实验舱
青少年编程
走近科学 走进名校

提高算法班

线性DP、背包DP

Mas

动态规划

动态规划(Dynamic Programming) 是一种通过把原问题分解为相对简单的子问题的方式求解复杂问题的方法

能用动态规划解决的问题，需要满足三个条件:

最优子结构，**无后效性** 和 **子问题重叠**

对于一个能用动态规划解决的问题,一般采用如下思路解决:

- 将原问题划分为若干 **阶段**，每个阶段对应若干个子问题,提取这些子问题的特征 (称之为 **状态**)
- 寻找每一个状态的可能 **决策**，或者说是各状态间的相互转移方式 (状态转移方程)
- 按顺序求解每一个阶段的问题

用图论的思想理解，建立一个 DAG，每个状态对应图上一个节点，决策对应节点间的连边

问题就转变为了一个在 DAG 上寻找最长/短路的问题

动态规划

最优子结构

具有最优子结构也可能适合用贪心求解，要确保考察了最优解中用到的所有子问题

- 证明问题最优解的第一个组成部分是做出一个选择 (存在子结构)
- 对于给定问题，在其可能的第一步选择中，假定已知哪种选择会得到最优解
- 给定可获得的最优解的选择后，确定这次选择会产生哪些子问题，以及如何最好地刻画子问题空间
- 证明作为构成原问题最优解的组成部分，每个子问题的解就是它本身的最优解方法

下面给出两个具体的例子

最短路

设 $p =< v_1, v_2, \cdots, v_k >$ 为 $v_1 \rightarrow v_k$ 的最短路径

对于任意的 $1 \leq i \leq j \leq k$，设 $p_{i,j} =< v_i, v_{i+1}, \cdots, v_j >$ 为 p 的子路径

那么 $p_{i,j}$ 是 $v_i \rightarrow v_j$ 的最短路径

动态规划

证明

将路径 p 分解为 $v_1 \xrightarrow{p_{1,i}} v_i \xrightarrow{p_{i,j}} v_j \xrightarrow{p_{j,k}} v_k$，那么

$$\mathrm{w}(p) = \mathrm{w}(p_{1,i}) + \mathrm{w}(p_{i,j}) + \mathrm{w}(p_{j,k})$$

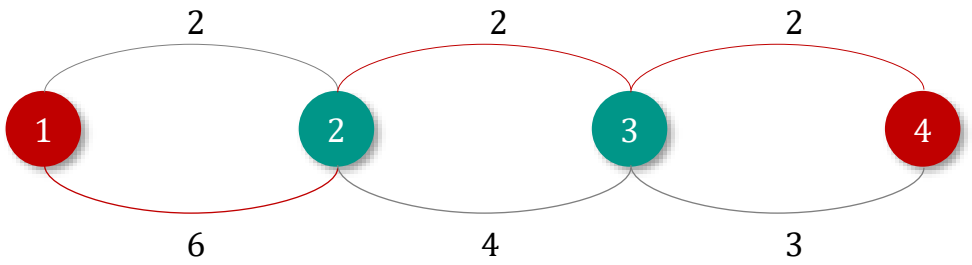
若存在 $p_{i,j}'$ 且 $\mathrm{w}(p_{i,j}) < \mathrm{w}(p_{i,j})$ 则有

$$\mathrm{w}(p') = \mathrm{w}(p_{1,i}) + \mathrm{w}(p_{i,j}') + \mathrm{w}(p_{j,k}) < \mathrm{w}(p) = \mathrm{w}(p_{1,i}) + \mathrm{w}(p_{i,j}) + \mathrm{w}(p_{j,k})$$

与条件矛盾，命题得证

模意义下最短路

要求总长模 10 意义下 $1 \to 4$ 的最短路



动态规划

$p_{1,4}$ 由 $p_{1,2}$ 和 $p_{2,4}$ 组成

$p_{1,2}$ 最优解为 2，$p_{2,4}$最优解为 4

但 w(1 , 4) 最优解为 0

无后效性

已经求解的子问题，不会再受到后续决策的影响。如下给出一个反例

对于 n × m 的网格，初始时位于 (1,1)

一次允许**上下右**移动一步但不可重复到达，请统计到达 (n, m) 的方案数

对于上述问题

位于 $(i-1,j)$ 时是否向下移动到达 (i,j) 将影响 **位于 $(i+1,j)$ 是否可以向上移动到达 (i,j)**

子问题重叠

若有大量的重叠子问题，可用空间将子问题的解记录，避免重复求解相同的子问题

最长上升子序列

- $O(n^2)$ 做法

  设 $\mathrm{dp}[i]$ 表示以 A_i 为结尾 LIS 长度，答案为 $\max\limits_{1 \le i \le n}(\mathrm{dp}[i])$

  将 A_i 接至其它最长不下降子序列后以更新答案

$$\mathrm{dp}[i] = \max_{1 \le j < i \wedge \mathrm{A}_j < \mathrm{A}_i}(\mathrm{dp}[j] + 1)$$

- $O(n \log n)$ 做法

  对于相同长度 LIS 末尾元素越小越容易使得后续序列变长

  令 d_{len} 记录长度为 len 的 LIS 末尾元素

  初始时 $d_1 = a_1, \mathrm{len} = 1$考虑新元素 A_i

  - 若 $\mathrm{A}_i > d_{\mathrm{len}}$ 直接将 A_i 插入到 d 序列末尾
  - 若 $\mathrm{A}_i \le d_{\mathrm{len}}$ 找到 **第一个大于等于** A_i 的元素插入 (丢弃在它之后的全部元素)

