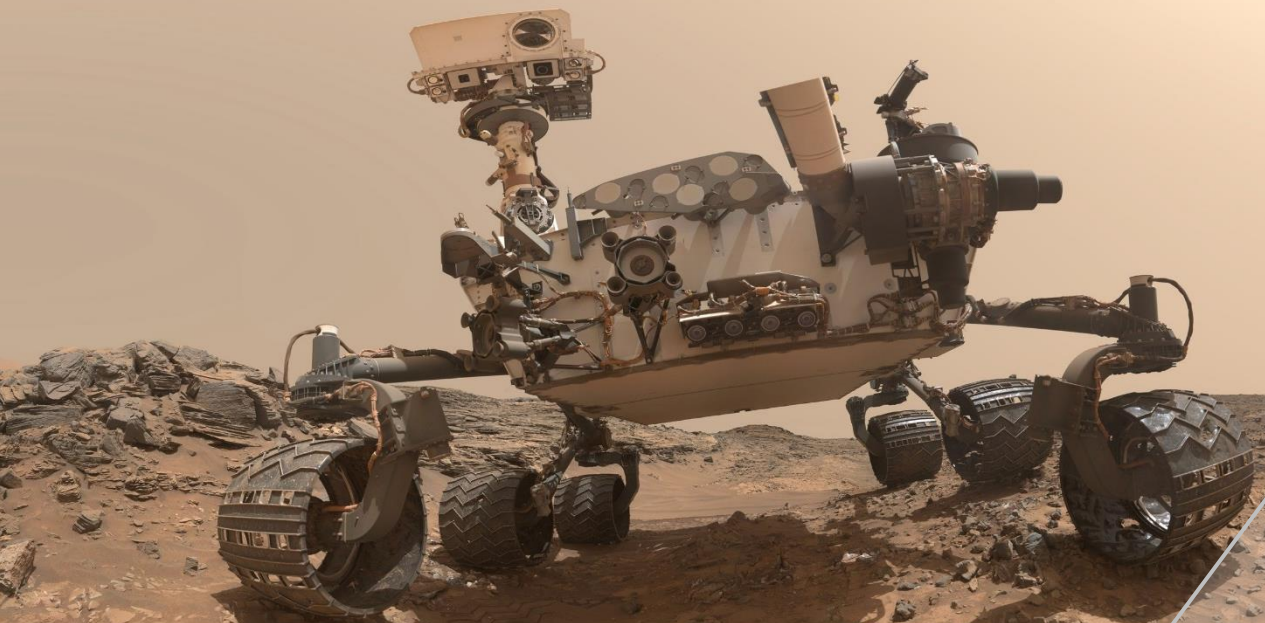
CS101



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

高级递推算法
动态规划

大盗：两种金砖

金块A, 3斤

金块B, 8斤

金块很多，
该怎么选？

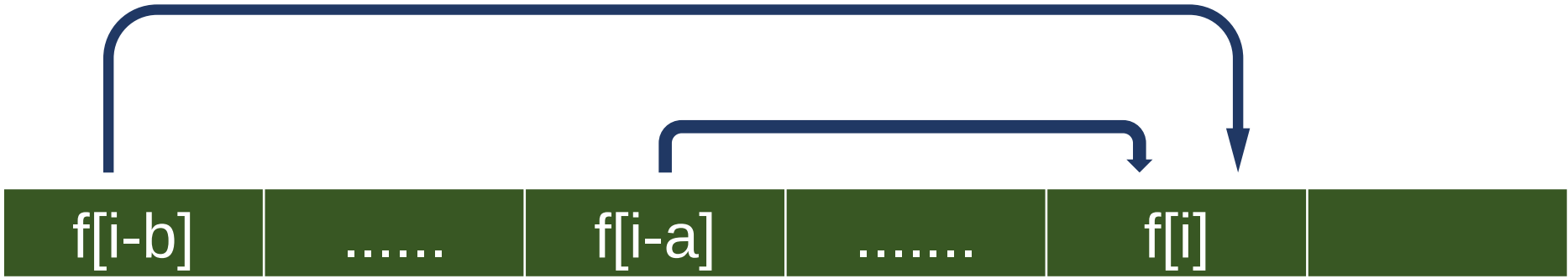
大盗：两种金砖，足够多

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。

输入样例： 10 3 8时	输入样例： 10 11 12	输入样例： 8 4 3
输出样例： 9	输出样例： 0	输出样例： 8

大盗：两种金砖，足够多

f[i]表示用i公斤以内的包最多拿多少金块



大盗：两种金砖，足够多

f[i]表示用i公斤以内的包最多拿多少金块

当i为0时	$f[0]=0$
当i<a并且i<b时	$f[i]=0$
当i>=a并且i<b时	$f[i]=f[i-a]+a$
当i<a并且i>=b时	$f[i]=f[i-a]+b$
当i>=a并且i>=b时	$f[i]=max(f[i-a]+a,\ f[i-b]+b)$

大盗：两种金砖，足够多

f[i]表示用i公斤以内的包最多拿多少金块

当i为0时	$f[0]=0$

初始时f值都清零

$$f[i]=max(\begin{matrix} f[i-a]+a|i\geq a, \\ f[i-b]+b|i\geq b \end{matrix})$$

若i>=a，可以拿重a的金块

若i>=b，可以拿重b的金块

```cpp
#include<iostream>
#define M 100001
using namespace std;
int f[M],n,a,b,xa,xb,i,j;
int main(){
    cin>>n>>a>>b;
    f[0]=0;
    for(i=1;i<=n;i++){
        if(i>=a) xa=f[i-a]+a;
        else xa=0;
        if(i>=b) xb=f[i-b]+b;
        else xb=0;
        f[i]=max(xa,xb);
    }
    cout<<f[n]<<endl;
    return 0;
}
```

银行盗贼：三种金砖，足够多

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	$f[0]=0$

$$f[i]=max(\begin{matrix} f[i-a]+a|i\geq a, \\ f[i-b]+b|i\geq b, \\ f[i-c]+c|i\geq c \end{matrix})$$

若i>=a
若i>=b
若i>=c

初始时f值都清零

银行盗贼：三种金砖，足够多

f[i]表示用i公斤以内的包最多拿多少金块

大盗的烦恼：m种金砖，足够多

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$x_1,x_2,\ldots,x_m$公斤，数量足够多，金块无法分割。
输入整数n,m, $x_1,x_2,\ldots,x_m$, 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	$f[0]=0$

