

信奥算法

动态规划

Dynamic Programming

背包问题

快快编程
kkcoding.net

DP实战步骤

1

定义状态

2

优化状态

3

手动
填写
数组
所有
格子

4

再总结状态转移方程

5

再总结边界状态

6

优化
决策

kkcoding.net

01背包

小偷来你家,他带的包只能装 c 斤的财物。你家有 n 种财物,分别重 w_1, w_2, \dots, w_n 斤,价值分别为 v_1, v_2, \dots, v_n

第一行输入整数 c 和 n , $c \leq 1000, n \leq 500$ 。接着 n 行每行包含两个正整数 w_i 和 v_i , 均不超过1000。输出能拿走的**最大总价值**。

输入样例 4 3 3 50 1 30 2 35	载重4斤的包 共3种财物
输出样例 80	

请设计状态
 $f[i][j]$ 表示什么含义?

抄原题
大法

$f[i][j]$ 表示只装前 i 种财物,
用载重 j 斤的包,最多拿多少价值

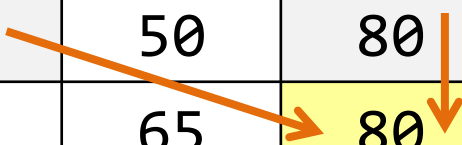
$f[i][j]$ 表示只装前*i*种财物，
用载重*j*斤的包，最多拿多少价值

c=4	n=3	$f[i][j]$	j=0	j=1	j=2	j=3	j=4
		i=0	0	0	0	0	0
$w[1]=3, v[1]=50$		i=1	0	0	0	50	50
$w[2]=1, v[2]=30$		i=2	0	30	30	50	80
$w[3]=2, v[3]=35$		i=3	0	30	35	65	80

$$\begin{aligned}
 f[3][3] &= \max(f[2][3], f[2][3-w[3]]+v[3]) \\
 &= \max(f[2][3], f[2][1]+35) \\
 &= \max(50, 30+35)
 \end{aligned}$$

$f[i][j]$ 表示只装前*i*种财物，
用载重*j*斤的包，最多拿多少价值

c=4	n=3	$f[i][j]$	j=0	j=1	j=2	j=3	j=4
		i=0	0	0	0	0	0
$w[1]=3, v[1]=50$		i=1	0	0	0	50	50
$w[2]=1, v[2]=30$		i=2	0	30	30	50	80
$w[3]=2, v[3]=35$		i=3	0	30	35	65	80



$$\begin{aligned}
 f[3][4] &= \max(f[2][4], f[2][4-w[3]]+v[3]) \\
 &= \max(f[2][4], f[2][2]+35) \\
 &= \max(80, 30+35)
 \end{aligned}$$

```
for(ll i=1;i<=n;i++)cin>>w[i]>>v[i];
for(ll i=1;i<=n;++i)
    for(ll j=1;j<=c;++j)
        if()
            f[i][j]=f[i-1][j];
        else
            f[i][j]=max(f[i-1][j],) );
cout<<f[n][c]<<endl;
```

01背包

覆盖式填写

快快编程
kkcoding.net

填表 顺序	从右 往左	可以 吗
----------	----------	---------

01背包

c=4	n=3
-----	-----

w[1]=3, v[1]=50

w[2]=1, v[2]=30

w[3]=2, v[3]=35

f[i][j]	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	0	0	50	50
i=2	0	30	30	50	80
i=3	0	30	35	65	80

填表顺序从右往左可以的

01背包

c=4 n=3

w[1]=3, v[1]=50

w[2]=1, v[2]=30

w[3]=2, v[3]=35

f[i][j]	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	0	0	50	50
i=2	0	30	30	50	80
i=3	0	30	35	65	80

