提高算法班

区间DP、DP优化1

Mas

#732、石子合并

题目描述

将 n 堆石子绕圆形操场排放,现要将石子有序地合并成一堆

规定每次只能选相邻的两堆合并成新的一堆,并将新的一堆的石子数记做该次合并的得分

请编写一个程序,读入堆数 n 及每堆的石子数,并进行如下计算:

选择一种合并石子的方案,使得做 $n-1$ 次合并得分总和最大

选择一种合并石子的方案,使得做 $n-1$ 次合并得分总和最小

输入格式

输入第一行一个整数 n ,表示有 n 堆石子

第二行 n 个整数,表示每堆石子的数量

输出格式

输出共两行:
第一行为合并得分总和最小值

第二行为合并得分总和最大值

样例输入

```
4
4 5 9 4
```

样例输出

```
43
54
```

数据范围

对于 100% 的数据,有 $1 \leq n \leq 200$

#732、石子合并

设 $\mathrm{dp}[l][r]$ 为将区间 $[l,r]$ 合并成一堆的最大/小代价

初始时

- 对于 $1 \le l \le n$ 都令 $\mathrm{dp}[l][l] = 0$
- 对于 $1 \le r < l \le n$ 都令 $\mathrm{dp}[l][r] = \infty/-\infty$

由于区间成环，尝试枚举一条边断开

破环成链存在 n 种情况，将每种情况合并成一堆取最值

时间复杂度 $O(n^4)$

不难发现上述做法中对一个区间有多次重复计算

考虑将链延长两倍变成 $2 \times n$ 堆，其中第 i 堆与第 $n+i$ 堆相同

答案为 $\min\limits_{1 \le i \le n} \{\mathrm{dp}[i][i+n-1]\}$

时间复杂度为 $O(n^3)$

#734、凸多边形的划分

题目描述

给定一个具有 N 个顶点的凸多边形

将顶点从 $1 \sim N$ 标号,每个顶点的权值都是一个正整数

将这个凸多边形划分成 $N-2$ 个互不相交的三角形

试求这些三角形顶点的权值乘积和至少为多少

输入格式

输入第一行为顶点数 N

第二行依次为顶点 1 至顶点 N 的权值

输出格式

输出仅一行，为这些三角形顶点的权值乘积和的最小值

数据范围

对于 100% 的数据,有 $N \le 50$,每个点权值小于 10^9

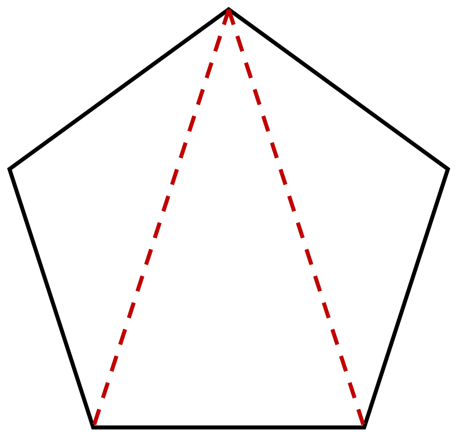
设 $\mathrm{dp}[l][r]$ 为将 $l \sim r$ 划分的最小代价

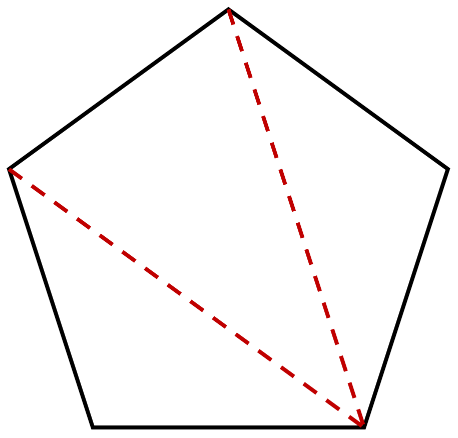
不难发现 $r-l+1$ 至少为 3

$$\mathrm{dp}[l][r] = \min_{l<k<r}\{\mathrm{dp}[l][k] + \mathrm{dp}[k][r] + \mathrm{w}_l \times \mathrm{w}_k \times \mathrm{w}_r\}$$

每个点权值不超过 10^9 三点相乘能达到 10^{27}

需高精度或 int128





#736、分离与合体

题目描述

杜神牛造了 n 个区域,他们紧邻着排成一行,编号 $1 \sim n$

在每个区域里都放着一把 OI 界的金钥匙,每一把都有一定的价值, LYD 当然想得到他们了

然而杜神牛规定 LYD 不能一下子把他们全部拿走,而是每次只可以拿一把

为了尽快得到所有金钥匙, LYD 自然就用上了刚学的分离与合体特技

一开始 LYD 可以选择 $1 \sim n-1$ 中的任何一个区域进入,我们不妨把这个区域记为 k

进入后 LYD 会在 k 区域发生分离,从而分离成两个小 LYD

分离完成的同时会有一面墙在 k 区域和 $k+1$ 区域间升起,从而把 $1 \sim k$ 和 $k+1 \sim n$ 阻断成两个独立的区间,并在各自区间内任选除区间末尾之外(即从 $1 \sim k-1$ 和 $k+1 \sim n-1$ 中选取)的任意一个区域再次发生分离,这样就有了四个小小 LYD ......重复以上所叙述的分离,直到每个小 LYD 发现自己所在的区间只剩下了一个区域,那么他们就可以抱起自己梦寐以求的 OI 金钥匙

但是 LYD 不能就分成这么多个个体存在于世界上,这些小 LYD 还会再合体,合体的小 LYD 所在区间中间的墙会消失

合体会获得 $($ 合并后所在区间左右端区域里金钥匙价值之和 $) \times ($ 之前分离的时候所在区域的金钥匙价值 $)$

例如 LYD 曾在 $1 \sim 3$ 区间中的 2 号区域分离成为 $1 \sim 2$ 和 $3 \sim 3$ 两个区间,合并时获得的价值就是 $($ 1 号金钥匙价值 $+$ 3 号金钥匙价值 $) \times ($ 2 号金钥匙价值 $)$

LYD 请你编程求出最终可以获得的最大总价值,并按照分离阶段从前到后,区域从左到右的顺序,输出发生分离区域编号

若有多种方案,选择分离区域尽量靠左的方案(也可以理解为输出了类似字典序最小的)

例如先打印一分为二的区域,然后从左到右打印二分为四的分离区域,然后是四分为八的......

