

CS101

A Mars rover, likely a Curiosity rover, is shown on a rocky, reddish-brown landscape. The rover is white with various instruments and cameras. It has six large, treaded wheels. The background shows a hazy, orange-tinted sky and distant hills. The overall scene is a typical representation of the Martian surface.

信奥
算法

课件下载地址:

<http://pan.baidu.com/s/1nu6kYkL>

作业01

```
1 #include<iostream>
2 using namespace std;
3 long long i,n,f[31];
4 int main(){
5     cin>>n;
6     f[0]=1;
7     for(i=1;i<=n;i++)
8         f[i]=f[i-1]*(4*i-2)/(i+1);
9     cout<<f[n]<<endl;
10    return 0;
11 }
```

作业02

```
1  #include<iostream>
2  #define M 21
3  using namespace std;
4  long long f[M][M],n,m,i,j;
5  int main(){
6      cin>>n>>m;
7      for(j=1;j<=m;j++) f[1][j]=1;
8      for(i=2;i<=n;i++){
9          f[i][1]=1;
10         for(j=2;j<=m;j++)
11             f[i][j]=f[i-1][j]+f[i][j-1];
12     }
13     cout<<f[n][m]<<endl;
14     return 0;
15 }
```

作业03

```
1 #include<iostream>
2 #define M 100001
3 using namespace std;
4 int f[M],n,a,b,c,xa,xb,xc,i,j;
5 int main(){
6     cin>>n>>a>>b>>c;
7     f[0]=0;
8     for(i=1;i<=n;i++){
9         xa=xb=xc=0;
10        if(i>=a) xa=f[i-a]+a;
11        if(i>=b) xb=f[i-b]+b;
12        if(i>=c) xc=f[i-c]+c;
13        f[i]=max(max(xa,xb),xc);
14    }
15    cout<<f[n]<<endl;
16    return 0;
17 }
```

大盗：两种金砖



金块A, 3斤



金块B, 8斤

怎么选？



只能装10斤

大盗：两种金砖

大盗去抢银行，他带的包只能装**n**公斤的金块。现在银行只有两种规格的金块，一种重**a**公斤，另一种重**b**公斤，数量足够多。输入**n,a,b**，输出他最多带走多少公斤。

f[i]表示用*i*公斤以内的包最多拿多少金块

当*i*为0时

$$f[0] = 0$$

$$f[i] = \max\{$$

$$f[i - a] + a,$$

$$f[i - b] + b$$

}

若*i* ≥ *a*

若*i* ≥ *b*

初始时*f*值
都清零

银行盗贼：三种金砖

盗贼去抢银行，他带的包只能装**n**公斤的金块。现在银行只有三种规格的金块，分别重**a**公斤，**b**公斤和**c**公斤，数量足够多，金块无法分割。输入整数**n,a,b,c**，输出他最多带走多少公斤。

f[i]表示用*i*公斤以内的包最多拿多少金块

当*i*为0时

$$f[0] = 0$$

$$f[i] = \max\{$$

$$f[i - a] + a,$$

$$f[i - b] + b,$$

$$f[i - c] + c$$

}

若*i* ≥ **a**

若*i* ≥ **b**

若*i* ≥ **c**

初始时**f**值
都清零

大盗的烦恼：m种金砖

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 x_1, x_2, \dots, x_m 公斤，
数量足够多，金块无法分割。
输入整数n, m, x_1, x_2, \dots, x_m ，输出最多带走多少公斤

$f[i]$ 表示用i公斤以内的包最多拿多少金块

当i为0时

$$f[0] = 0$$

$$f[i] = \max_{j: x_j \leq i} \{ f[i - x_j] + x_j \}$$

初始时f值
都清零

在所有满足 $x_j \leq i$ 的金
块j中选择,求最优决策

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, \dots, 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, \dots, w_m ，输出**最多带走多少公斤**

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

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, \dots, 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, \dots, w_m ，输出**最多带走多少公斤**

