蛟龙五班

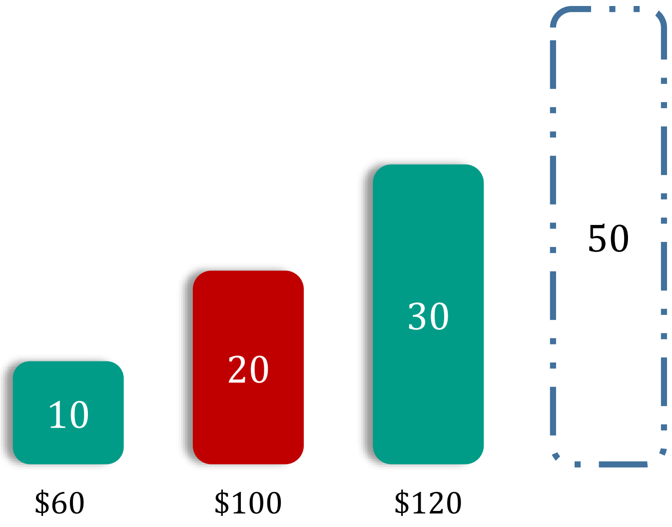
背包DP

Mas

01背包

有N件物品和一个容量为W的背包

第i件物品的重量是w_i,价值是v_i, 装入背包的物品重量总和不超过W时价值最大是多少?



设 $dp[i][j]$ 为在只考虑前 i 个物品的情况下,容量为 j 的背包所能达到的最大总价值

假设当前已经处理好了前 $i-1$ 个物品的所有状态,那么对于第 i 个物品

当其不放入时,背包的剩余容量不变,背包中物品的总价值也不变,故这种情况的最大价值为$dp[i-1][j]$

当其放入时,背包的剩余容量会减小 w_i,背包中物品的总价值会增大 v_i ,这种情况的最大价值为 $dp[i-1][j-w_i]+v_i$

状态转移方程

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

答案为$dp[n][m]$

01背包

观察发现$dp[i]$仅与$dp[i-1]$有关,可以使用两个数组交替滚动节省内存

体积	价值	0	1	2	3	4	5	6	7	8	9	10
		0	0	0	0	0	0	0	0	0	0	0
2	6	0	0	6	6	6	6	6	6	6	6	6
2	3	0	0	6	6	9	9	9	9	9	9	9
6	5	0	0	6	6	9	9	9	9	11	11	14
5	4	0	0	6	6	9	9	9	10	11	13	14
4	6	0	0	6	6	9	9	12	12	15	15	15

$i\&1$与$(i-1)\&1$奇偶性交替,可以实现数组的交替更新

```
for (int i = 1; i <= n; i++)
    for (int j = 0; j <= m; j++)
        if (j < w[i])
            dp[i][j] = dp[i - 1][j];
        else
            dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - w[i]] + v[i]);
```

```
for (int i = 1; i <= n; i++)
    for (int j = 0; j <= m; j++)
        if (j < w[i])
            dp[i & 1][j] = dp[(i - 1) & 1][j];
        else
            dp[i & 1][j] = max(dp[(i - 1) & 1][j], dp[(i - 1) & 1][j - w[i]] + v[i]);
```

01背包

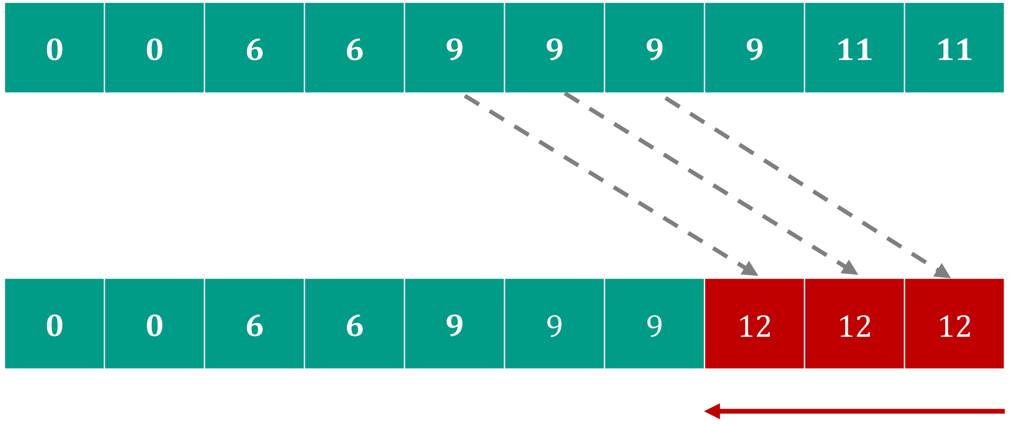
再进一步观察，对于每一个$dp[i][j]$仅与$dp[i-1][j-w_i]$有关