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

初始时f值都清零

在所有满足$x_j \le i$的金块j中选择,求最优决策

大盗：m种金砖，01限制

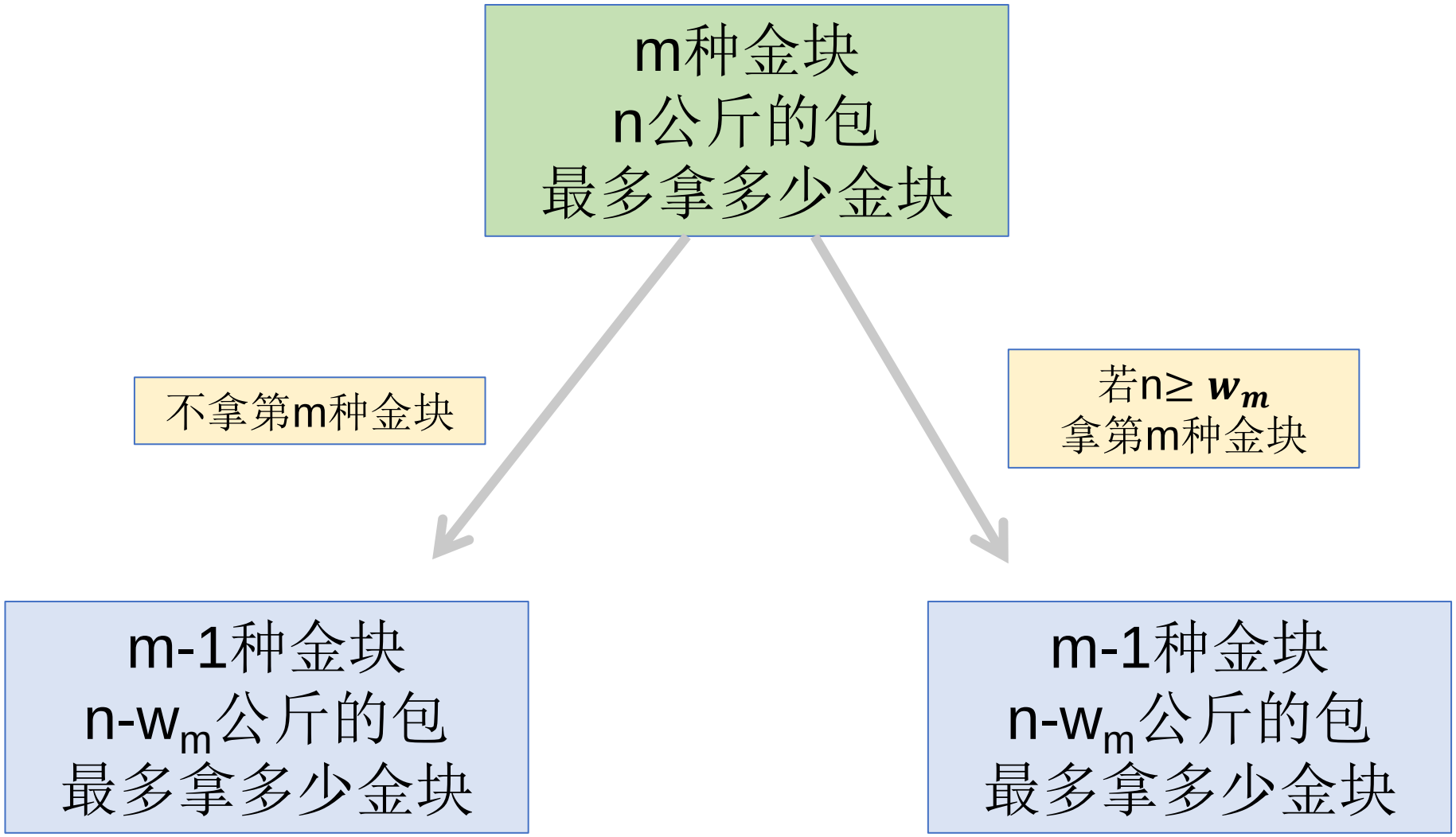
盗贼去抢银行，他带的包只能装n公斤的金块。银行有m种规格的金块，分别重$w_1,w_2,\ldots,w_m$公斤，数量都**只有1块**，金块无法分割。输入整数n,m, $w_1,w_2,\ldots,w_m$, 输出**最多带走多少公斤**