数据范围

对于 100% 的数据, $n, a_i \leq 300$,保证运算过程和结果不超过 32 位正整数范围

#736、分离与合体

设 $\mathrm{dp}[l][r]$ 为区间 $[l,r]$ 最高得分，答案为 $\mathrm{dp}[1][n]$

枚举$[l,r]$ 区间内的分离点

$$\mathrm{dp}[l][r] = \max_{l \le k < r} \begin{Bmatrix} \mathrm{dp}[l][k] + \mathrm{dp}[k+1][r] \\ + (\mathrm{w}_l + \mathrm{w}_r) \times \mathrm{w}_k \end{Bmatrix}$$

令 $\mathrm{p}[l][r]$ 为 $[l,r]$ 内分离点编号

$\mathrm{p}[l][r]$ 不难在转移过程中维护

对于输出方案需要分层次输出，BFS 输出即可

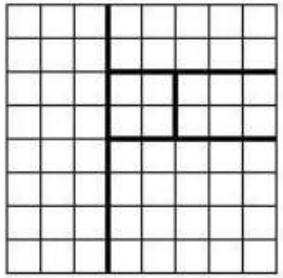
时间复杂度 $O(n^3)$

#2480、棋盘分割

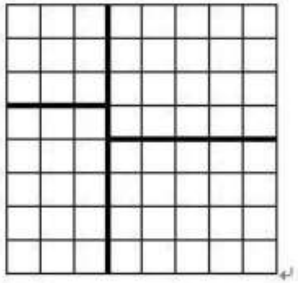
题目描述

将一个 8×8 的棋盘进行如下分割：

将原棋盘割下一块矩形棋盘并使剩下部分也是矩形,再将剩下的部分继续如此分割,这样割了 $n-1$ 次后,连同最后剩下的矩形棋盘共有 n 块矩形棋盘。(每次切割都只能沿着棋盘格子的边进行))



允许的分割方案



不允许的分割方案

原棋盘上每一格有一个分值,一块矩形棋盘的总分为其所含各格分值之和。现在需要把棋盘按上述规则分割成 n 块矩形棋盘,并使各矩形棋盘总分的均方差最小

均方差

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n}}$$

其中平均值 $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$, x_i 为第 i 块矩形棋盘的总分

请编程对给出的棋盘及 n ,求出 σ 的最小值

输入格式

第一行为一个整数 $n(1 < n < 15)$

第二行至第九行每行为 8 个小于 100 的非负整数,表示棋盘上相应格子的分值,每行相邻两数之间用一个空格分隔

输出格式

仅一个数 σ ,四舍五入精确到小数点后三位

#2480、棋盘分割

不难发现 $\bar{x}$ 为定值

$$\bar{x} = \frac{1}{n} \times \sum_{i=1}^{n} x_i = \frac{1}{n} \times \sum_{i=1}^{8} \sum_{j=1}^{8} w_{ij}$$

根据题意

$$\sigma = \sqrt{\frac{1}{n} \times \sum_{i=1}^{n} (x_i - \bar{x})^2}$$

只需要

$$\sigma^2 = \frac{1}{n} \times \sum_{i=1}^{n} (x_i - \bar{x})^2$$

最小，将其拆开

$$\sigma^2 = \frac{1}{n} \times \sum_{i=1}^{n} (x_i - \bar{x})^2 =$$

#2480、棋盘分割

$$\frac{1}{n}\times\sum_{i=1}^{n}(\mathrm{x}_i^2-2\bar{\mathrm{x}}\mathrm{x}_i+\bar{\mathrm{x}}^2)=\frac{1}{n}\times\left(\left(\sum_{i=1}^{n}\mathrm{x}_i^2\right)-\left(2\bar{\mathrm{x}}\sum_{i=1}^{n}\mathrm{x}_i\right)+\left(\sum_{i=1}^{n}\bar{\mathrm{x}}^2\right)\right)$$

$$=\frac{1}{n}\times\left(\left(\sum_{i=1}^{n}\mathrm{x}_i^2\right)-\left(2\bar{\mathrm{x}}\sum_{i=1}^{n}\mathrm{x}_i\right)+n\bar{\mathrm{x}}^2\right)=\frac{1}{n}\times\left(\sum_{i=1}^{n}\mathrm{x}_i^2\right)-\frac{1}{n}\times\left(2\bar{\mathrm{x}}\sum_{i=1}^{n}\mathrm{x}_i\right)+\bar{\mathrm{x}}^2$$

$$=\frac{1}{n}\times\left(\sum_{i=1}^{n}\mathrm{x}_i^2\right)-2\bar{\mathrm{x}}^2+\bar{\mathrm{x}}^2=\frac{1}{n}\times\left(\sum_{i=1}^{n}\mathrm{x}_i^2\right)-\bar{\mathrm{x}}^2$$

设 $\mathrm{dp}[i][x_1][y_1][x_2][y_2]$ 为将矩形区域 $\mathrm{w}[\mathrm{x}_1\sim\mathrm{x}_2][\mathrm{y}_1\sim\mathrm{y}_2]$ 进行 i 次划分的最小矩形权值平方和