  也可考虑线段树/树状数组优化求区间(前缀)最值

#2619、树上选点

题目描述

有一棵 n 个节点的二叉树, 1 为根节点,每个节点有一个值 w_i

现在要选出尽量多的点，对于任意一棵子树都要满足：

- 如果选了根节点的话,在这棵子树内选的其他的点都要比根节点的值大
- 如果在左子树选了一个点,在右子树中选的其他点要比它小

输入描述

第一行一个整数 n

第二行 n 个整数 w_i ,表示每个点的权值

接下来 n 行,每行两个整数 a, b

第 $i+2$ 行表示第 i 个节点的左右儿子节点(没有为 0)

输出描述

一行一个整数表示答案

数据规模

对于 10% 的数据 $1 \leq n \leq 10$

对于 70% 的数据 $n, a, b \leq 5000$

对于 100% 的数据 $n, a, b \leq 10^5, -2 \times 10^9 \leq w_i \leq 2 \times 10^9$

对树进行 **根右左** 顺序的 DFS

求出其 DFS 序列

对 DFS 序列求 LIS 即可

#2713、又是最长上升子序列

题目描述

一个数的序列 b_i

当 $b_1 < b_2 < \ldots < b_S$ 的时称这个序列是上升的

对于给定的一个序列 $a_1, a_2, \ldots, a_N$,我们可以得到一些上升的子序列

$$a_{i_1}, a_{i_2}, \ldots, a_{i_K}$$

其中

$$1 \leq i_1 < i_2 < \ldots < i_K \leq N$$

比如,对于序列 $(1, 7, 3, 5, 9, 4, 8)$,有它的一些上升子序列,如 $(1, 7), (3, 4, 8)$ 等等

这些子序列中最长的为 $(1, 3, 5, 9)$

现在请你求出最长上升子序列的长度,同时输出具体的子序列

输入格式

输入的第一行是序列的长度

第二行给出序列中的 N 个整数

输出格式

第一行最长上升子序列的长度

接下来输出若干个数,表示具体的子序列,使用空格分隔

如果有多组可行解,输出最小的解方案

> 若 A 比 B 小
> 是指存在 $k \geq 1$ 使得 $A_i = B_i$ 对所有 $i < k$ 成立,并且 $A_k < B_k$

数据规模

其中 10% 的数据,子序列方案唯一

对于前 10% 的数据 $1 \leq n \leq 20$

对于前 60% 的数据 $1 \leq n \leq 10000$

对于 100% 的数据 $1 \leq n \leq 10^5, -10^9 \leq a_i \leq 10^9$

#2713、又是最长上升子序列

以 $O(n \log n)$ 代价求出 LIS 同时维护 dp 数组，记 LIS 长度为 X

从后往前遍历 dp 数组

若 $\mathrm{dp}[i] = \mathrm{X}$ 说明 a_i 为答案中元素并令 $\mathrm{X} \leftarrow \mathrm{X} - 1$

证明

若上述做法求出的并非最小的 LIS

即对于某一个方案 $p_{i_1}, p_{i_2}, p_{i_3} \cdots p_{i_j} \cdots p_{i_s}$ 存在 $i_{j-1} < i_{j*} < i_j$ 使得 $p_{i_{j*}} < p_j$

显然 $\mathrm{dp}[i_{j-1}] + 1 = \mathrm{dp}[i_{j*}] = \mathrm{dp}[i_j] = \mathrm{dp}[i_{j+1}] - 1$

由于 $p_{i_{j*}} < p_j$ 那么 p_j 可接至 $p_{i_{j*}}$ 后 ，即 $\mathrm{dp}[i_{j*}] + 1 = \mathrm{dp}[i_j]$

与 $\mathrm{dp}[i_{j*}] = \mathrm{dp}[i_j]$ 矛盾

即该方式所求为最小方案

#2677、还是最长上升子序列

题目描述

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$
请求出它的最长上升子序列的长度
以及有多少个位置上的元素可能出现在最长上升子序列中
多少个位置上的元素一定出现在最长上升子序列中?

如给定序列 $3, 1, 2, 5, 4$
$1, 2, 5$ 与 $1, 2, 4$ 均为满足条件的最长上升子序列,该序列的最长上升子序列的长度为 3
元素 $1, 2, 4, 5$ 均有可能出现在最长上升子序列中,故有 4 个位置上的元素可能出现在最长上升子序列中
而元素 $1, 2$ 必然出现在最长上升子序列中,故有 2 个位置上的元素一定出现在最长上升子序列中

输入格式

第一行输入一个整数 n

第二行输入 n 个正整数,分别表示 $a_1, a_2, \ldots, a_n$

输出格式

输出三个整数数
分别表示最长上升子序列长度
可能出现在最长上升子序列中的元素位置个数
必然出现在最长上升子序列中元素位置个数

数据范围

对于 20% 的数据, $1 \le n \le 10$
对于 70% 的数据, $1 \le n \le 10^4$
对于 100% 的数据, $1 \le n \le 10^5, 1 \le a_i \le 10^9$

记 LIS 长度为 X

dp[i] 为以 a_i **结尾** LIS 长度

rdp[i] 为以 a_i **开头** LIS 长度

对于 rdp[i] 可逆序对 a 求最长下降子序列

若 a_i 为 LIS 中可能的元素

那么有 dp[i] + rdp[i] = X + 1

在可能元素中若 dp[i] 唯一 , 其为 LIS 的必然元素

时间复杂度 $O(n \log n)$

#570、传纸条

题目描述

小渊和小轩是好朋友也是同班同学,他们在一起总有谈不完的话题

一次素质拓展活动中,班上同学安排做成一个 m 行 n 列的矩阵,而小渊和小轩被安排在矩阵对角线的两端,因此,他们就无法直接交谈了

幸运的是,他们可以通过传纸条来进行交流

纸条要经由许多同学传到对方手里,小渊坐在矩阵的左上角,坐标 $(1,1)$,小轩坐在矩阵的右下角,坐标 (m,n)