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$f[i][0] = 0$

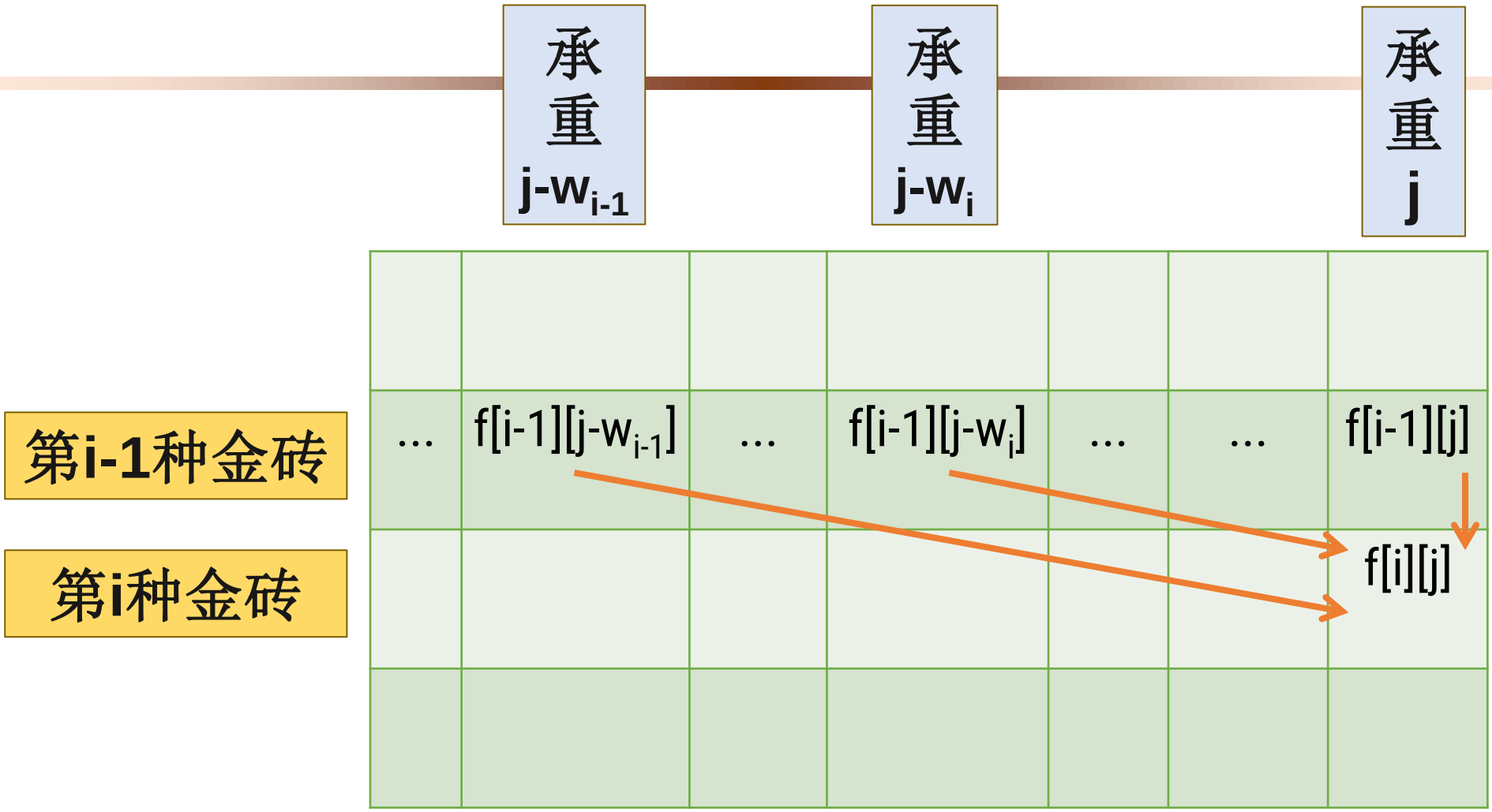
大盗：m种金砖，01限制

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-1}
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-1}] … f[i-1][j-w_i] … … f[i-1][j]
f[i][j]

大盗m种金砖01限制- 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];
9      //初始化清零，已经默认执行了
10     for(int i=1;i<=m;i++)    //循环查看每种金块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])    //金块i太重，无法放入
13                 f[i][j]=f[i-1][j];
14             else          //比较两种决策：金块i可以放，或者不放
15                 f[i][j]=max(f[i-1][j],f[i-1][j-w[i]]+w[i]);
16         }
17     cout<<f[m][n]<<endl;
18     return 0;
19 }
```

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 $O(2^m)$