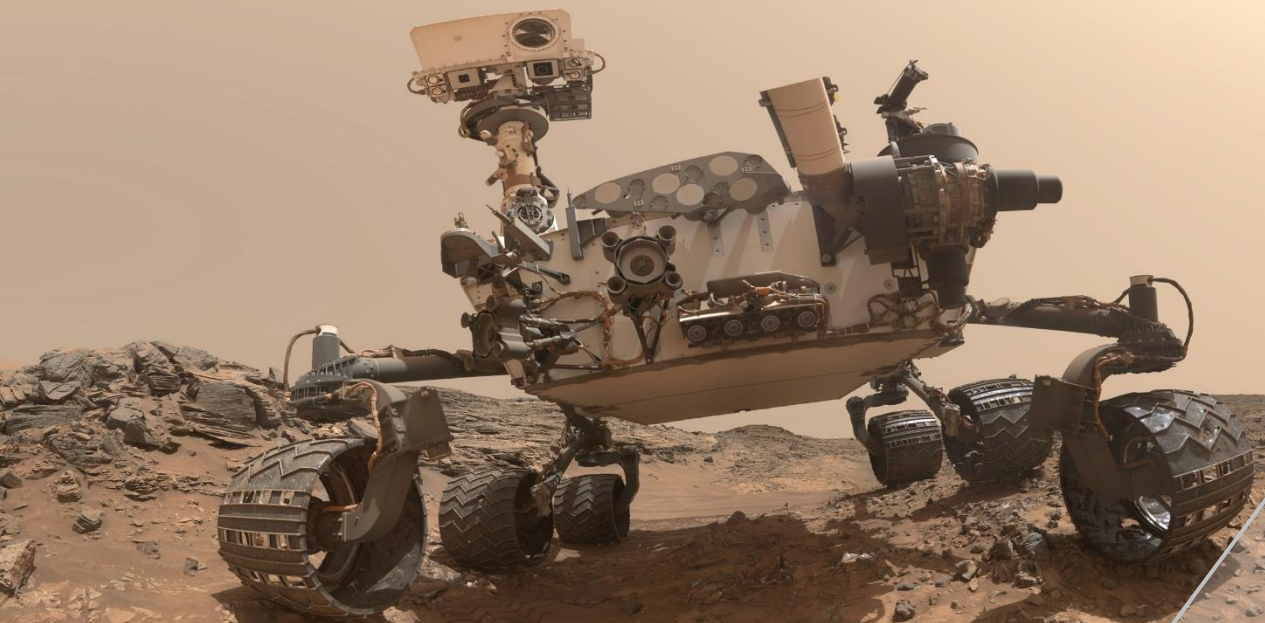
CS101



信奥算法

课件下载地址：
http://pan.baidu.com/s/1o885tz0

作业网站：
http://120.132.18.213:8080/thrall-web/main#home

01背包问题
动态规划

大盗：m种金砖，01限制

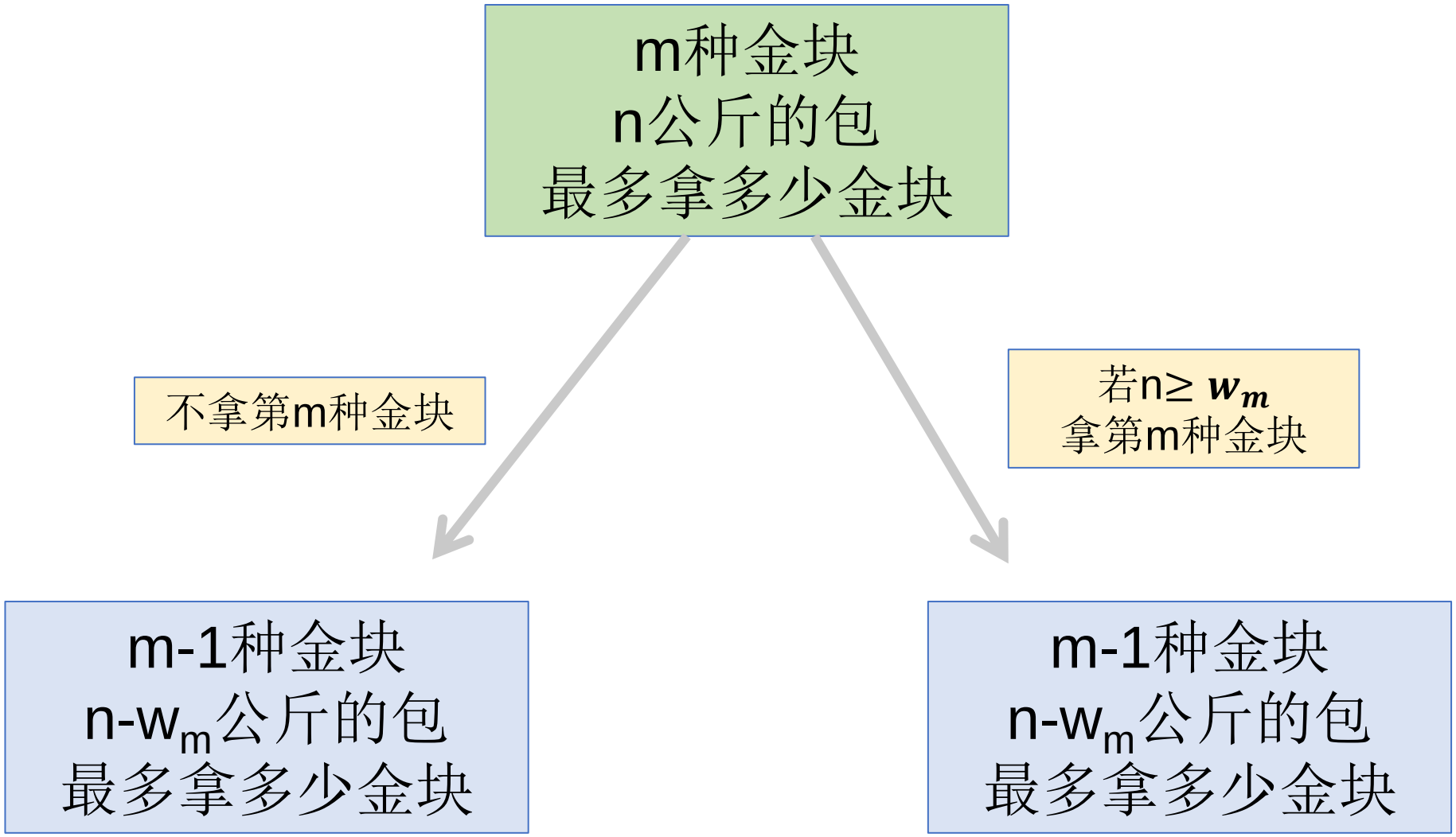
盗贼去抢银行，他带的包只能装n公斤的金块。银行有m种规格的金块，分别重$w_1,w_2,\ldots,w_m$公斤，数量都**只有1块**，金块无法分割。输入整数n,m, $w_1,w_2,\ldots,w_m$, 输出**最多带走多少公斤**

暴力枚举算法

对于第i种金块，两个可能决策：
不拿第i种金块; 拿第i种金块

枚举算法：所有可能性共2^m种可能

原问题 分解成 子问题


m种金块
n公斤的包
最多拿多少金块
不拿第m种金块
若n≥ w_m
拿第m种金块
m-1种金块
n-w_m公斤的包
最多拿多少金块
m-1种金块
n-w_m公斤的包
最多拿多少金块

大盗：m种金砖，01限制

递推算法

盗贼去抢银行，他带的包只能装n公斤的金块。银行有m种规格的金块，分别重$w_1,w_2,\ldots,w_m$公斤，数量都只有1块，金块无法分割。
输入整数n,m, $w_1,w_2,\ldots,w_m$, 输出**最多带走多少公斤**