答案为 $\sqrt{\mathrm{dp}[n-1][1][1][8][8]-\bar{\mathrm{x}}^2}$

令 $\mathrm{sum}_{\mathrm{x}_1,\mathrm{y}_1,\mathrm{x}_2,\mathrm{y}_2}$ 为矩形区域 $\mathrm{w}[\mathrm{x}_1\sim\mathrm{x}_2][\mathrm{y}_1\sim\mathrm{y}_2]$ 权值和的平方与 n 的比值

对于一个区域有两种分割方式

#2480、棋盘分割

横着分隔,枚举分割线

$$dp[i][x_1][y_1][x_2][y_2] = \min_{x_1 \le k < x_2} \begin{Bmatrix} dp[i-1][x_1][y_1][k][y_2] + sum_{k+1,y_1,x_2,y_2} \\ dp[i-1][k+1][y_1][x_2][y_2] + sum_{x_1,y_1,k,y_2} \end{Bmatrix}$$

其中 $dp[i-1][x_1][y_1][k][y_2] + sum_{k+1,y_1,x_2,y_2}$ 为切出下半部分

其中 $dp[i-1][k+1][y_1][x_2][y_2] + sum_{x_1,y_1,k,y_2}$ 为切出上半部分

竖着分隔,枚举分割线

$$dp[i][x_1][y_1][x_2][y_2] = \min_{y_1 \le k < y_2} \begin{Bmatrix} dp[i-1][x_1][y_1][y_1][k] + sum_{x_1,k+1,x_2,y_2} \\ dp[i-1][x_1][k+1][x_2][y_2] + sum_{x_1,k,x_2,y_2} \end{Bmatrix}$$

其中 $dp[i-1][x_1][y_1][y_1][k] + sum_{x_1,k+1,x_2,y_2}$ 为切出左半部分

其中 $dp[i-1][x_1][k+1][x_2][y_2] + sum_{x_1,k,x_2,y_2}$ 为切出右半部分

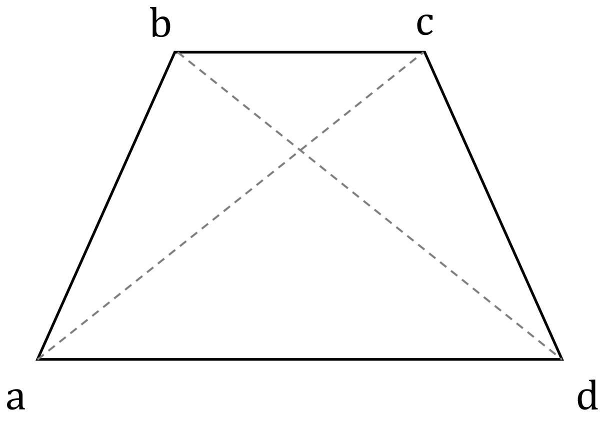
从小区间往大区间枚举，时间复杂度 $O(8^5 n)$

四边形不等式优化

在区间类动态规划中状态转移方程常为如下形式:

$$\mathrm{dp}[l][r]=\begin{cases}0, & l=r\\ \min\limits_{l\le k<r}\left\{\begin{array}{c}\mathrm{dp}[l][k]+\mathrm{dp}[k+1][r]\\ +\,\mathrm{w}(l,r)\end{array}\right\}, & l<r\\ \infty, & l>r\end{cases}$$

朴素转移时间复杂度为 $O(n^3)$



当函数 $\mathrm{w}(l,r)$ 满足一些特殊的性质时可利用决策单调性进行优化

区间包含单调性

$\forall\, l\le l'\le r'\le r$ 有 $\mathrm{w}(l',r')\le \mathrm{w}(l,r)$ 则称函数 w 对于区间包含关系具有单调性

四边形不等式

$\forall\, a\le b\le c\le \mathrm{d}$ 有 $\mathrm{w}(a,c)+\mathrm{w}(b,d)\le \mathrm{w}(a,d)+\mathrm{w}(b,c)$ 则称函数 w 满足四边形不等式 (简记 **交叉小于包含**)

若等号永远成立则称函数 w 满足 **四边形恒等式**

四边形不等式优化

四边形不等式另一表述形式：

$\forall\, a < b$ 有 $\mathrm{w}(a,b+1)+\mathrm{w}(a+1,b) \geq \mathrm{w}(a,b)+\mathrm{w}(a+1,b+1)$ 则称 w 满足四边形不等式

证明

对于 $a < c$ 有

$$\mathrm{w}(a,c+1)+\mathrm{w}(a+1,c) \geq \mathrm{w}(a,c)+\mathrm{w}(a+1,c+1)$$

对于 $a+1 < c$ 有

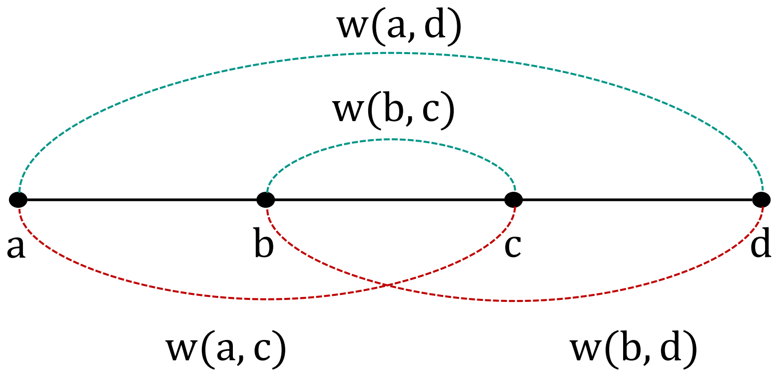
$$\mathrm{w}(a+1,c+1)+\mathrm{w}(a+2,c) \geq \mathrm{w}(a+1,c)+\mathrm{w}(a+2,c+1)$$