每一格只依赖上一行的最多2个数字
每一格依赖上一行里正上方邻居和左侧某格

从右往左填写时每格所依赖的格子
都在上一行都已经计算完毕了

填表 顺序	从右 往左	可以 的
----------	----------	---------

01背包

c=4	n=3
-----	-----

w[1]=3,v[1]=50

w[2]=1,v[2]=30

w[3]=2,v[3]=35

f[i][j]	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	0	0	50	50
i=2	0	30	30	50	80
i=3					80

$$f[3][4]=\max(f[2][4],f[2][4-w[3]]+v[3])$$

$$=\max(f[2][4],f[2][2]+35)$$

$$=\max(80,30+35)$$

填表
顺序

从右
往左

可以
的

01背包

c=4 n=3

w[1]=3, v[1]=50

w[2]=1, v[2]=30

w[3]=2, v[3]=35

f[i][j]	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	0	0	50	50
i=2	0	30	30	50	80
i=3				65	80

$$\begin{aligned} f[3][3] &= \max(f[2][3], f[2][3-w[3]]+v[3]) \\ &= \max(f[2][3], f[2][1]+35) \\ &= \max(50, 30+35) \end{aligned}$$

填表
顺序

从右
往左

可以
的

01背包

c=4 n=3

w[1]=3, v[1]=50

w[2]=1, v[2]=30

w[3]=2, v[3]=35

f[i][j]	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	0	0	50	50
i=2	0	30	30	50	80
i=3			35	65	80

$$\begin{aligned} f[3][2] &= \max(f[2][2], f[2][2-w[3]]+v[3]) \\ &= \max(f[2][2], f[2][0]+35) \\ &= \max(30, 0+35) \end{aligned}$$

填表
顺序

从右
往左

可以
的

01背包

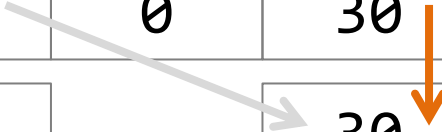
$c=4$ $n=3$

$w[1]=3, v[1]=50$

$w[2]=1, v[2]=30$

$w[3]=2, v[3]=35$

$f[i][j]$	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$
$i=0$	0	0	0	0	0
$i=1$	0	0	0	50	50
$i=2$	0	30	30	50	80
$i=3$		30	35	65	80



$$f[3][1] = f[2][1]$$

因为背包载重 $j=1$ 比物品重量 $w[3]=2$ 还小
只能选择不拿3号物品

$j-w[i]$ 是负数越界了

填表 顺序	从右 往左	可以 的
----------	----------	---------

01背包

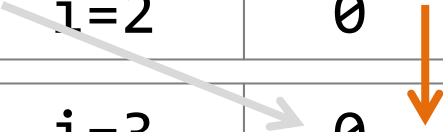
c=4	n=3
-----	-----

w[1]=3, v[1]=50

w[2]=1, v[2]=30

w[3]=2, v[3]=35

f[i][j]	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	0	0	50	50
i=2	0	30	30	50	80
i=3	0	30	35	65	80



$$f[3][0] = f[2][0]$$

因为背包载重 $j=0$ 比物品重量 $w[3]=2$ 还小
只能选择不拿3号物品

$j - w[i]$ 是负数越界了

填表 顺序	从右 往左	可以 的
----------	----------	---------

01背包

c=4	n=3
-----	-----

w[1]=3, v[1]=50

w[2]=1, v[2]=30

w[3]=2, v[3]=35

f[i][j]	j=0	j=1	j=2	j=3	j=4
i=0	0	0	0	0	0
i=1	0	0	0	50	50
i=2	0	30	30	50	80
i=3	0	30	35	65	80

观察
发现

当 $j < w[i]$ 时, i 号物品装不下
 填表格 $f[i][j]$ 直接抄写上一行 $f[i-1][j]$

填表
顺序

从右
往左

可以
的

01背包

$c=4$ $n=3$

$w[1]=3, v[1]=50$

$w[2]=1, v[2]=30$

$w[3]=2, v[3]=35$

$f[i][j]$	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$
$i=0$	0	0	0	0	0
$i=1$	0	0	0	50	50
$i=2$	0	30	30	50	80
$i=3$	0	30	35	65	80

重大
发现

从右往左填写时

当 $f[i][j]$ 算完以后
 $f[i-1][j]$ 再也不会被依赖了

只需要一维数组记录 $f[j]$

覆盖填写 从右往左 可以的

01背包

$f[j]$	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$
	0	0	0		
$i=3$		30	35	65	80