二维递推算法：f[i][j]表示只装前i种金块，用j公斤以内的包，最多拿多少金块

当i为0时$f[0][j] = 0$	当j为0时$f[i][0] = 0$

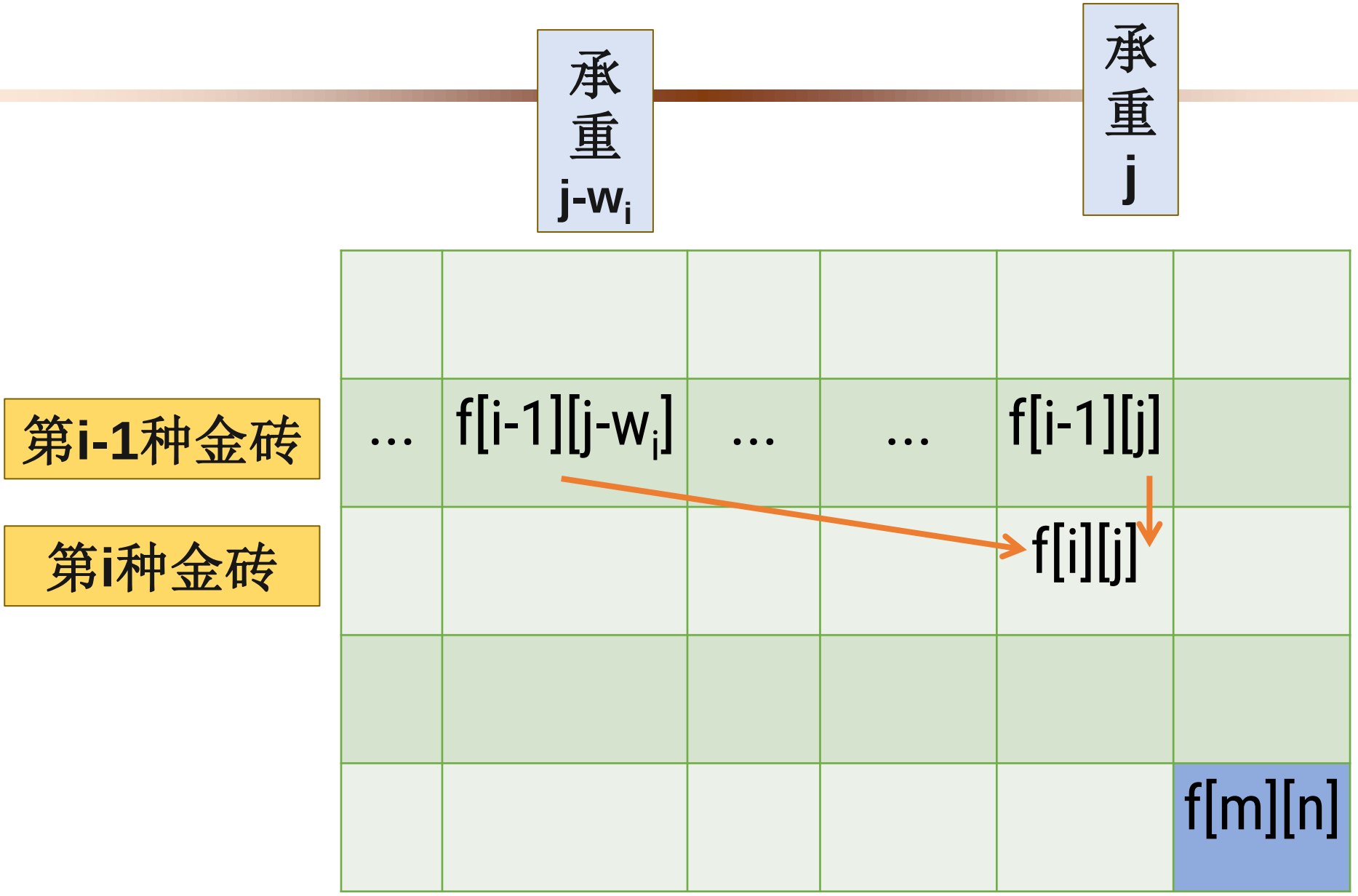
大盗：m种金砖，01限制

二维递推算法：
f[i][j]表示只装前i种金块，
用j公斤以内的包，最多拿多少金块

当i为0时$f[0][j] = 0$	当j为0时$f[i][0] = 0$

$$f[i][j] = max(f[i-1][j], f[i-1][j-w_i] + w_i \mid j \geq w_i)$$

若j>=w_i
可以拿第i件


承重 j-w_i
承重 j
第i-1种金砖
第i种金砖
… f[i-1][j-w_i] … … f[i-1][j]
f[i][j]
f[m][n]