从小渊传到小轩的纸条只可以向下或者向右传递,从小轩传给小渊的纸条只可以向上或者向左传递

在活动进行中,小渊希望给小轩传递一张纸条,同时希望小轩给他回复

班里每个同学都可以帮他们传递,但只会帮他们一次,也就是说如果此人在小渊递给小轩纸条的时候帮忙,那么在小轩递给小渊的时候就不会再帮忙,反之亦然

还有一件事情需要注意,全班每个同学愿意帮忙的好感度有高有低,可以用一个 $0 \sim 100$ 的自然数来表示,数越大表示越好心

> 小渊和小轩的好心程度没有定义,输入时用 0 表示

小渊和小轩希望尽可能找好心程度高的同学来帮忙传纸条,即找到来回两条传递路径,使得这两条路径上同学的好心程度之和最大

现在,请你帮助小渊和小轩找到这样的两条路径

输入格式

第一行有 2 个用空格隔开的整数 m 和 n ,表示班里有 m 行 n 列

接下来的 m 行是一个 $m \times n$ 的矩阵,矩阵中第 i 行 j 列的整数表示坐在第 i 行 j 列的学生的好心程度

每行的 n 个整数之间用空格隔开

输出格式

共一行,包含一个整数

表示来回两条路上参与传递纸条的学生的好心程度之和的最大值

数据规模

对于 30% 的数据满足: $1 \leq m,n \leq 10$

对于 100% 的数据满足: $1 \leq m,n \leq 50$

先求一遍最大路径和

回溯路径并置为极小值，再求一遍最大路径和?

第二次 DP 的状态受限于第一次 DP (后效性)

该问题等价于求两条不相交路径和最大值 (都只能向下/右)

第一条路径到达 (x_1,y_1) 第二条路径到达 (x_2,y_2)

设 $\mathrm{dp}[x_1][y_1][x_2][y_2]$ 为最大路径和

- 第一条路径到达 (x_1,y_1) 可从 (x_1-1,y_1) 到达，也可从 (x_1,y_1-1) 到达
- 第二条路径到达 (x_2,y_2) 可从 (x_2-1,y_2) 到达，也可从 (x_2,y_2-1) 到达

以上四种情况取最大值加上 $w[x_1][y_1]+w[x_2][y_2]$

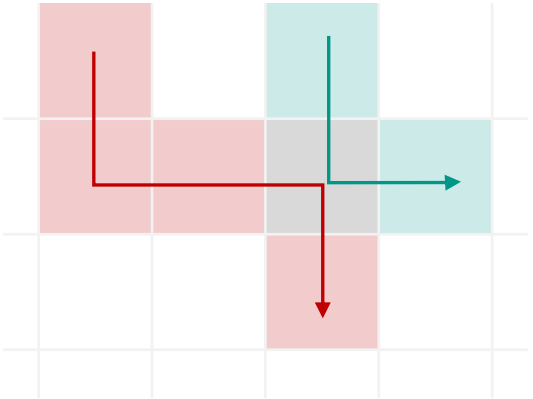
#570、传纸条

若 $x_1 = x_2 \wedge y_1 = y_2$ 说明两条路径重合

减去重复部分即可避免重合的影响

若路径都到达 (x, y) 接下来分别到达 $(x_1^*, y_1^*), (x_2^*, y_2^*)$

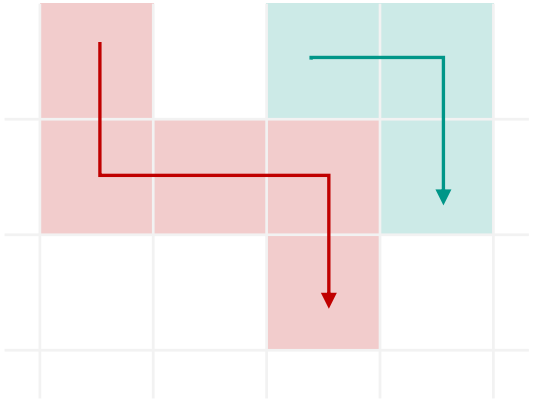
设 $\mathrm{dp}[x][y][x][y]$ 从 $\mathrm{dp}[x_1][y_1][x_2][y_2]$ 转移得到



此时路径和为

$$S_1 = \mathrm{dp}[x_1][y_1][x_2][y_2] + w[x][y] + w[x_1^*][y_1^*] + 0 + w[x_2^*][y_2^*]$$

必存在两条路径 $(x_1, y_1) \to (x, y) \to (x_1^*, y_1^*), (x_2, y_2) \to (x_2', y_2') \to (x_2^*, y_2^*)$



那么此时路径和为

$$S_2 = \mathrm{dp}[x_1][y_1][x_2][y_2] + w[x][y] + w[x_1^*][y_1^*] + w[x_2'][y_2'] + w[x_2^*][y_2^*]$$

显然 $S_2 \geq S_1$，即重合路径必然不优于不重合路径，在取最大值时不被考虑

时/空间复杂度 $O(n^2m^2)$

#570、传纸条

不难发现对答案有贡献的两条路径步数一致

对于一个位置 x 若已知其所走步数 k 那么其坐标为 $(x, k-x+1)$

不妨将状态记为 $dp[k][x_1][x_2]$ 表示两条路径其中一条走到 $(x_1, k-x_1)$ 另一条走到 $(x_2, k-x_2)$ 的路径和最大值

$$\mathrm{dp}[k][x_1][y_1] = \max\begin{pmatrix} \mathrm{dp}[k-1][x_1-1][y_1] \\ \mathrm{dp}[k-1][x_1-1][y_1-1] \\ \mathrm{dp}[k-1][x_1][y_1-1] \\ \mathrm{dp}[k-1][x_1][y_1] \end{pmatrix} + a[x_1][k-x_1+1] + a[x_2][k-x_2+1]$$