两式相加整理有

$$\mathrm{w}(a,c+1)+\mathrm{w}(a+2,c) \geq \mathrm{w}(a+1,c)+\mathrm{w}(a+2,c+1)$$

依此类推，$\forall\, a \leq b \leq c$ 有

$$\mathrm{w}(a,c+1)+\mathrm{w}(b,c) \geq \mathrm{w}(a,c)+\mathrm{w}(b,c+1)$$



同理可推出：$\forall\, a \leq b \leq c \leq d$ 有 $\mathrm{w}(a,c)+\mathrm{w}(b,d) \leq \mathrm{w}(a,d)+\mathrm{w}(b,c)$

四边形不等式优化

称 $\mathrm{dp}[l][r]$ 取到最值的点中间点为最优决策点，记其为 $\mathbf{opt}(l,r)$，特殊的 $\mathbf{opt}(l,l)=l$

若 $\mathrm{w}(l,r)$ 满足 **区间包含单调性** 和 **四边形不等式**，则状态 $\mathrm{dp}[l][r]$ 满足 **四边形不等式**

即

$$\mathrm{dp}[l][r+1]+\mathrm{dp}[l+1][r]\geq \mathrm{dp}[l][r]+\mathrm{dp}[l+1][r+1]$$

证明

通过数学归纳法证明（对区间长度进行归纳）

- 当 $l+1=r$ 时

$$\mathrm{dp}[l][r+1]+\mathrm{dp}[l+1][r]=\mathrm{dp}[l][l+2]+\mathrm{dp}[l+1][l+1]$$

$$=\mathrm{dp}[l][l+2]$$

  - 若 $\mathbf{opt}(l,l+2)=l$

$$\mathrm{dp}[l][l+2]=\mathrm{dp}[l][l]+\mathrm{dp}[l+1][l+2]+\mathrm{w}(l,l+2)=0+\mathrm{w}(l+1,l+2)+\mathrm{w}(l,l+2)$$

根据 w 的区间包含单调性有

四边形不等式优化

$$\begin{aligned} \mathrm{w}(l+1,l+2)+\mathrm{w}(l,l+2) &\geq \mathrm{w}(l+1,l+2)+\mathrm{w}(l,l+1) \\ &= \mathrm{dp}[l+1][l+2]+\mathrm{dp}[l][l+1] \\ &= \mathrm{dp}[l+1][r+1]+\mathrm{dp}[l][r] \end{aligned}$$

当 $\mathrm{dp}[l][l+2]$ 最优决策点为 l 时状态满足四边形不等式

- 若 $\mathbf{opt}(l,l+2)=l+1$

$$\mathrm{dp}[l][l+2]=\mathrm{dp}[l][l+1]+\mathrm{dp}[l+2][l+2]+\mathrm{w}(l,l+2)=\mathrm{w}(l,l+1)+0+\mathrm{w}(l,l+2)$$

根据 w 的区间包含单调性有

$$\begin{aligned} \mathrm{w}(l,l+1)+\mathrm{w}(l,l+2) &\geq \mathrm{w}(l,l+1)+\mathrm{w}(l+1,l+2) \\ &= \mathrm{dp}[l][l+1]+\mathrm{dp}[l+1][l+2] \\ &= \mathrm{dp}[l][r]+\mathrm{dp}[l+1][r+1] \end{aligned}$$

当 $\mathrm{dp}[l][l+2]$ 最优决策点为 $l+1$ 时状态满足四边形不等式

四边形不等式优化

综上当 $l+1=r$ 时状态满足四边形不等式

- 考虑更一般情况

设当 $r-l+1\le \text{len}$ 时状态满足四边形不等式

要证 $r-l+1=\text{len}+1$ 时

$$\text{dp}[l][r+1]+\text{dp}[l+1][r]\ge \text{dp}[l][r]+\text{dp}[l+1][r+1]$$

记 $\textbf{opt}(l,r+1)=x$ ，$\textbf{opt}(l+1,r)=r$ 不妨设 $l\le x<y$

要求 $x<y$ 是为了保证 $\text{dp}[l+1][y]\ne\infty$, 当 $l+1\le x\le y$ 时证明类似

根据最优性

$$\text{dp}[l][r+1]+\text{dp}[l+1][r]=$$

$$\text{dp}[l][x]+\text{dp}[x+1][r+1]+\text{w}(l,r+1)+$$

$$\text{dp}[l+1][y]+\text{dp}[y+1][r]+\text{w}(l+1,r)$$

对于 $\text{dp}[l][r],\text{dp}[l+1][r+1]$ 而言 x,y 不一定为最优决策点

四边形不等式优化

$\mathrm{dp}[l][r]$ 和 $\mathrm{dp}[l+1][r+1]$ 分别以 x,y 为决策点时不优于最优解

即

$$\mathrm{dp}[l][r]+\mathrm{dp}[l+1][r+1]\le$$

$$\mathrm{dp}[l][x]+\mathrm{dp}[x+1][r]+\mathrm{w}(l,r)+$$

$$\mathrm{dp}[l+1][y]+\mathrm{dp}[y+1][r+1]+\mathrm{w}(l+1,r+1)$$

转证

$$\mathrm{dp}[l][x]+\mathrm{dp}[x+1][r+1]+\mathrm{w}(l,r+1)+\mathrm{dp}[l+1][y]+\mathrm{dp}[y+1][r]+\mathrm{w}(l+1,r)\ge$$

$$\mathrm{dp}[l][x]+\mathrm{dp}[x+1][r]+\mathrm{dp}[l+1][y]+\mathrm{dp}[y+1][r+1]+\mathrm{w}(l+1,r+1)$$

将其整理

$$\mathrm{dp}[x+1][r+1]+\mathrm{dp}[y+1][r]+\mathrm{w}(l,r+1)+\mathrm{w}(l+1,r)\ge$$

$$\mathrm{dp}[x+1][r]+\mathrm{dp}[y+1][r+1]+\mathrm{w}(l,r)+\mathrm{w}(l+1,r+1)\qquad(1)$$