大盗m种金砖01限制- 代码

```cpp
#include <iostream>
#define M 505
#define N 2005
using namespace std;
int n,m,w[M],f[M][N];
int main(){
    cin>>n>>m;
    for(int i=1;i<=m;i++) cin>>w[i];
    //初始化清零，已经默认执行了
    for(int i=1;i<=m;i++)   //循环查看每种金块i
        for(int j=0;j<=n;j++) {  //循环查看背包剩余承重j
            if(j<w[i])   //金块i太重，无法放入
                f[i][j]=f[i-1][j];
            else          //比较两种决策：金块i可以放，或者不放
                f[i][j]=max(f[i-1][j],f[i-1][j-w[i]]+w[i]);
        }
    cout<<f[m][n]<<endl;
    return 0;
}
```

讨论题

为什么递推算法比暴力枚举算法快？

递推时储存已计算完的结果

暴力枚举时重复计算了子问题

计算次数约 $O(n*m)$

计算次数约 $O(2^m)$

01背包问题 - 经典版本

3斤,￥5000，

1斤,￥3000，

2斤,￥3500，

怎么选？

01背包：小偷

小偷来你家，他带的包只能装n公斤的财物。你家有m种财务，分别重$w_1,w_2,\ldots,w_m$公斤，价值分别为$v_1,v_2,\ldots,v_m$
输入整数n,m, $w_1,w_2,\ldots,w_m$, $v_1,v_2,\ldots,v_m$
输出带走**最大的总价值**是多少

费用　消耗载重量

收益　赚取价值

01背包：小偷

小偷来你家，他带的包只能装n公斤的财物。你家有m种财务，分别重$w_1,w_2,\ldots,w_m$公斤，价值分别为$v_1,v_2,\ldots,v_m$
输入整数n,m, $w_1,w_2,\ldots,w_m$, $v_1,v_2,\ldots,v_m$
输出带走**最大的总价值**是多少