递推算法

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 $f[i][0] = 0$

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

若 $j \geq w_j$

大盗m种金砖01限制- 代码

```
1 #include <iostream>
2 #define M 501
3 #define N 2001
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];
9     for (int i=1; i<=m; ++i)    //循环查看物品i
10         for (int j=0; j<=n; ++j) { //循环查看背包剩余重量j
11             if (j<w[i])    // 物品i太重, 无法放入
12                 f[i][j]=f[i-1][j];
13             else    // 比较两种决策: 物品i可以放, 或者不放
14                 f[i][j]=max(f[i-1][j], f[i-1][j - w[i]] + w[i]);
15         }
16     cout<<f[m][n]<<endl;
17     return 0;
18 }
```

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

计算次数约
 $O(n*m)$

暴力枚举时重复
计算了子问题

计算次数约
 $O(2^m)$

01背包问题



3斤, ¥5000,



1斤, ¥3000,



2斤, ¥3500,

怎么选?



只能装4斤

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 种财物，分别重 w_1, w_2, \dots, w_m 公斤，价值分别为 v_1, v_2, \dots, v_m
输入整数 $n, m, w_1, w_2, \dots, w_m, v_1, v_2, \dots, v_m$
输出带走**最大的总价值**是多少



只能装4斤

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 种财物，分别重 w_1, w_2, \dots, w_m 公斤，价值分别为 v_1, v_2, \dots, v_m
输入整数 $n, m, w_1, w_2, \dots, w_m, v_1, v_2, \dots, v_m$
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

小偷来你家，他带的包只能装 n 公斤的财物。你家有 m 种财物，分别重 w_1, w_2, \dots, w_m 公斤，价值分别为 v_1, v_2, \dots, v_m
输入整数 $n, m, w_1, w_2, \dots, w_m, v_1, v_2, \dots, v_m$
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 $f[i][0] = 0$

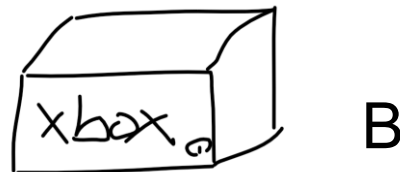
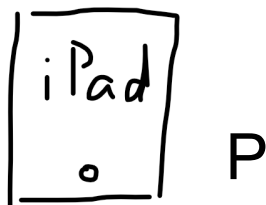
$$f[i][j] = \max \{ f[i-1][j], f[i-1][j-w_j] + v_j \}$$

若 $j \geq w_j$

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



01背包问题

穷举搜索

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。

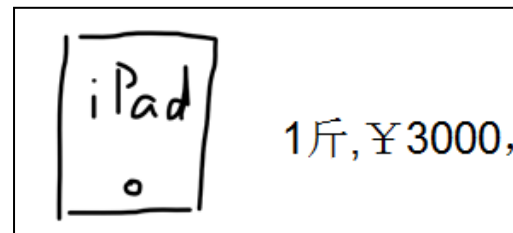
时间复杂度 $O(2^n)$

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



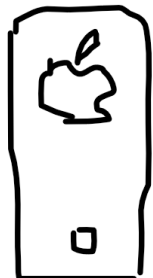
3斤, ¥5000,



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

01背包问题

$$f[i, j] = \begin{cases} f[i-1, j] & \text{当 } w_i > j \\ \max\{f[i-1, j], f[i-1, j-w_i] + v_i\} & \text{其他} \end{cases}$$

```
9   for (int i=1; i<=n; ++i)    //循环查看物品i
10  for (int j=0; j<=C; ++j) {  //循环查看背包剩余重量j
11      if (j<w[i])             // 物品i太重, 无法放入
12          f[i][j]=f[i-1][j];
13      else                     // 比较两种决策: 物品i可以放, 或者不放
14          f[i][j]=max(f[i-1][j], f[i-1][j-w[i]] + v[i]);
15  }
16  cout<<f[n][C]<<endl;
```



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 ($1 \leq T \leq 1000$) 和 M ($1 \leq M \leq 100$)，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

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 j_1, j_2, \dots, j_k ，则所求的总和为： $v[j_1]*w[j_1]+v[j_2]*w[j_2]+ \dots +v[j_k]*w[j_k]$ 。

请你帮助金明设计一个满足要求的购物单。

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。

作业

作业网站:

<http://120.132.20.20/thrall-web/main#home>