当 $j < w[i]$ 时, 无需计算保留 $f[j]$ 即可

只需要一维数组记录 $f[j]$

一维
数组

$[i]$ 维度
被隐藏

$f[j]$ 最终含义为载重 j 的包能装的最大价值

计算中, 每步增加物品 i , $f[j]$ 含义在变化

$f[j]$ 中间含义为只装前 i 种物品

用载重 j 的包能装的最大价值

覆盖
填写

从右
往左

01背包

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  const int MAXC=2009;
4  int n,c,w,v,f[MAXC];
5  int main(){
6      cin>>c>>n;
7      for(int i=1;i<=n;i++){
8          cin>>w>>v;
9          for(int j=c;j>=w;j--){
10             f[j]=max(f[j],f[j-w]+v);
11         }
12         cout<<f[c]<<endl;
13         return 0;
14     }
```

一维数组

从大到小
枚举载重j

快快编程
kkcoding.net

思考题

如果枚举载重 j 的循环
改为从小到大
 $f[j]$ 对应什么结果?

本质就是完全背包

$f[j]$ 中间含义为载重为 j 的包
只考虑前 i 种物品
每种物品无限数量
可以装的最大价值

完全背包

无限数量

快快编程
kkcoding.net

完全背包

小偷来你家,他带的包只能装 c 斤的财物。你家有 n 种财物,每种数量无限多,分别重 w_1, w_2, \dots, w_n 斤,价值分别为 v_1, v_2, \dots, v_n

第一行输入整数 c 和 n , $c \leq 10000, n \leq 100$ 。接着 n 行每行包含两个正整数 w_i 和 v_i , 均不超过1000。输出能拿走的**最大总价值**。

输入样例