第 k 步仅与 $k-1$ 步相关滚动数组优化

时间复杂度 $O\big((n+m)\times n^2\big)$, 空间复杂度 $O(n^2)$

若传递 k 次可考虑费用流求解

#2228、方格取数

题目描述

设有 $n \times m$ 的方格图，每个方格中都有一个整数。现有一只小熊，想从图的左上角走到右下角，每一步只能向上、向下或向右走一格，并且不能重复经过已经走过的方格，也不能走出边界。小熊会取走所有经过的方格中的整数，求它能取到的整数之和的最大值。

输入格式

第 1 行两个正整数 n , m 。

接下来 n 行每行 m 个整数，依次代表每个方格中的整数。

输出格式

一个整数，表示小熊能取到的整数之和的最大值。

输出时每行末尾的多余空格，不影响答案正确性

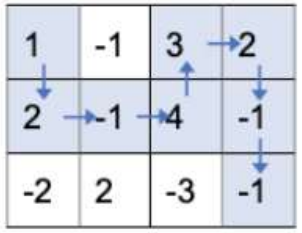
样例输入1

```
3 4
1 -1 3 2
2 -1 4 -1
-2 2 -3 -1
```

样例输出1

```
9
```

样例解释1



按上述走法,取到的数之和为 $1+2+(-1)+4+3+2+(-1)+(-1)=91+2+(-1)+4+3+2+(-1)+(-1)=9$,可以证明为最大值

数据范围与提示

对于 20% 的数据， $n,m \leq 5$ 。

对于 40% 的数据， $n,m \leq 50$ 。

对于 70% 的数据， $n,m \leq 300$ 。

对于 100% 的数据， $1 \leq n,m \leq 1000$ 。方格中整数的绝对值不超过 10^4 。

#2228、方格取数

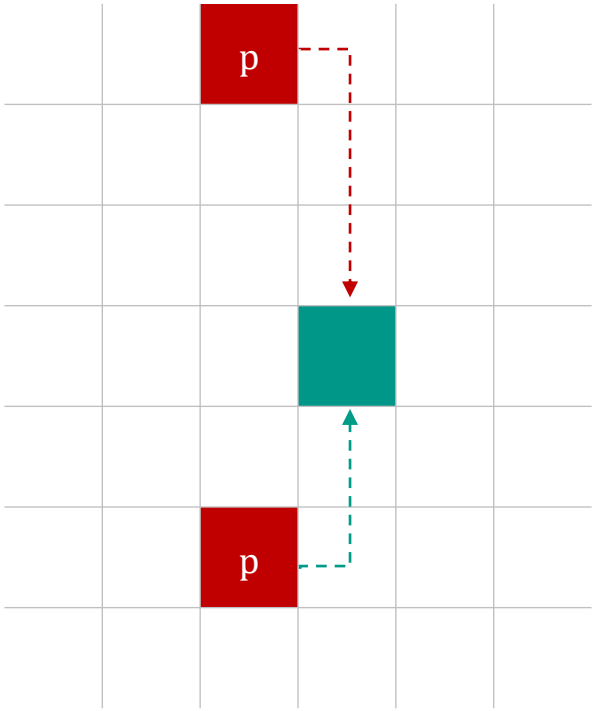
题目限制不允许重复经过，某一列不可能同时出现向上向下

若 $\mathrm{dp}[x][y]$ 表示到达 (x,y) 的最大路径和，从三个方向转移存在后效性

令 $\mathrm{dp}[x][y][0]$ 为最后一步仅考虑向 下/右 到达 (x,y) 的最大路径和

令 $\mathrm{dp}[x][y][1]$ 为最后一步仅考虑向 上/右 到达 (x,y) 的最大路径和

枚举前一列向右移动的的行号，需要计算一列中连续某几行的和



$$\mathrm{dp}[x][y][0]=\max_{1\le i\le x}\left(\max\begin{pmatrix}\mathrm{dp}[p][y-1][0]\\ \mathrm{dp}[p][y-1][0]\end{pmatrix}+\sum_{i=p}^{x}a[i][y]\right)$$

$$\mathrm{dp}[x][y][1]=\max_{x\le i\le n}\left(\max\begin{pmatrix}\mathrm{dp}[p][y-1][0]\\ \mathrm{dp}[p][y-1][0]\end{pmatrix}+\sum_{i=x}^{p}a[i][y]\right)$$

#2228、方格取数

应当以列作为阶段

可使用前缀和优化，时间复杂度 $O(n^2m)$

上述枚举过程中存在大量重复子问题，考虑优化

(x,y) 仅可从 $(x,y-1)$ 向右走到达，从 $(x-1,y)$ 向下走到达，从 $(x+1,y)$ 向上走到达

其三个方向的状态已包含所有前一列所有走法的最大值

因此无需枚举前一列的行号

$$\mathrm{dp}[x][y][0] = \max(\mathrm{dp}[x-1][y][0], \mathrm{dp}[x][y-1][0], \mathrm{dp}[x][y-1][1]) + a[i][j]$$

$$\mathrm{dp}[x][y][1] = \max(\mathrm{dp}[x+1][y][1], \mathrm{dp}[x][y-1][0], \mathrm{dp}[x][y-1][1]) + a[i][j]$$

对于 $\mathrm{dp}[x][y][1]$ 中 x 应当从大到小递推 (从下往上)