四边形不等式优化

由于 $l \le x$ 且 $r-l+1=\text{len}+1$ 那么

$$r-(x+1)+1<r-l+1=\text{len}+1 \Rightarrow r-(x+1)+1 \le \text{len}$$

根据归纳假设

$$\text{dp}[x+1][r+1]+\text{dp}[y+1][r] \ge \text{dp}[x+1][r]+\text{dp}[y+1][r+1] \tag{2}$$

又因为 w 满足四边形不等式

$$\text{w}(l,r+1)+\text{w}(l+1,r) \ge \text{w}(l,r)+\text{w}(l+1,r+1) \tag{3}$$

(2), (3) 式相加即为 (1) 式

即

$$\text{dp}[l][r+1]+\text{dp}[l+1][r] \ge \text{dp}[l][r]+\text{dp}[l+1][r+1]$$

对于 $l \le y < x$ 同理可证

综上状态满足四边形不等式

若将状态转移中的 min 换为 max 证明过程类似

四边形不等式优化

决策单调性

若状态 dp 满足四边形不等式，$\forall\, l < r$ 有 $\mathbf{opt}(l,r-1) \le \mathbf{opt}(l,r) \le \mathbf{opt}(l+1,r)$

证明

记 $p = \mathbf{opt}(l,r)$

设 $k \le p$，将 $\mathrm{dp}[l+1][k]$ 以 k,p 两种决策拆分

$$\big(\mathrm{dp}[l+1][k] + \mathrm{dp}[k+1][r] + w(l+1,r)\big) - \big(\mathrm{dp}[l+1][p] + \mathrm{dp}[p+1][r] + \mathrm{w}(l+1,r)\big)$$

$$= (\mathrm{dp}[l+1][k] - \mathrm{dp}[l+1][p]) + (\mathrm{dp}[k+1][r] - \mathrm{dp}[p+1][r])$$

根据四边形不等式

$$\mathrm{dp}[l][p] + \mathrm{dp}[l+1][k] \ge \mathrm{dp}[l][k] + \mathrm{dp}[l+1][p] \Rightarrow (\mathrm{dp}[l+1][k] - \mathrm{dp}[l+1][p]) \ge (\mathrm{dp}[l][k] - dp[l][p])$$

那么

$$(\mathrm{dp}[l+1][k] - \mathrm{dp}[l+1][p]) + (\mathrm{dp}[k+1][r] - \mathrm{dp}[p+1][r])$$

$$\ge (\mathrm{dp}[l][k] - dp[l][p]) + (\mathrm{dp}[k+1][r] - \mathrm{dp}[p+1][r])$$

四边形不等式优化

$$= (\mathrm{dp}[l][k] + \mathrm{dp}[k+1][r]) - (\mathrm{dp}[l][p] + \mathrm{dp}[p+1][r])$$

根据 p 的最优性有

$$\mathrm{dp}[l][k] + \mathrm{dp}[k+1][r] \geq \mathrm{dp}[l][p] + \mathrm{dp}[p+1][r]$$

$$\Rightarrow (\mathrm{dp}[l][k] + \mathrm{dp}[k+1][r]) - (\mathrm{dp}[l][p] + \mathrm{dp}[p+1][r]) \geq 0$$

那么

$$\left(\mathrm{dp}[l+1][k] + \mathrm{dp}[k+1][r] + w(l+1,r)\right) - \left(\mathrm{dp}[l+1][p] + \mathrm{dp}[p+1][r] + \mathrm{w}(l+1,r)\right) \geq 0$$

这意味着对 $\mathrm{dp}[l+1][r]$ 而言，选择 $k \leq p$ 作为决策点不优于选择 $\mathrm{dp}[l][r]$ 的最优决策点 p

那么 $\mathrm{dp}[l+1][r]$ 的最优决策点仅有可能 $\geq p$，即 $\mathbf{opt}(l+1,r) \geq \mathbf{opt}(l,r)$

同理可证 $\mathbf{opt}(l,r-1) \leq \mathbf{opt}(l,r)$

给出另一形式的证明 $\mathbf{opt}(l,r-1) \leq \mathbf{opt}(l,r)$

记 $k = \mathbf{opt}(l-1,r)$，$p = \mathbf{opt}(l+1,r)$，考虑反证法

若有 $p < k < r-1 \Rightarrow p+1 < k+1 \leq r-1 < r$，根据四边形不等式有

四边形不等式优化

$$\mathrm{dp}[p+1][r]+\mathrm{dp}[k+1][r-1]\geq \mathrm{dp}[p+1][r-1]+\mathrm{dp}[k+1][r]$$

由于 u 为 $\mathrm{dp}[l][r]$ 最优决策点，那么

$$\mathrm{dp}[l][k]+\mathrm{dp}[k+1][r]\geq \mathrm{dp}[l][p]+\mathrm{dp}[p+1][r]$$

两式相加有

$$\mathrm{dp}[p+1][r]+\mathrm{dp}[k+1][r-1]+\mathrm{dp}[l][k]+\mathrm{dp}[k+1][r]\geq \mathrm{dp}[p+1][r-1]+\mathrm{dp}[k+1][r]+\mathrm{dp}[l][p]+\mathrm{dp}[p+1][r]$$

$$\Rightarrow \mathrm{dp}[l][k]+\mathrm{dp}[k+1][r-1]\geq \mathrm{dp}[l][p]+\mathrm{dp}[p+1][r-1]$$

这与 k 为 $\mathrm{dp}[l][r-1]$ 最优决策点矛盾，即 $\mathbf{opt}(l,r-1)\leq \mathbf{opt}(l,r)$