对于第i种财物，两个可能决策：
不拿第i种财物; 拿第i种财物

f[i][j]表示只装前i种财物，用j公斤以内的包，最大总价值能拿多少

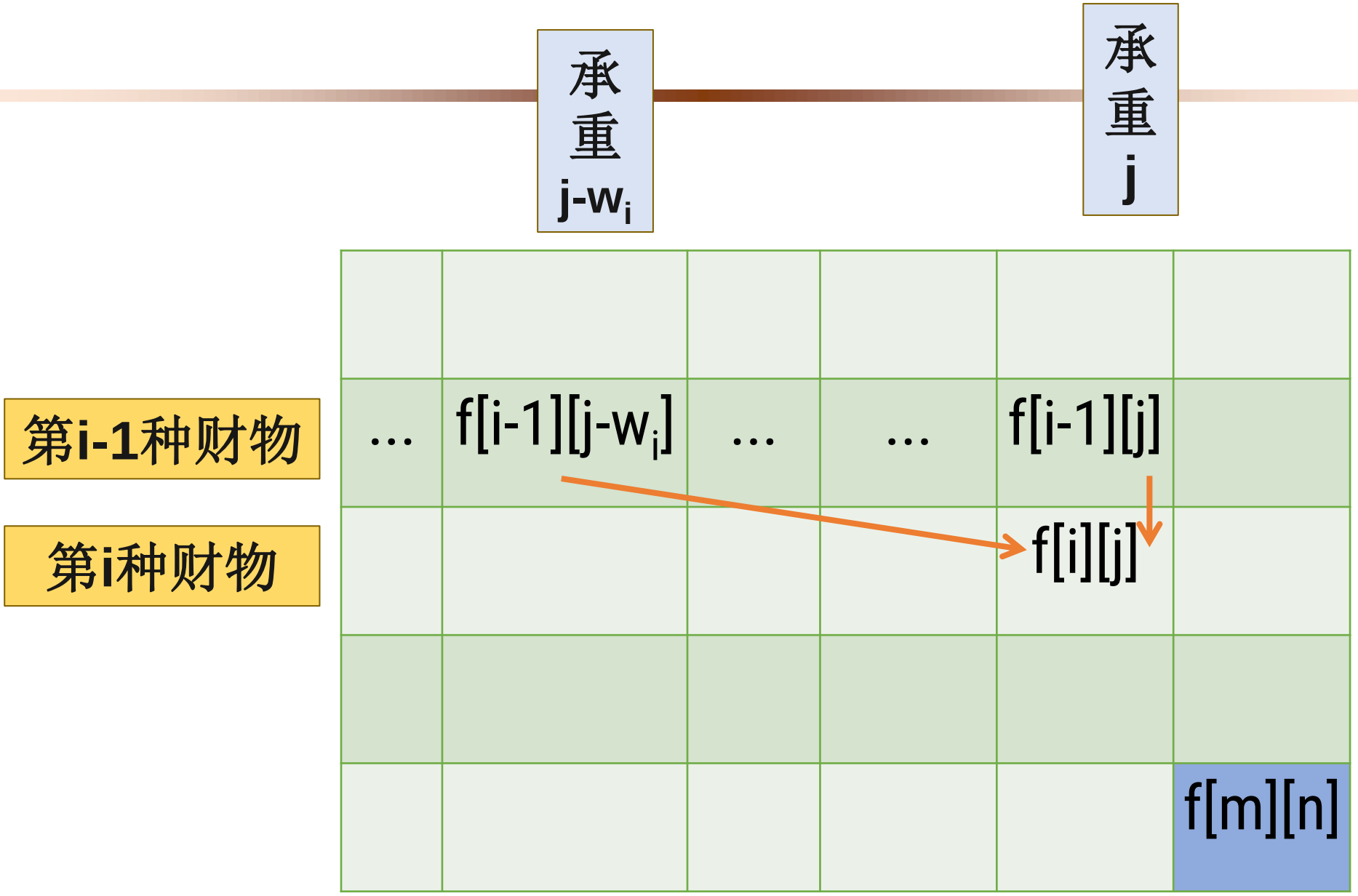
01背包：小偷

二维递推算法：
f[i][j]表示只装前i种财物，
用j公斤以内的包，最多拿多少总价值

当i为0时$f[0][j] = 0$	当j为0时$f[i][0] = 0$

$$f[i][j] = max(f[i-1][j], f[i-1][j-w_i] + v_i \mid j \geq w_i)$$

若j>=w_i
可以拿第i件


承重 j-w_i
承重 j
第i-1种财物
第i种财物
…
f[i-1][j-w_i]
…
…
f[i-1][j]
f[i][j]
f[m][n]

01背包：小偷

小偷来你家，他带的包只能装n公斤的财物。你家有m种财务，分别重$w_1,w_2,\ldots,w_m$公斤，价值分别为$v_1,v_2,\ldots,v_m$
输入整数n,m, $w_1,w_2,\ldots,w_m$, $v_1,v_2,\ldots,v_m$
输出带走**最大的总价值**是多少

f[i][j]表示只装前i种财物，用j公斤以内的包，最大总价值能拿多少

当i为0时$f[0][j] = 0$	当j为0时$f[i][0] = 0$

$$f[i][j] = max\{ f[i-1][j], f[i-1][j-w_i] + v_i \}$$

若j>=wi

01背包问题 - 代码

```
1  #include <iostream>
2  #define M 505
3  #define N 2005
4  using namespace std;
5  int n,m,w[M],v[M],f[M][N];
6  int main(){
7      cin>>n>>m;
8      for(int i=1;i<=m;i++) cin>>w[i]>>v[i];
9      //初始化清零，已经默认执行了
10     for(int i=1;i<=m;i++)    //循环查看每种财物i
11         for(int j=0;j<=n;j++) {  //循环查看背包剩余承重j
12             if(j<w[i])    //财物i太重，无法放入
13                 f[i][j]=f[i-1][j];
14             else          //比较两种决策：财物i可以放，或者不放
15                 f[i][j]=max(f[i-1][j],f[i-1][j-w[i]]+v[i]);
16         }
17     cout<<f[m][n]<<endl;
18     return 0;
19 }
```

01背包问题

穷举搜索

组合	重量	价值	
什么也不拿	0	0	请叫我雷锋
L	3	5000	
P	1	3000	
B	2	3500	
LP	4	8000	能装下的最值钱的了
LB	5	8500	可惜包小点
PB	3	6500	
LPB	6	11500	虽然很想拿，可惜装不下