若改变枚举容量的顺序(从大到小),对于每个当前容量j

此时$j-w[i]$还未被修改,依然保留$dp[i-1][j-w[i]]$的值

可仅保留一维

时间复杂度$O(NW)$,空间复杂度$O(W)$




```
for (int i = 1; i <= n; i++)
    for (int j = m; j >= w[i]; j--)
        dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
```

#1158、01背包

题目描述

一个旅行者有一个最多能装 M 公斤的背包,现在有 n 件物品,它们的重量分别是 $W_1, W_2, ..., W_n$,它们的价值分别为 $C_1, C_2, ..., C_n$,求旅行者能获得最大总价值

输入格式

第一行：两个整数, M (背包容量, $M \leq 2000$)和 N (物品数量, $N \leq 32$)

第 $2 \sim N+1$ 行：每行二个整数 W_i, C_i ,表示每个物品的重量和价值

输出格式

仅一行,一个数,表示最大总价值

输入样例

```
10 4
2 1
3 3
4 5
7 9
```

输出样例

```
12
```

多重背包

有N种物品,每种物品数量为K_i,和一个容量为W的背包,第i件物品的重量是w_i,价值是v_i,

装入背包的物品重量总和不超过W时价值最大是多少

设 $dp[i][j]$ 为在只考虑前 i 个物品的情况下,容量为 j 的背包所能达到的最大总价值

每种物品有k_i件等价于01背包中有K_i件相同物品

对于每件物品可以尝试枚举选取的数量k转移

$$dp[i][j] = \max_{0\le k\le K_i}(dp[i-1][j-k\times w_i]+k\times v_i)$$

时间复杂度$O((\sum_{i=1}^{n} K_i)\times W)$

#1313、庆功会

题目描述

为了庆贺班级在校运动会上取得全校第一名成绩,班主任决定开一场庆功会,为此拨款购买奖品犒劳运动员

期望拨款金额能购买最大价值的奖品,可以补充他们的精力和体力

输入格式

第一行二个数 $n(n \leq 500), m(m \leq 6000)$,其中 n 代表希望购买的奖品的种数, m 表示拨款金额

接下来 n 行,每行 3 个数, v、 w、 s

分别表示第 i 种奖品的价格、价值(价格与价值是不同的概念)和能购买的最大数量（买 0 件到 s 件均可）,其中 $v \leq 100, w \leq 1000, s \leq 10$

输出格式

一行：一个数,表示此次购买能获得的最大的价值(注意！不是价格)。

```
for (int i = 1; i <= n; i++)
    for (int j = m; j >= w[i]; j--)
        for (int k = 0; k <= s[i] && k * w[i] <= j; k++)
            dp[j] = max(dp[j], dp[j - w[i] * k] + v[i] * k);
```

多重背包

将K_i拆分成$2^0, 2^1, 2^2, \cdots, 2^{p-1}, 2^p, K_i-(2^{p+1}-1)$

其中$K_i \geq 2^{p+1}-1, K_i < 2^{p+2}-1$

不难发现$2^0, 2^1, 2^2, \cdots, 2^{p-1}, 2^p$可以组合出$[0, 2^{p+1}-1]$所有整数

对于$x \geq 2^{p+1}$可以x将表示为$K_i-(2^{p+1}-1)+t$,其中t显然是$[0, 2^{p+1}-1]$范围内整数

所以对于K_i使用$2^0, 2^1, 2^2, \cdots, 2^{p-1}, 2^p, K_i-(2^{p+1}-1)$进行组合,可以组合出$[0, K_\mathrm{i}]$范围内所有整数

若使用该种拆分方式K_i件物品可被拆分成重量和价值不一的$\log_2 n$组

多重背包时间复杂度可被优化至$O(\sum_{i=1}^{n}(\log_2 K_i) \times W)$

```
void spilt(int x, int y, int s)
{
    for (int t = 1; s >= t; s -= t, t <<= 1)
        v[pos] = t * x, w[pos++] = t * y;
    if (s)
        v[pos] = s * x, w[pos++] = s * y;
}
```