~~也可通过打表验证决策单调性~~

四边形不等式 (quadrangle inequality) 用于 DP 优化源自 Donald E. Knuth 于1971 年的一篇论文，用于解决最优二叉搜索树问题

1980 年储枫 (F. Frances Yao 姚期智夫人) 做了深入研究扩展为一般性的 DP 优化方法

该方法又被称为 Knuth – Yao DP Speedup Theorem

#2461、又是合并石子

题目描述

在一个操场上一排地摆放着 N 堆石子

现要将石子有次序地合并成一堆

规定每次只能选相邻的 2 堆石子合并成新的一堆，并将新的一堆石子数记为该次合并的得分

计算出将 N 堆石子合并成一堆的最小得分

输入格式

第一行为一个正整数 N

第二行 N 个正整数,分别表示第 i 堆石子的个数

输出格式

一个正整数,即最小得分

数据规模

对于全部的数据 $2 \le N \le 5000$,石子个数不超过 1000

本题 $\mathrm{w}(l,r) = \mathrm{sum}_r - \mathrm{sum}_{l-1}$

$\forall\, l_1 \le l_2 \le r_1 \le r_2$ 有

$$\mathrm{w}(l_1,r_2) + \mathrm{w}(l_2,r_1) = \mathrm{w}(l_1,r_1) + \mathrm{w}(l_2,r_2)$$

那么状态满足四边形不等式，决策满足单调性

限制中间点 k 的枚举范围 $\mathbf{opt}(l,r-1) \sim \mathbf{opt}(l+1,r)$

未限制前时间复杂度为

$$O\left(\sum_{l=1}^{n-1}\sum_{r=1+1}^{n}(r-l+1)\right) \le O(n^3)$$

在限制后递推时区间长度 $\delta = 1,2,3,\cdots,n$ 逐步递增

对于一个固定的 δ 有 $1 \le l \le n-\delta+1$, $r = l+\delta-1$

$\mathrm{dp}[l][r]$ 转移决策点枚举范围为 $\mathbf{opt}[l][r-1] \sim \mathbf{opt}[l+1][r]$

#2461、又是合并石子

当 $l=1$ 时，枚举次数为

$$\mathbf{opt}[2][\delta]-\mathbf{opt}[1][\delta-1]+1$$

当 $l=2$ 时，枚举次数为

$$\mathbf{opt}[3][\delta+1]-\mathbf{opt}[2][\delta]+1$$

当 $l=3$ 时，枚举次数为

$$\mathbf{opt}[4][\delta+2]-\mathbf{opt}[3][\delta+1]+1$$

$$\vdots$$

当 $l=n-\delta$ 时，枚举次数为

$$\mathbf{opt}[n-\delta+1][n-1]-\mathbf{opt}[n-\delta][n-2]+1$$

当 $l=n-\delta+1$ 时，枚举次数为

$$\mathbf{opt}[n-\delta+2][n]-\mathbf{opt}[n-\delta+1][n-1]+1$$

#2461、又是合并石子

将上式累加，总枚举次数为

$$\mathbf{opt}[n-\delta+2][n]-\mathbf{opt}[1][\delta-1]+n-\delta+1$$

$\mathbf{opt}[n-\delta+2][n] \le n$ 且 $1 \le \mathbf{opt}[1][\delta-1]$ 那么

$$\mathbf{opt}[n-\delta+2][n]-\mathbf{opt}[1][\delta-1]+n-\delta+1$$

$$\le n-1+n-\delta+1=2n-\delta$$

即 δ 固定时枚举次数不超过 $2n$

那么

$$O\left(\sum_{l=1}^{n-1}\sum_{r=1+1}^{n}(\mathbf{opt}[l+1][r]-\mathbf{opt}[l][r-1]+1)\right) \le O\left(\sum_{\delta=1}^{n}(2n-\delta\,)\right) \sim O(n^2)$$

Garcia Wachs *算法可以* $O(n\log n)$ *代价解决本题*

#3993、 绝世好题

题目描述

给定一个长度为 n 的数列 A

求 A 的子序列 B 的最长长度 k

要求 $\forall 2 \le \mathrm{i} \le \mathrm{K}$ 都满足 $\mathrm{B_i}$ **and** $\mathrm{B_{i-1}} \neq 0$