穷举搜索

01背包问题

n件物品，每一件都有放入背包（1）或不放入（0）两种选项，总共有2^n种组合。

遍历每一种组合，求出符合条件（总重量小于W）而且价值最大的组合。

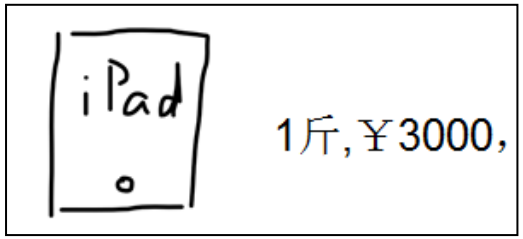
时间复杂度$O(2^n)$

01背包问题

对于每个物品，拿还是不拿，这是个问题

	0	1	2	3	4
0.nothing	0	0	0	0	0
1.Laptop	0	0	0	5000	5000
2.iPad	0	3000	3000	5000	8000
3.xBox	0	3000	3500	6500	8000

举例：计算f[3][3]时
比较f[2][3]和f[2][1]+3500

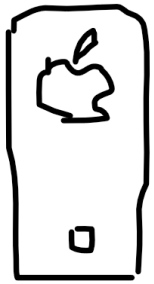


01背包问题

3斤,￥5000，

1斤,￥3000，

2斤,￥3500，



1斤,￥5000，

新增财物

01背包问题

	0	1	2	3	4
0.nothing	0	0	0	0	0
1.Laptop	0	0	0	5000	5000
2.iPad	0	3000	3000	5000	8000
3.xBox	0	3000	3500	6500	8000
4.iPhone	0	5000	8000	8500	11500

新增财物

01背包问题

穷举搜索 VS 二维表格递推算法

其实叫做动态规划算法

每增加一个物品，搜索次数增大一倍

每增加一个物品，表格增加一行

复杂度分析

N件物品
背包容量为C

能不能优化？

O(NC)

时间复杂度已经最小，
空间复杂度还可以优化

01背包问题

	0	1	2	3	4
0.nothing	0	0	0	0	0
1.Laptop	0	0	0	5000	5000
2.iPad	0	3000	3000	5000	8000
3.xBox	0	3000	3500	6500	8000
4.iPhone	0	5000	8000	8500	11500

计算第i件物品只用到i-1行的结果
无需保留整个表格

变种描述：采药

辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”如果你是辰辰，你能完成这个任务吗？

输入第一行有两个整数T（1 <= T <= 1000）和M（1 <= M <= 100），用一个空格隔开，T代表总共能够用来采药的时间，M代表山洞里的草药的数目。接下来的M行每行包括两个在1到100之间（包括1和100）的整数，分别表示采摘某株草药的时间和这株草药的价值。输出一行，这一行只包含一个整数，表示在规定的时间内，可以采到的草药的最大总价值

NOIP2005普及组

变种描述：开心的金明

金明今天很开心，家里购置的新房就要领钥匙了，新房里有一间他自己专用的很宽敞的房间。更让他高兴的是，妈妈昨天对他说：“你的房间需要购买哪些物品，怎么布置，你说了算，只要不超过N元钱就行”。今天一早金明就开始做预算，但是他想买的东西太多了，肯定会超过妈妈限定的N元。于是，他把每件物品规定了一个重要度，分为5等：用整数1~5表示，第5等最重要。他还从因特网上查到了每件物品的价格（都是整数元）。他希望在不超过N元（可以等于N元）的前提下，使每件物品的价格与重要度的乘积的总和最大。设第j件物品的价格为v[j]，重要度为w[j]，共选中了k件物品，编号依次为j1，j2，……，jk，则所求的总和为：v[j1]*w[j1]+v[j2]*w[j2]+ …+v[jk]*w[jk]。
请你帮助金明设计一个满足要求的购物单。

NOIP 2006 普及组

01背包问题 - 总结规律

给定总的容量/时间/预算限制

有若干种备选物品/草药，每种只能用**0次或1次**

每次装包/购买/使用，都会消耗/费用

最大化总价值/收益/满足度

动态规划
dynamic programming

动态规划(DP)，是求解决策最优化问题的数学方法，把多阶段过程转化为一系列子问题，利用各阶段间的关系，依次逐个求解。