注意：多重背包还可以用单调队列优化至 $O(NW)$

完全背包

有N种物品(每种物品数量无限)和一个容量为W的背包,第i件物品的重量是w_i,价值是v_i,

装入背包的物品重量总和不超过W时价值最大是多少

设 $dp[i][j]$ 为在只考虑前 i 个物品的情况下,容量为 j 的背包所能达到的最大总价值

对于每件物品可以尝试枚举选取的数量k转移

$$dp[i][j] = \max_{0 \le k \le \left\lfloor \frac{j}{w_i} \right\rfloor} (dp[i-1][j-k \times w_i] + k \times v_i)$$

最坏情况下时间复杂度$O(NW^2)$

套用之前多重背包的优化方法,计算 $dp[i][j]$ 时枚举当前物品的组合 $[w_i, v\_i]. [2w_i, 2v_i], [4w_i, 4v_i], \ldots$。

这样一个完全背包的物品会扩展成 $O(\log W)$ 个 01 背包的物品

最坏情况下时间复杂度$O(NW\text{logW})$

完全背包

$$dp[i][j] = \max\{\, dp[i-1][j] \,, dp\,[i-1][j-w_i] + v_i, dp[i-1][j-2w_i] + 2v_i \,, \ldots \ \}$$

$$dp[i][j-w_i] = \max\{dp[i-1][j-w_i], dp\,[i-1][j-2w_i] + v_i \,, dp\,[i-1][j-3w_i] + 2v_i, \ldots\}$$

可使用 $dp[\,i\,][\,j - w_i] + v_i$ 更新 $dp[i][v]$

状态转移方程

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

不需要再枚举k

相比于01背包,仅需要从小到大枚举背包容量可优化时间/压缩空间

时间复杂度$O(NW)$,空间复杂度$O(W)$

#1312、完全背包问题

题目描述

设有 n 种物品,每种物品有一个重量及一个价值,但每种物品的数量是无限的

同时有一个背包,最大载重量为 M ,今从 n 种物品中选取若干件(同一种物品可以多次选取),使其重量的和小于等于 M ,而价值的和为最大

输入格式

第一行：两个整数, M (背包容量, $M \le 200$)和 N (物品数量, $N \le 30$);

第 $2 \sim N+1$ 行：每行二个整数 W_i, C_i ,表示每个物品的重量和价值。

输出格式

仅一行,输出 `max=一个数` ,表示最大总价值

输入样例

```
10 4
2 1
3 3
4 5
7 9
```

输出样例

```
max=12
```

```
for (int i = 1; i <= n; i++)
    for (int j = w[i]; j <= m; j++)
        dp[j] = max(dp[j], dp[j - w[i]] + v[i]);
```

二维费用背包

有N种物品和一个承重量为W体积为V的背包,第i件物品的重量是w_i,体积是v_i,价值是c_i,

装入背包的物品重量总和不超过W、体积总和不超过V时价值最大是多少

设 $dp[i][j][k]$ 为在只考虑前 i 个物品的情况下,承重量为 j体积为k的背包所能达到的最大总价值

类比于01背包增加一个体积维度

$$dp[i][j][k] = \max(dp[i-1][j][k], dp[i-1][j-w_i][k-v_i] + c_i)$$

时间复杂度$O(NWV)$,若使用滚动数组优化空间复杂度为$O(WV)$

#1489、大胃王

题目描述

小明是个大胃王，他在吃自足餐时不仅能够吃很多东西，而且他能够快速地判断出眼前哪些食物的价值更高，他在填饱肚子的前提下更倾向于吃更贵的食物，但是对他来说，食物还有热量，他不能吃太多热量高的食物。

假设小明面前有 N 件食物，每件食物只能选择吃或者不吃，他最多可以吃重量为 W 的食物，并且所吃食物的总热量不能超过 K ，并且他也知道每件食物的重量 w_i ，热量 k_i 和价值 v_i ，请问他最多可以吃总价值为多少的食物?