> **and** *表示按位与*

输入格式

第一行输入一个整数 n

第二行输入 n 个整数，第 i 个整数表示 $\mathrm{A_i}$

输出格式

输出一个整数表示 K

数据规模

对于 100% 的数据 $1 \le \mathrm{n} \le 100000, 1 \le \mathrm{a_i} \le 10^9$

类比 LIS 不难想出 $O(n^2)$ 的做法

子序列中相邻两数同一二进制位为 1 即可满足条件

重新设计状态

设 dp[x] 为第 x 个二进制位为 1 时的最大长度

对于 A_i 将其转为二进制考虑

若 A_i 第 x 个二进制位为 1，选取 A_i 时最大长度为

$$\max(\mathrm{dp}[x] + 1)$$

记 p 为选取 A_i 时的最大长度

需更新 A_i 二进制为 1 对应位处的 dp 值

$$\mathrm{dp}[x] = \max(\mathrm{dp}[\mathrm{x}], p)$$

答案为 $\max\limits_{0 \le x \le 29}(\mathrm{dp}[x])$，时间复杂度 $O(30n)$

#2147、LCIS

题目描述

对于两个数列 A 和 B

若其都包含一段位置不一定连续的数且数值是严格递增的,那么称这一段数是两个数列的公共上升子序列

而所有的公共上升子序列中最长的就是最长公共上升子序列了

请你求出最长公共上升子序列长度

输入格式

第一行包含一个整数 N ,表示数列 A, B 的长度

第二行包含 N 个整数,表示数列 A

第三行包含 N 个整数,表示数列 B

输出格式

输出一个整数,表示最长公共上升子序列的长度

数据范围

对于 40% 的数据 $1 \le N \le 200$

对于 100% 的数据 $1 \le N \le 10000$ ，序列中的数字均不超过 $2^{31} - 1$

设 $dp[i][j]$ 为考虑 $A_{1\sim i}$, $B_{1\sim j}$ 且 **以 B_j 结尾** 的 LCIS 长度

若与 LCS 问题一样不限定结尾，那么 B_j 不便转移

当 $A_i \neq B_j$ 时仅需考虑 $A_{1\sim i-1}, B_{1\sim j}$

$$dp[i][j] = dp[i-1][j]$$

否则需找到 B_k $(1 \le k < i)$ 构成上升

$$dp[i][j] = \max_{1 \le k < j \ \wedge \ B_k < B_j} \{dp[i-1][k] + 1\}$$

时间复杂度 $O(|A|\ |B|^2)$

上述做法需以 $O(|B|)$ 的代价找出

$$\max_{1 \le k < j \ \wedge \ B_k < B_j} \{dp[i-1][k] + 1\}$$

#2147、LCIS

考虑优化

记 $\mathrm{S}(i,j)$ 为满足 $k<j$ 且 $\mathrm{B}_k<\mathrm{A}_i$ 的 $\mathrm{dp}[i-1][k]$ 构成的可选决策集合

每一轮枚举 i 时 A_i 已经确定

已加入的决策点不再移除，仅需考察 $\mathrm{B}_j<\mathrm{A}_i$ 决定是否加入 $\mathrm{dp}[i-1][j]$ 即可

$$\mathrm{S}_{i,j+1}=\begin{cases}\mathrm{S}_{i,j}, & \mathrm{B}_j\geq\mathrm{A}_i\\ \mathrm{S}_{i,j}\cup\mathrm{dp}[i-1][j], & \mathrm{B}_j<\mathrm{A}_i\end{cases}$$

具体实现时可仅维护 $\max\{\mathrm{S}_{i,*}\}$

转移时使用 $\max\{\mathrm{S}_{i,j}\}+1$ 更新 $\mathrm{dp}[i][j]$ 即可

$\mathrm{dp}[i][1\ \sim\ n]$ 仅与 $\mathrm{dp}[i-1][1\ \sim\ n]$ 有关可使用滚动数组优化

时/空间复杂度$O(\ |\mathrm{A}|\ |\mathrm{B}|\ )$

#3944、 通配符

题目描述

给出字符串 S 和模式串 P

其中 S 仅由小写字母组成，P 由小写字母和 $?*$ 组成

其中

- $?$ 可以匹配任何单个字符
- $*$ 可以匹配任意字符序列 (包括空字符序列)

若 P 能完全匹配 S 则输出 Yes 否则输出 No

判定匹配成功的充要条件是：字符模式必须能够 完全匹配 输入字符串（而不是部分匹配）.

输入格式

第一行字符串 S

第二行输入字符串 P

输出格式

若能匹配则输出 Yes 否则输出 No

数据规模

对于全部的数据 $0 \le |S|, |P| \le 5000$

设 $dp[i][j]$ 表示 $S_{1 \dots i}$ 和 $P_{1 \dots j}$ 是否能完全匹配

- 若 P_i 为字母

  仅当 $S_i = P_j$ 且 $S_{1 \dots i-1}$ 和 $P_{1 \dots j-1}$ 完全匹配时为 true

  即此时有

  $$dp[i][j] = dp[i-1][j-1]$$

- 若 $P_j = ?$

  P_j 可作为任意字符与 S_i 匹配

  仅需 $S_{1 \dots i-1}$ 和 $P_{1 \dots j-1}$ 完全匹配时为 true

  即此时有

  $$dp[i][j] = dp[i-1][j-1]$$

- 若 $P_j = *$

#3944、 通配符

- 若不使用 P_j

  仅当 $S_{1\dots i}$与 $P_{1\dots j-1}$ 完全匹配时为 true

  即此时有

$$\mathrm{dp}[i][j] = \mathrm{dp}[i][j-1]$$

- 若使用 P_j

  此时 P_j 可匹配多个字符

  若 P_j 匹配 1 个字符则仅需考察 $S_{1\dots i-1}$ 与 $P_{1\dots j-1}$

  若 P_j 匹配 2 个字符则仅需考察 $S_{1\dots i-2}$ 与 $P_{1\dots j-1}$

$$\vdots$$

  仅需 $\mathrm{dp}[i-1][j-1] \vee \mathrm{dp}[i-2][j-1] \vee \mathrm{dp}[i-3][j-1] \vee \cdots$ 的结果为 true 即可完全匹配

  对于 $\mathrm{dp}[i-1][j]$ 当使用 P_j 时，其已考察 $\mathrm{dp}[i-2][j-1] \vee \mathrm{dp}[i-1][j-3] \vee \cdots$

#3944、 通配符

当不使用 P_j 时，$\mathrm{dp}[i-1][j]$ 已考察 $\mathrm{dp}[i-1][j-1]$

那么

$$\mathrm{dp}[i][j] = \mathrm{dp}[i][j-1] \vee \mathrm{dp}[i-1][j]$$

可将上述情况的状态转移进行整理

$$\mathrm{dp}[i][j] = \begin{cases} \mathrm{dp}[i-1][j-1], & P_j = S_i \vee P_j = '?' \\ \mathrm{dp}[i-1][j] \vee \mathrm{dp}[i][j-1], & P_j = '*' \\ \mathrm{false}, & \mathrm{else} \end{cases}$$

考虑边界

- $\mathrm{dp}[0][0] = \mathrm{true}$
- 当 $P_{1 \dots i}$ 都为 $'*'$ 时 $\mathrm{dp}[0][i] = \mathrm{true}$

时间复杂度 $O(|S||P|)$

#2496、 混乱程度

问题描述

N 位选手排成一队，选手的实力被编号为 $1 \sim \mathrm{N}$ (每位选手的实力两两不同)

本来应该从 1 排到 N ，但他们排队常常出现混乱

有人认为是某些选手过于谦让，然而 Mas 不是这么认为

Mas 经过仔细推敲，做出如下定义:

一个队列的混乱程度正是逆序对数!