时间复杂度 $O(nm)$

01背包

有 N 件物品和一个容量为 W 的背包,第 i 件物品重量 w_i 价值 v_i, 装入背包物品重量总和不超过 W 时价值最大是多少

设 $\mathrm{dp}[i][j]$ 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

状态转移方程

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

观察发现 $\mathrm{dp}[i]$ 仅与 $\mathrm{dp}[i-1]$ 有关 , 可使用两个数组交替滚动节省内存

再进一步观察

对于每一个 $\mathrm{dp}[i][j]$ 仅与 $\mathrm{dp}[i][j-w_i]$ 有关

若改变枚举容量的顺序 (从大到小) 可仅保留一维

时间复杂度 $O(\mathrm{NW})$, 空间复杂度 $O(\mathrm{W})$

#2472、反复背包

题目描述

有 n 个物品,编号 0 至 $n-1$,每个物品有重量 a_i 和价值 b_i

q 次询问,每次不能选第 d_i 个物品,问容量为 e_i 的背包能收获的最大价值

输入格式

第一行一个整数 n

接下来 n 行,每行两个整数 a_i 和 b_i 表示物品

接下来一行一个整数 q

接下来 q 行,每行两个整数 d_i 和 e_i 表示询问

输出格式

q 行,每行一个整数表示对应询问的答案

数据范围

对于 20% 的数据,有 $n \leq 10$

对于 40% 的数据,有 $n \leq 100, q \leq 100$

对于 100% 的数据,有 $n \leq 10^3, q \leq 10^5, 1 \leq a_i, b_i \leq 100, 0 \leq d_i < n, 0 \leq e_i \leq 10^3$

设 $\mathrm{dp}_1[i][j]$ 为仅考虑 $1 \sim i$ 件物品背包容量为 j 的最大价值和

设 $\mathrm{dp}_2[i][j]$ 为仅考虑 $i \sim n$ 件物品背包容量为 j 的最大价值和

dp_2 需逆序递推

对于每一组询问的 d_i, e_i 答案为

$$\max_{0 \leq j \leq e_i} (\mathrm{dp}_1[d_i - 1][j] + \mathrm{dp}_2[d_i + 1][e_i - j])$$

时间复杂度 $O(n \times \max(e_i) + \sum_{i=1}^{q} e_i)$

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 w_i 价值 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 $\mathrm{dp}[i][j]$ 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

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

时间复杂度 $O(\mathrm{W} \sum_{i=1}^{\mathrm{N}} \left\lfloor \frac{\mathrm{W}}{w_i} \right\rfloor)$

当 $\sum_{i=1}^{\mathrm{N}} \left\lfloor \frac{\mathrm{W}}{w_i} \right\rfloor$ 较大时效率较低 , 考虑优化

完全背包

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

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

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

状态转移方程

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

无需再枚举 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 $O(\mathrm{NW})$，空间复杂度 $O(\mathrm{W})$

#1904、存钱罐

题目描述

aMs 是个不懂得存钱的人，每次花钱如流水

saM 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 saM 送的，aMs 自然就可以存下钱来

经过半年存款后，aMs 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

aMs 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 aMs 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

第一行输入一个整数 T，表示 T 组数据

每组第一个行输入两个整数 e, f，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,表示有 n 种硬币

接下来有 n 行输入,每行有两个整数 p, w，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.

数据规模

对于全部的数据 $1 \leq e \leq f \leq 10000, 1 \leq \sum n \leq 5000, 1 \leq p, w \leq 50000$

#1904、存钱罐

求解重量 **恰好** 为 f 的最小价值

问题等价于初始重量为 0 , 最终重量恰为 $f-e$ 的最小价值

需保证恰好装满状态仅由上一阶段恰好装满的状态转移而来

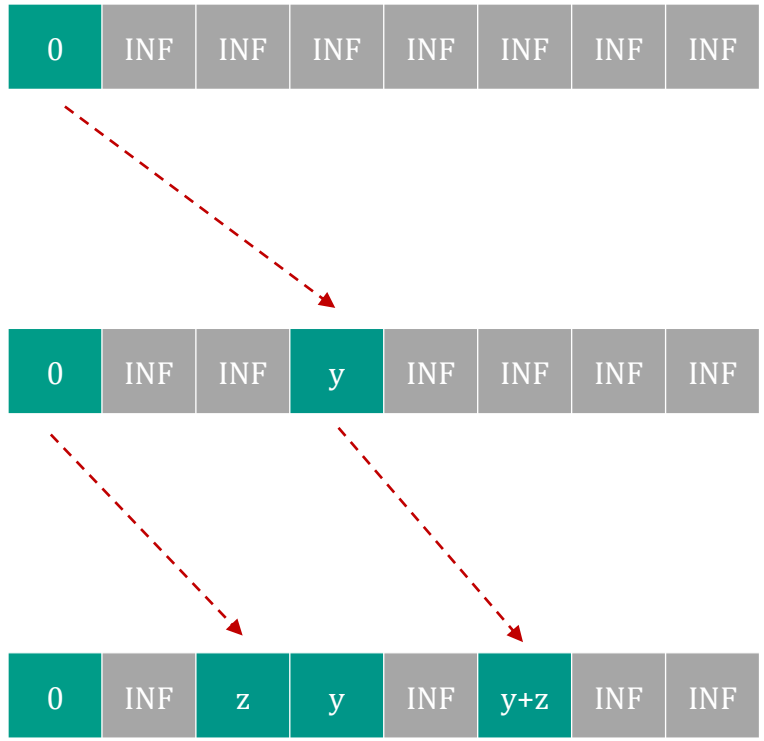
初始时令 $dp \leftarrow \infty$, 再令 $dp[0][0] \leftarrow 0$