输入格式

输入第一行为三个数， N, W, K ；

输入第二行为 N 个数，表示每个食物的重量 w ；

输入第三行为 N 个数，表示每个食物的热量 k ；

输入第四行为 N 个数，表示每个食物的价值 v 。

输出格式

输出一行一个数，表示最多的食物价值。

数据范围

对于 40% 的数据， $1 \le N, W, K \le 10$；

对于 100% 的数据， $1 \le N, w_i, k_i, v_i \le 100$， $1 \le W, K \le 1000$

输入样例1

```
3 10 10
1 5 6
5 1 5
3 6 4
```

输出样例1

```
9
```

```
scanf("%d%d%d", &N, &W, &K);
for (int i = 1; i <= N; i++)
  scanf("%d", w + i);
for (int i = 1; i <= N; i++)
  scanf("%d", k + i);
for (int i = 1; i <= N; i++)
  scanf("%d", v + i);
for (int i = 1; i <= N; i++)
  for (int j = W; j >= w[i]; j--)
    for (int l = K; l >= k[i]; l--)
      dp[j][l] = max(dp[j][l], dp[j - w[i]][l - k[i]] + v[i]);
printf("%d", dp[W][K]);
```

分组背包

有T组物品和一个承重量为W的背包,第i件物品的重量是w_i, 价值是v_i,每件物品只属于一个分组,同组内最多只能选择一个物品

装入背包的物品重量总和不超过W时价值最大是多少

设 $dp[i][j]$ 为在只考虑前 i 个分组的情况下,承重量为j的背包所能达到的最大总价值

在每次确定第i组后,在确定当前背包容量,最后枚举确定组内的物品,可确保每个组别仅选择一件

$$dp[i][j] = \max_{k \in T_i}(dp[i-1][j-w_k] + v_i)$$

#1314、分组背包

题目描述

一个旅行者有一个最多能装 V 公斤的背包。

现在有 n 件物品，它们的重量分别是 W_1， W_2， $\ldots, W_n$ ，它们的价值分别为 $C_1, C_2, \ldots, C_n$ 。

这些物品被划分为若干组，每组中的物品互相冲突，最多选一件。

求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

输入格式

第一行：三个整数， V (背包容量， $V \leq 200$)， N (物品数量， $N \leq 30$)和 T (最大组号， $T \leq 10$);

第 $2 \sim N+1$ 行：每行三个整数 W_i, C_i, P ，表示每个物品的重量，价值，所属组号。

输出格式

仅一行，一个数，表示最大总价值。

输入样例

```
10 6 3
2 1 1
3 3 1
4 8 2
6 9 2
2 8 3
3 9 3
```

输出样例

```
20
```

```cpp
cin >> v >> n >> t;
for (int i = 1; i <= n; i++)
{
  cin >> tw >> tv >> id;
  p[id].push_back({tw, tv});
}
for (int k = 1; k <= t; k++)
  for (int j = v; j >= 0; j--)
    for (int i = 0; i < p[k].size(); i++)
      if (j >= p[k][i].w)
        dp[j] = max(dp[j], dp[j - p[k][i].w] + p[k][i].v);
cout << dp[v] << endl;
```

#1315、混合背包

题目描述

一个旅行者有一个最多能装 V 公斤的背包,现在有 n 件物品,它们的重量分别是 $W_1, W_2, \ldots, W_n$,它们的价值分别为 $C_1, C_2, \ldots, C_n$

有的物品只可以取一次(01 背包),有的物品可以取无限次(完全背包),有的物品可以取的次数有一个上限(多重背包)

求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量,且价值总和最大

输入格式

第一行:二个整数, M (背包容量, $M \leq 200$), N (物品数量, $N \leq 30$)

第 $2 \sim N+1$ 行:每行三个整数 W_i, C_i, P_i ,前两个整数分别表示每个物品的重量,价值,第三个整数若为 0 ,则说明此物品可以购买无数件,若为其他数字,则为此物品可购买的最多件数(P_i)

输出格式

仅一行,一个数,表示最大总价值

提示

选第一件物品 1 件和第三件物品 2 件

输入样例

```
10 3
2 1 0
3 3 1
4 5 4
```

输出样例

```
11
```

#1315、混合背包

对于每种物品枚举体积

分别按照 01/多重/完全背包 的方式转移即可

```
for (int i = 1; i <= n; i++)
{
    if (!p[i]) //完全背包
        for (int j = w[i]; j <= v; j++)
            dp[j] = max(dp[j - w[i]] + c[i], dp[j]);
    else //多重背包
        for (int j = v; j >= 0; j--)
            for (int k = 0; k <= p[i] && j >= k * w[i]; k++)
                dp[j] = max(dp[j - k * w[i]] + c[i] * k, dp[j]);
}
```