```
5 2
1 10
2 30
```

输出样例

```
70
```

$f[i][j]$ 表示只装前 i 种财物,用载重 j 斤的包,最多拿多少价值

额外条件: 物品数量无限


完全背包

c=5	n=2
-----	-----

w[1]=1, v[1]=10

w[2]=2, v[2]=30

f[i][j]	j=0	j=1	j=2	j=3	j=4	j=5
i=0	0	0	0	0	0	0
i=1	0	10	20	30	40	50
i=2	0	10	30	40	60	70



$$f[2][0] = f[1][0]$$

$$f[2][1] = f[1][1]$$

完全背包

c=5	n=2
-----	-----

f[i][j]	j=0	j=1	j=2	j=3	j=4	j=5
i=0	0	0	0	0	0	0
i=1	0	10	20	30	40	50
i=2	0	10	30	40	60	70

w[1]=1, v[1]=10

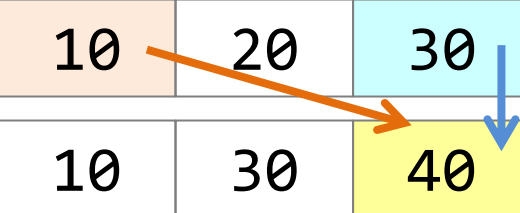
w[2]=2, v[2]=30

$$\begin{aligned}f[2][2] &= \max\{f[1][2], f[1][0] + 30\} \\ &= \max\{20, 0 + 30\}\end{aligned}$$

完全背包

c=5	n=2
-----	-----

f[i][j]	j=0	j=1	j=2	j=3	j=4	j=5
i=0	0	0	0	0	0	0
i=1	0	10	20	30	40	50
i=2	0	10	30	40	60	70



The diagram illustrates the calculation of the value at f[2][3] in the DP table. An orange arrow points from the cell at i=1, j=3 (value 30) to the cell at i=2, j=3 (value 40). A blue arrow points from the cell at i=1, j=1 (value 10) to the same cell at i=2, j=3 (value 40). The cell at i=2, j=3 is highlighted in yellow, indicating it is the current state being calculated.

w[1]=1, v[1]=10

w[2]=2, v[2]=30

$$\begin{aligned}f[2][3] &= \max\{f[1][3], f[1][1] + 30\} \\ &= \max\{30, 10 + 30\}\end{aligned}$$

完全背包

$c=5$	$n=2$
-------	-------

$f[i][j]$	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
-----------	-------	-------	-------	-------	-------	-------

$i=0$	0	0	0	0	0	0
-------	---	---	---	---	---	---

$w[1]=1, v[1]=10$

$i=1$	0	10	20	30	40	50
-------	---	----	----	----	----	----

$w[2]=2, v[2]=30$

$i=2$	0	10	30	40	60	70
-------	---	----	----	----	----	----

$$f[2][4] = \max\{f[1][4], f[1][2] + 30, f[1][0] + 30 \times 2\}$$

$$= \max\{40, 20 + 30, 0 + 30 \times 2\}$$

决策

状态间
依赖关系

i 号物品
选几件

多选一
选择题

完全背包

c=5	n=2	f[i][j]	j=0	j=1	j=2	j=3	j=4	j=5
		i=0	0	0	0	0	0	0
w[1]=1,v[1]=10		i=1	0	10	20	30	40	50
w[2]=2,v[2]=30		i=2	0	10	30	40	60	70

$$f[2][5] = \max\{f[1][5], f[1][3] + 30, f[1][1] + 30 \times 2\}$$

$$= \max\{50, 30 + 30, 10 + 30 \times 2\}$$

重大发现

$f[2][5]$ 和 $f[2][3]$ 依赖的格子有重叠

决策

状态间
依赖关系

i号物品
选几件

多选一
选择题

完全背包

c=5	n=2
-----	-----

f[i][j]	j=0	j=1	j=2	j=3	j=4	j=5
i=0	0	0	0	0	0	0
i=1	0	10	20	30	40	50
i=2	0	10	30	40	60	70

w[1]=1, v[1]=10

w[2]=2, v[2]=30

$$f[2][5] = \max\{f[1][5], f[1][3] + 30, f[1][1] + 30 * 2\}$$

$$= \max\{f[1][5], f[2][3] + 30\}$$

重大发现

f[2][5]和f[2][3]依赖的格子有重叠

决策

状态间
依赖关系

i号物品
选几件

多选一
选择题

完全背包

$c=5$	$n=2$
-------	-------

$f[i][j]$	$j=0$	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
-----------	-------	-------	-------	-------	-------	-------

$i=0$	0	0	0	0	0	0
-------	---	---	---	---	---	---

$w[1]=1, v[1]=10$

$i=1$	0	10	20	30	40	50
-------	---	----	----	----	----	----

$w[2]=2, v[2]=30$

$i=2$	0	10	30	40	60	70
-------	---	----	----	----	----	----



当 $j \geq w[i]$	$f[i][j] = \max(f[i-1][j], f[i][j-w[i]] + v[i])$
-----------------	--

决策 简化	i 号物品 选不选	二选一 是非题
----------	----------------	------------

小结：DP决策优化加速



决策加速
经典方法

i号物品
选几件

多选一
选择题



i号物品
选不选



二选一
是非题

完全背包

c=5	n=2
-----	-----

f[i][j]	j=0	j=1	j=2	j=3	j=4	j=5
---------	-----	-----	-----	-----	-----	-----

i=0	0	0	0	0	0	0
-----	---	---	---	---	---	---

w[1]=1, v[1]=10

i=1	0	10	20	30	40	50
-----	---	----	----	----	----	----

w[2]=2, v[2]=30

i=2	0	10	30	40	60	70
-----	---	----	----	----	----	----



当j>=w[i]	$f[i][j] = \max(f[i-1][j], f[i][j-w[i]] + v[i])$
----------	--

覆盖 填写	重大 发现	当f[i][j]算完以后 f[i-1][j]再也不会被依赖了 只需要一维数组记录f[j]
----------	----------	--

覆盖
填写

从左
往右

完全背包

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  const int C=10009;
4  int c,n,w,v,f[C];
5  int main(){
6      cin>>c>>n;
7      for(int i=1;i<=n;i++){
8          cin>>w>>v;
9          for(int j=w;j<=c;j++){
10             f[j]=max(f[j],f[j-w]+v);
11         }
12         cout<<f[c]<<endl;
13         return 0;
14     }
```

从小到大
枚举载重j

快快编程
kkcoding.net

**01
背包**

算法
对比

完全
背包

快快编程
kkcoding.net

01 背包

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

完全 背包

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

01 背包

```

6
7
8
9
10
11
12
cin>>c>>n;
for(int i=1;i<=n;i++) {
    cin>>w>>v;
    for(int j=c;j>=w;j--) ←
        f[j]=max(f[j],f[j-w]+v);
}
cout<<f[c]<<endl;

```

完全 背包

```

6
7
8
9
10
11
12
cin>>c>>n;
for(int i=1;i<=n;i++){
    cin>>w>>v;
    for(int j=w;j<=c;j++) ←
        f[j]=max(f[j],f[j-w]+v);
}
cout<<f[c]<<endl;

```

二维背包

两种约束条件

快快编程
kkcoding.net

二维完全背包

学校发暑期福利,给了你总共能容纳 V 体积的背包,最大承重量为 W 。允许你到超市里任意选购商品,学校报销哦。
现在超市有 n 种物品,每种物品无限件。
每种物品有其体积 v_i ,价值 p_i ,重量 w_i 。
请计算出你能得到的最大价值是多少?