即认为初始时仅有容量为 0 的背包合法

由于合法状态值小于 ∞ , 那么合法状态的转移必优于非法状态的转移

不难归纳证明在求解过程中 , 非法状态必不干扰决策

完全背包求解即可



类似的若要求恰好装满的最大价值和

仅需令 $dp \leftarrow -\infty$, 再令 $dp[0][0] \leftarrow 0$

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 w_i 价值 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，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 $O\left(W \sum_{i=1}^{N} K_i\right)$

不难发现当 $\sum_{i=1}^{N} K_i$ 较大时并不优，考虑进一步优化

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 p 为满足 $\left(\sum_{i=0}^{p} 2^i\right) \le K_i$ 的最大整数

显然 $2^0, 2^1, 2^2, \cdots, 2^{p-1}, 2^p$ 可组合出 $0 \sim 2^{p+1} - 1$ 所有整数

多重背包

由于 p 的极大性有 $K_i - 2^{p+1} + 1 < 2^{p+1}$

若 $K_i - 2^{p+1} + 1 \geq 2^{p+1}$ 那么 $K_i \geq 2^{p+2} - 1$ 即 $\left(\sum_{i=0}^{p+1} 2^i\right) \leq K_i$ 这与 p 的极大性矛盾

对于 $2^{p+1} \leq X \leq K_i$ 可 X 将表示为 $K_i - 2^{p+1} + 1 + T$ (即剩余数+ T)

而 T 为 $0 \sim 2^{p+1} - 1$ 范围内整数

若 $T \geq 2^{p+1}$ 那么 $K_i - 2^{p+1} + 1 + T \geq K_i - 2^{p+1} + 1 + 2^{p+1} = K_i + 1$, 这与 $2^{p+1} \leq X \leq K_i$ 矛盾

即 X 可用 $2^0, 2^1, 2^2, \cdots, 2^{p-1}, 2^p$ 与 $K_i - 2^{p+1} + 1$ 搭配得到

综上用 $2^0, 2^1, 2^2, \cdots, 2^{p-1}, 2^p$, $K_i - 2^{p+1} + 1$ 进行 0/1 组合选取 , 可覆盖 $0 \sim K_i$ 范围内整数

而该种拆分方式 K_i 件物品拆分成约 $\log_2 K_i$ 组重量、价值不一的物品

时间复杂度 $O\left(W \times \left(\sum_{i=1}^{N} \log_2 K_i\right)\right)$

可使用单调队列进一步优化至 O(NW)

二维费用背包 & 分组背包

有 N 种物品和一个承重量为 W 体积为 V 背包，第 i 件物品的重量 w_i 体积 v_i 价值 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 个物品承重量为 j 体积为 k 的背包的最大总价值，类比于 01 背包增加一个体积维度

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

时间复杂度 $O(\mathrm{NWV})$

有 N 个物品分为 T 组及承重量为 W 的背包，第 i 件物品的重量 w_i 价值 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 $\mathrm{dp}[i][j]$ 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

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

时间复杂度 $O(\mathrm{NW})$

#2842、背包最优方案数

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 **最优选法的方案数**

注意答案可能很大,请输出答案 $\mod 1000000007$ 的结果

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

输出一个整数,表示 方案数 $\mod 1000000007$ 的结果

数据范围

对于全部的数据 $1 \leq N, V \leq 1000, 1 \leq v_i, w_i \leq 1000$

#2842、背包最优方案数

思路1

令 $\mathrm{dp}[i][j]$ 表示考虑前 i 件物品，物品总重不超过 j 的最大价值和

令 $\mathrm{cnt}[i][j]$ 表示考虑前 i 个物品背包容量为 j 时最优方案数

考虑在 $\mathrm{dp}[i][j]$ 求解过程中维护 $\mathrm{cnt}[i][j]$

初始时令 $\mathrm{cnt}[0][0 \sim \mathrm{W}] \longleftarrow 1$

若 $\mathrm{dp}[i-1][j] < \mathrm{dp}[i-1][j-w_i] + v_i$

说明选取物品 i 更优

令 $\mathrm{cnt}[i][j] \longleftarrow \mathrm{cnt}[i-1][j-w_i]$

若 $\mathrm{dp}[i-1][j] = \mathrm{dp}[i-1][j-w_i] + v_i$

说明最优方案不唯一

令 $\mathrm{cnt}[i][j] \longleftarrow \mathrm{cnt}[i-1][j] + \mathrm{cnt}[i-1][j-w_i]$

#2842、背包最优方案数

cnt[N][W] 即为答案

思路2

令 dp[i][j] 表示考虑前 i 件物品，物品总重为 j 的最大价值和

令 cnt[i][j] 表示考虑前 i 个物品物品总重为 j 时最优方案数

考虑在 dp[i][j] 求解过程中维护 cnt[i][j]

初始时令 **cnt[0][0] ⟵ 1**

若 $\text{dp}[i][j] \neq \text{dp}[i-1][j] \wedge \text{dp}[i][j] = \text{dp}[i-1][j-w_i] + v_i$

说明选取物品 i 更优

令 $\text{cnt}[i][j] \longleftarrow \text{cnt}[i-1][j-w_i]$

若 $\text{dp}[i][j] = \text{dp}[i-1][j] \wedge \text{dp}[i][j] \neq \text{dp}[i-1][j-w_i] + v_i$

说明不选物品 i 更优

#2842、背包最优方案数

令 $\mathrm{cnt}[i][j] \longleftarrow \mathrm{cnt}[i-1][j]$

若 $\mathrm{dp}[i][j] = \mathrm{dp}[i-1][j] \wedge \mathrm{dp}[i][j] = \mathrm{dp}[i-1][j-w_i] + v_i$