#1384、Knapsack 2

题目描述

有 N 个物品，它们的编号分别是 $1, 2, \ldots, N$,其中第 i 个物品的重量是 w_i ,价值是 v_i 。
已知背包的所能装的最大重量是 W ,也就是说选择的所有物品的重量之和不能超过 W 。
可以选择一些物品放入背包中,求所能获得的最大的物品价值之和。

输入格式

第一行包含两个正整数 N 和 W ，分别表示物品的个数和背包能装的最大重量。
接下来 N 行，每行包含两个正整数 w_i 和 v_i ，分别表示第 i 件物品的重量和价值。

输出格式

输出一个整数 ans ，表示Roundgod所能获得的最大的物品价值之和。

数据规模与约定

对于 30% 的数据，有 $1 \le N \le 10, 1 \le W \le 100, 1 \le w_i, v_i \le 100$
对于 100% 的数据，有 $1 \le N \le 100, 1 \le W \le 10^9, 1 \le w_i \le W, 1 \le v_i \le 10^3$

对于本题发现背包容量很大,而$\sum_{i=1}^{n} v_i$较小

该问题可转化为花同样的钱买到重量最小的物品

设$dp[i][j]$为仅考虑前i件物品,价值和不超过j的最小重量

$$dp[i][j] = \min(dp[i-1][j], dp[i-1][j-v_i] + w_i)$$

最终答案为

$$\max_{dp[n][j] \le W} j$$

因为价值和是 $O(N\max(v_i))$,总复杂度为 $O(N^2\max(v_i))$

#502、石板划分

题目描述

有价值分别为 $1 \sim 6$ 的大理石各 $a_{1\sim6}$ 块，现要将它们分成两部分，使得两部分价值之和相等，问是否可以实现。

其中大理石的总数不超过 20000 。

输入格式

输入包含多组数据！

每组数据占一行，包含 6 个整数，表示 $a_1 \sim a_6$ 。

当输入为 `0 0 0 0 0 0` 时表示输入结束，且该行无需考虑。

输出格式

每组数据输出一个结果，每个结果占一行。

如果可以实现则输出 `Can` ，否则输出 `Can't`

输入样例

```
4 7 4 5 9 1
9 8 1 7 2 4
6 6 8 5 9 2
1 6 6 1 0 7
5 9 3 8 8 4
0 0 0 0 0 0
```

若$2 \nmid sum$或$\max(a_i) > \frac{sum}{2}$

显然无法平分

$dp[j]$设为$bool$类型

若$dp[\frac{sum}{2}]$为$true$那么可以平分

将物品二进制拆分多重背包求解即可

#1904、存钱罐

题目描述

Mas 是个不懂得存钱的人，每次花钱如流水。

于是他的好友 $Moen$ 送给他一个存钱罐，这个存钱罐存钱是不可逆的，只能往里存钱不能取钱，如果想取钱除非打破存钱罐。可这存钱罐又是 $Moen$ 送的，Mas 自然就可以存下钱来。

经过半年存款后，Mas 已经存了不少的钱了，但是又不知道这个存钱罐里有多少钱。Mas 知道存钱罐的初始重量和现在的重量，以及每种硬币的面额和重量。

于是 Mas 就找到了聪明的你来帮忙计算，这个存钱罐最少已经有了多少钱呢?

输入格式

输入有多组数据

第一行输入一个整数 $T(1 \le T \le 10)$

每组第一个行输入两个整数 e, f $(1 \le e \le f \le 10000)$ ，分别表示存钱罐的初始重量和现在重量。

每组第二行输入一个整数 n $(1 \le n \le 500)$ ，表示有 n 种硬币。

接下来有 n 行输入，每行有两个整数 p, w $(1 \le p \le 50000, 1 \le w \le 10000)$ ，分别表示硬币的面额和硬币的重量。

输出格式

如果这些钱可以凑出存钱罐的重量，那么请输出存钱罐里最少有多少钱?