第一行,3个正整数, n, V, W 。
第二行, n 个正整数, v_i 。
第三行, n 个正整数, p_i 。
第四行, n 个正整数, w_i 。

输入样例
2 10 10
3 8
20 30
4 5

输出样例
40

第1种物品体积3重量4价值20
第2种物品体积8重量5价值30

二维完全背包

学校发暑期福利,给了你总共能容纳 V 体积的背包,最大承重量为 W 。允许你到超市里任意选购商品,学校报销哦。
现在超市有 n 种物品,每种物品无限件。
每种物品有其体积 v_i ,价值 p_i ,重量 w_i 。
请计算出你能得到的最大价值是多少?

第一行,3个正整数, n, V, W 。
第二行, n 个正整数, v_i 。
第三行, n 个正整数, p_i 。
第四行, n 个正整数, w_i 。

$f[i][j][k]$ 表示
只装前 i 种物品
用容量 j 载重 k 的包
最多拿多少价值

对 $[i]$ 维度进行隐藏
保留 $f[j][k]$ 用覆盖式填表
枚举 j, k 都从小到大

二维完全背包

```
for(11 i=1;i<=n;i++)  
    for(11 j=v[i];j<=V;j++)  
        for(11 k=w[i];k<=W;k++)  
            f[j][k]=max(f[j][k],  );  
cout<<f[V][W]<<endl;
```

$f[i][j][k]$ 表示
只装前 i 种物品
用容量 j 载重 k 的包
最多拿多少价值

对 $[i]$ 维度进行隐藏
保留 $f[j][k]$ 用覆盖式填表
枚举 j, k 都从小到大大

二维01背包

学校发暑期福利,给了你总共能容纳 V 体积的背包,最大承重量为 W 。允许你到超市里任意选购商品,学校报销哦。
现在超市有 n 种物品,每种物品只有1件。
每种物品有其体积 v_i ,价值 p_i ,重量 w_i 。
请计算出你能得到的最大价值是多少?

第一行,3个正整数, n, V, W 。
第二行, n 个正整数, v_i 。
第三行, n 个正整数, p_i 。
第四行, n 个正整数, w_i 。

输入样例
2 10 10
3 8
20 30
4 5

输出样例
30

第1种物品体积3重量4价值20
第2种物品体积8重量5价值30

二维01背包

学校发暑期福利,给了你总共能容纳 V 体积的背包,最大承重量为 W 。允许你到超市里任意选购商品,学校报销哦。
现在超市有 n 种物品,每种物品只有1件。
每种物品有其体积 v_i ,价值 p_i ,重量 w_i 。
请计算出你能得到的最大价值是多少?

第一行,3个正整数, n, V, W 。
第二行, n 个正整数, v_i 。
第三行, n 个正整数, p_i 。
第四行, n 个正整数, w_i 。

$f[i][j][k]$ 表示
只装前 i 种物品
用容量 j 载重 k 的包
最多拿多少价值

对 $[i]$ 维度进行隐藏
保留 $f[j][k]$ 用覆盖式填表
枚举 j, k 都从大到小

二维01背包

```
for(11 i=1;i<=n;i++)  
    for(11 j=V;j>=v[i];j--)  
        for(11 k=W;k>=w[i];k--)  
            f[j][k]=max(f[j][k],  );  
cout<<f[V][W]<<endl;
```

$f[i][j][k]$ 表示
只装前 i 种物品
用容量 j 载重 k 的包
最多拿多少价值

对 $[i]$ 维度进行隐藏
保留 $f[j][k]$ 用覆盖式填表
枚举 j, k 都从大到小

快快编程作业

133

134

1169

拓展题

1170, 1098