说明可选可不选物品 i

令 $\mathrm{cnt}[i][j] \longleftarrow \mathrm{cnt}[i][j] + \mathrm{cnt}[i-1][j]$

$$\left(\sum_{i=o}^{\mathrm{W}} \mathrm{cnt}[n][i] \times \left[\mathrm{dp}[n][i] = \max_{0 \le i \le \mathrm{W}} \{ \mathrm{dp}[n][i] \} \right] \right)$$

即为答案

该做法可求出恰好装满的方案数

两种思路时间复杂度都为 $O(NW)$

#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

输出最小的方案

最小方案是指所选物品的编号所构成的序列最小

> *若 A 比 B 小*
> *是指存在 $k \geq 1$ 使得 $A_i = B_i$ 对所有 $i < k$ 成立,并且 $A_k < B_k$*

物品的编号范围是 $1 \sim N$

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

数据范围

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

#2843、背包问题求具体方案

若要求解具体方案，一般从最终状态倒退求解（起点可能不唯一）

若 $\mathrm{dp}[i][j] \neq \mathrm{dp}[i-1][j] \wedge \mathrm{dp}[i][j] = \mathrm{dp}[i-1][j-w_i] + v_i$，说明选取物品 i 更优

若 $\mathrm{dp}[i][j] = \mathrm{dp}[i-1][j] \wedge \mathrm{dp}[i][j] \neq \mathrm{dp}[i-1][j-w_i] + v_i$，说明不选物品 i 更优

若 $\mathrm{dp}[i][j] = \mathrm{dp}[i-1][j] \wedge \mathrm{dp}[i][j] = \mathrm{dp}[i-1][j-w_i] + v_i$，说明可选可不选物品 i

$n \to 1$ 求解 dp 数组，那么可 $1 \to n$ 倒推求解具体方案

初始时令 $\mathrm{X} = \mathrm{W}$

- 若物品 i 必选时，将物品 i 加入方案令 $\mathrm{X} \leftarrow \mathrm{X} - w_i$
- **若物品 i 可选可不选时，将物品 i 加入方案令 $\mathbf{X} \leftarrow \mathbf{X} - w_i$**

容易归纳证明该方案为最小方案

最大方案？面额最大最小方案？

#2837、 背包第 k 大方案

题目描述

有 N 个物品和一个容量为 W 的背包

每个物品有重量 w_i 和价值 v_i 两种属性

要求选若干物品放入背包使背包中物品的总价值最大且背包中物品的总重量不超过背包的容量

如果仅需要求出最大价值和大家都会,现在需要你求出第 K 大价值和

输出格式

第一行输入一个正整数 T ,表示 T 组数据

每组数据第一行输入一个正整数 N, W, K

每组数据第二行输入 N 个整数表示 v_i

每组数据第三行输入 N 个整数表示 w_i

输出格式

对于每组数据输出一行,输出第 K 大价值和

数据规模

对于全部的数据 $1 \leq T \leq 10, 1 \leq N \leq 100, 1 \leq M \leq 3000, 1 \leq K \leq 50, 1 \leq v_i, w_i \leq 1000$

#2837、 背包第 k 大方案

令 $\mathrm{dp}[i][j]$ 为仅考虑 $1 \sim i$ 件物品背包容量为 j 时的前 K 大价值和 (可使用 set 维护)

合并 $\mathrm{dp}[i-1][j]$ 与 $\mathrm{dp}[i-1][j-w_i]+v_i$ 保留前 K 大即为 $\mathrm{dp}[i][j]$

一组询问时间复杂度 $O(\mathrm{N\ M\ K} \log \mathrm{K})$ 无法通过本题

令 $\mathrm{dp}[i][j][k]$ 为仅考虑 $1 \sim i$ 件物品背包容量为 j 时的第 k 大价值和

$\mathrm{dp}[i-1][j][1 \sim \mathrm{K}]$ 和 $\mathrm{dp}[i-1][j-w_i][1 \sim \mathrm{K}]$ 都分别为有序序列

与二路归并排序类似合并后去重即可

合并时应当取出 $\mathrm{dp}[i-1][j-w_i][k]+v_i$

一组询问时间复杂度 $O(\mathrm{N\ M\ K})$

#1909、纪念品

题目描述

小伟突然获得一种超能力，他知道未来 T 天内 N 种纪念品每天的价格
某个纪念品的价格是指购买一个该纪念品所需的金币数量，以及卖出一个该纪念品换回的金币数量

每天，小伟可以进行以下两种交易**无限次**：

> *任选一个纪念品，若手上有足够金币，以当日价格购买该纪念品，注意同一个纪念品可以在同一天重复买；*
> *卖出持有的任意一个纪念品，以当日价格换回金币。*

每天卖出纪念品换回的金币可以**立即**用于购买纪念品，当日购买的纪念品也可以**当日**卖出换回金币。
当然，一直持有纪念品也是可以的。 T 天之后，小伟的超能力消失。因此他一定会在第 T 天卖出**所有**纪念品换回金币。

小伟现在有 M 枚金币，他想要在超能力消失后拥有尽可能多的金币。

输入格式

第一行包含三个正整数 T, N, M ，相邻两数之间以一个空格分开，分别代表未来天数 T ，纪念品数量 N ，小伟现在拥有的金币数量 M 。

接下来 T 行，每行包含 N 个正整数，相邻两数之间以一个空格分隔。

第 i 行的 N 个正整数分别为 $P_{i1}, P_{i2}, \cdots, P_{iN}$

> P_{ij} *表示第* i *天第* j *种纪念品的价格。*

输出格式

输出仅一行，包含一个正整数,表示小伟在超能力消失后最多能拥有的金币数量

输入样例1

```
6 1 100
50
20
25
20
25
50
```

输入样例2

```
3 3 100
10 20 15
15 17 13
15 25 16
```

输出样例2

```
217
```

输出样例1

```
305
```

数据范围

对于 10% 的数据，$T = 1$ 。

对于 30% 的数据，$T \leq 4, N \leq 4, M \leq 100$ ，所有价格 $10 \leq P_{ij} \leq 100$ 。

对于 15% 的数据，$T \leq 100, N = 1$ 。

对于 15% 的数据，$T = 2, N \leq 100$ 。

对于 100% 的数据，$T \leq 100, N \leq 100, M \leq 10^3$ ，所有价格 $1 \leq P_{ij} \leq 10^4$ ，数据保证任意时刻，小明手上的金币数不可能超过 10^4 。