如果不能，请输出 `impossible.` 。

样例输入

```
1
1 6
2
10 3
20 4
```

样例输出

```
impossible.
```

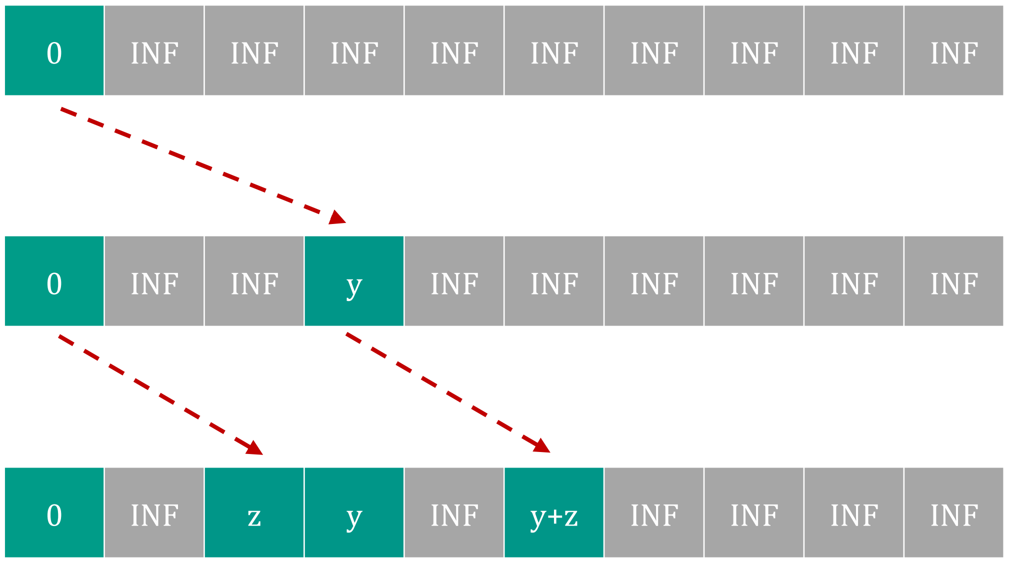
#1904、存钱罐

求解重量为b且恰好的最小价值

对于恰好装满,仅需要在初始时将dp数组全部置为$+\infty$,$dp[0]$置为0

对于恰好装满的状态其值小于$+\infty$,那么这使得每一个状态都从前一阶段的恰好装满的状态转移过来

直接完全背包求解即可



#2843、背包问题求具体方案

题目描述

有 N 件物品和一个容量是 V 的背包。每件物品只能使用一次。

第 i 件物品的体积是 v_i ，价值是 w_i 。

求解将哪些物品装入背包，可使这些物品的总体积不超过背包容量，且总价值最大。

输出 字典序最小的方案。这里的字典序是指：所选物品的编号所构成的序列。物品的编号范围是 $1 \ldots N$ 。

输入格式

第一行两个整数，N，V ，用空格隔开，分别表示物品数量和背包容积。

接下来有 N 行，每行两个整数 v_i, w_i ，用空格隔开，分别表示第 i 件物品的体积和价值。

输出格式

输出一行，包含若干个用空格隔开的整数，表示最优解中所选物品的编号序列，且该编号序列的字典序最小。
物品编号范围是 $1 \sim N$ 。

数据范围

对于全部的数据 $0 < N, V \leq 1000, 0 < v_i, w_i \leq 1000$

输入样例

```
4 5
1 2
2 4
3 4
4 6
```

输出样例

```
1 4
```

#2843、背包问题求具体方案

若$dp[i-1][j]=dp[i-1][j-w_i]+v_i$说明物品$i$可选可不选

若$dp[i-1][j]\neq dp[i-1][j-w_i]+v_i$

- $dp[i][j]=\ dp[i-1][j-w_i]+v_i$说明此时必选了第$i$件物品
- $dp[i][j]=\ dp[i-1][j]$说明此时没有选第i件物品

从$n\rightarrow 1$检查dp数组初始时令$x=W$,若 $dp[i][x]=\ dp[i-1][x-w_i]+v_i$此时输出$i,x$减去$w_i$,可以得出一组最优选取方案

对于需要序号最小方案,从$n\rightarrow 1$递推求出dp数组,贪心构造方案

从$1\rightarrow n$检查dp数组初始时令$x=W$

对于必选物品i时,输出物品i

对于可选可不选状态也输出物品i

若需要数组序号最大的方案如何处理?若需要数组物品中重量序列最小的方案如何处理?

实验舱
青少年编程
走近科学 走进名校

谢谢观看