序列 $(1,4,3,2)$ 的混乱程度为 3 ，因为它含有三对逆序对 $(4,3),(4,2),(3,2)$

Mas 每天都会在他的手册上写上两个新的数 N,C

而他当天的计算机密码是:

这 N 位选手排成混乱程度为 C 的队列的不同排列方式总数 $\bmod 1000000007$.

输入格式

仅含一行两个数 N,C .

输出格式

Mas 当天的密码

设 $dp[i][j]$ 表示使用数字 1~i 产生了 j 个逆序对的数量

考虑数字 i 被插入的位置(此时 i 为最大数)

- 若插在最后无法产生新的逆序对，即

$$dp[i-1][j]$$

- 若插在 $i-1$ 前可与 $i-1$ 产生 1 个逆序对，即

$$dp[i-1][j-1]$$

数据规模

对于 10% 的数据 $1 \le \mathrm{N},\mathrm{C} \le 5$

对于 50% 的数据 $1 \le \mathrm{N},\mathrm{C} \le 100$

对于全部的数据 $1 \le \mathrm{N} \le 1000, 0 \le \mathrm{C} \le 10000$

#2496、 混乱程度

- 若插在 $i-2, i-1$ 前可与 $i-1, i-2$ 产生 2 个逆序对，即

$$\mathrm{dp}[i-1][j-1]$$

$$\vdots$$

- 若插在 $1 \sim i-1$ 前可与 $1 \sim i-1$ 产生 $i-1$ 个逆序对，即

$$\mathrm{dp}[i-1][j-i+1]$$

综上有

$$\mathrm{dp}[i][j] = \sum_{k=\max(0, j-i+1)}^{j} \mathrm{dp}[i-1][k]$$

朴素转移时间复杂度 $O(NC^2)$

维护 $\mathrm{dp}[i-1][*]$ 前缀和，可实现 $O(1)$ 转移

时间复杂度 $O(NC)$

#2890、砖块染色

题目描述

Mas 正在工作

具体工作任务是这样的

有 N 块砖排成一排染色,每一块砖需要涂上红、蓝、绿、黄这 4 种颜色中的其中 1 种

当这 N 块砖中红色和绿色的块数均为偶数时，染色效果最佳

为了使工作效率更高, Mas 想要知道一共有多少种方案可以使染色效果最佳

你能帮帮他吗?

输入格式

第一行为 T ，代表数据组数

接下来 T 行每行包括一个数字 N ,代表有 N 块砖

输出格式

输出满足条件的方案数，答案模 1000000007

数据规模

对于 5% 的数据 $1 \leq n \leq 100$

对于全部的数据 $1 \leq T \leq 100, 1 \leq n \leq 10^{18}$

设 $dp[i][0]$ 为前 i 个砖块红绿个数都为偶数方案数

设 $dp[i][1]$ 为前 i 个砖块红绿个数只有一种为偶数方案数

设 $dp[i][2]$ 为前 i 个砖块红绿个数都为奇数方案数

其中 $dp[1][0] = 2, dp[1][1] = 2, dp[1][2] = 0$

答案为$dp[n][0]$

- 对于 $dp[i][0]$

  在 $dp[i-1][0]$ 基础上，将第 i 块染成 蓝/黄 色

  在 $dp[i-1][1]$ 基础上，将第 i 块染成奇数对应颜色

$$dp[i][0] = 2 \times dp[i-1][0] + dp[i-1][1]$$

#2890、砖块染色

- 对于 $\mathrm{dp}[i][1]$

  在 $\mathrm{dp}[i-1][0]$ 基础上，将第 i 块染成 红/绿 色

  在 $\mathrm{dp}[i-1][1]$ 基础上，将第 i 块染成 蓝/黄 色

  在 $\mathrm{dp}[i-1][2]$ 基础上，将第 i 块染成 红/绿 色

$$\mathrm{dp}[i][1]=2\times\mathrm{dp}[i-1][0]+2\times\mathrm{dp}[i-1][1]+2\times\mathrm{dp}[i-1][2]$$

- 对于 $\mathrm{dp}[i][2]$

  在 $\mathrm{dp}[i-1][1]$ 基础上，将第 i 块染成奇数对应颜色

  在 $\mathrm{dp}[i-1][2]$ 基础上，将第 i 块染成 蓝/黄 色

$$\mathrm{dp}[i][2]=\mathrm{dp}[i-1][1]+2\times\mathrm{dp}[i-1][2]$$

每组询问朴素转移，时间复杂度 $O(n)$

考虑优化

#2890、砖块染色

构造 3×1 的矩阵 $F_i=\begin{bmatrix}dp[i-1][0]\\dp[i-1][1]\\dp[i-1][2]\end{bmatrix}$

考虑将其变为 $\begin{bmatrix}dp[i][0]\\dp[i][1]\\dp[i][2]\end{bmatrix}$

构造一个 3×3 的矩阵 $A=\begin{bmatrix}2&1&0\\2&2&2\\0&1&2\end{bmatrix}$ 那么有 $F_i=AF_{i-1}=A^2F_{i-2}=\cdots$

即 $F_n=A^{n-1}F_1$ 显然 $F_1=\begin{bmatrix}2\\2\\0\end{bmatrix}$

使用矩阵快速幂加速递推即可

时间复杂度 $O(3^3\times\log n)$

实验舱
青少年编程
走近科学 走进名校

谢谢观看