#1909、纪念品

$T = 1$

答案为 M (不做任何购入)

$T = 2$

将第一天价格 $p[1][j]$ 作为重量(新购入)，差价 $p[2][j] - p[1][j]$ 作为价值(卖出赚取差价)

每种物品在当天可以无限交易，不难看出这是完全背包问题

设 $\mathrm{dp}[i][j]$ 为第 2 天仅考虑前 i 件物品当前持有现金不超过 j

在**当天结束后全部卖出**获得的最大现金收益

$$\mathrm{dp}[i][j] = \max(\mathrm{dp}[i][j], \mathrm{dp}[i][j - p[1][j]] + p[2][j] - p[i][j])$$

可使用滚动数组优化空间复杂度

#1909、纪念品

$T \geq 2$

若某一件物品在第 x 天购入第 y 天卖出收益最优

等价于第 x 天买入,第 $x+1$ 天卖出,第 $x+1$ 天买入,第 $x+2$ 天卖出,…,第 $y-1$ 天买入,第 y 天卖出

即任何一个购买方案可以转换为一系列连续的隔天交易

$$-p[x]+p[x+1]-p[x+1]+p[x+2]+\cdots-p[y-1]+p[y]=-p[x]+p[y]$$

若在相邻两天之内做交易保证获得的收益最大

那么进行 $T-1$ 轮相邻两天交易后，得到的必然为全局最优解

对每一轮进行完全背包求解即可

时间复杂度 $O(TNM)$

#2026、搭建双塔

题目描述

2001 年 9 月 11 日,一场突发的灾难将纽约世界贸易中心大厦夷为平地, $Mr.F$ 曾亲眼目睹了这次灾难

为了纪念 9.11 事件, $Mr.F$ 决定自己用水晶来搭建一座双塔

$Mr.F$ 有 N 块水晶,每块水晶有一个高度,他想用这 N 块水晶搭建两座有同样高度的塔,使他们成为一座双塔

$Mr.F$ 可以从这 N 块水晶中任取一些水晶来搭建
但是他不知道能否使两座塔有同样的高度,也不知道如果能搭建成一座双塔

给定水晶的数量 N 和每块水晶的高度 H_i ,你的任务是判断 $Mr.F$ 能否用这些水晶搭建成一座双塔(两座塔有同样的高度)

如果能,则输出所能搭建的双塔的最大高度,否则输出 `Impossible`

输入格式

输入的第一行为一个数 N ,表示水晶的数量
第二行为 N 个数,第 i 个数表示第 i 个水晶的高度

输出格式

输出仅包含一行
如果能搭成一座双塔,则输出双塔的最大高度,否则输出 `Impossible`

数据规模

对于全部的数据 $1 \leq N \leq 100, \sum_{i=1}^{n} H_i \leq 2000$

设 dp[i][j] 为仅考虑前 i 块水晶

两座塔高度差为 j 时 **较高** 塔高度

可 01 背包求解 , 该问题为恰好装满问题

对于每块水晶有三种选择

#2026、搭建双塔

不放在任何一座塔此时高度差不变

可从 $\mathrm{dp}[i-1][j]$ 转移

如图1，放在高塔上

可从 $\mathrm{dp}[i-1][j-\mathrm{H}_i]+\mathrm{H}_i$ 转移（此时应满足 $j \geq \mathrm{H}_i$）

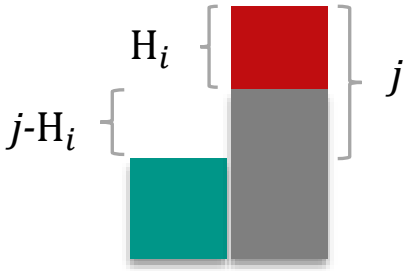


图1

放在矮塔上

如图2，矮塔并没有成为新的高塔，原高塔高度不变

可从 $\mathrm{dp}[i-1][j+\mathrm{H}_i]$ 转移

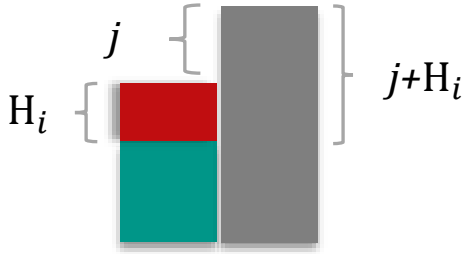


图2

如图3，矮塔并成为了新的高塔

可从 $\mathrm{dp}[i-1][\mathrm{H}_i-j]+j$ 转移（此时应满足 $j < \mathrm{H}_i$）

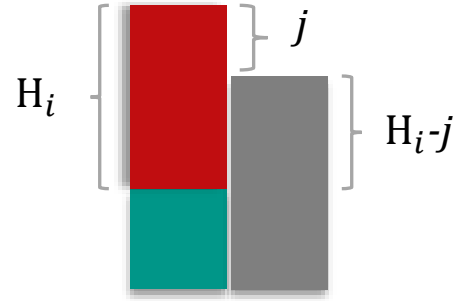


图3

由于**依赖关系交错**，不能逆序递推优化空间

可使用滚动数组优化，时间复杂度 $O(n\ \sum_{i=1}^{n}\mathrm{H}_i)$

谢谢观看