国外选拔赛题选讲与IOI2018 Day2讲解

YMDragon

IIIS

2019 年 1 月 27 日

doll 机械娃娃

构建一个由1 个起点，M 个触发器和S 个开关的网络。开关的数量S 由你决定。

起点和触发器都是是只有一条出边的节点。开关有X 和Y 两条出边，奇数次经过时选择X 这条出边，偶数次经过时选择Y 这条出边。

现在要求构建的网络满足如下条件：

- 从起点出发，经过若干步后回到起点。
- 首次回到起点时，所有的开关都是经过偶数次。
- 首次回到起点时，经过触发器的次数恰好为N 次，且触发器编号顺序依次为$A_0, A_1, \ldots, A_{N-1}$ 。
- 首次回到起点之前，经过开关的次数不能超过$2 * 10^7$ 次。

$1 \leq M \leq 100000, 1 \leq N \leq 200000.$

doll 机械娃娃

子任务1（2分）：对于每个i，整数i 在序列$A_0, A_1, \ldots, A_{N-1}$ 中最多出现1 次。

子任务2（4分）：对于每个i，整数i 在序列$A_0, A_1, \ldots, A_{N-1}$ 中最多出现2 次。

子任务3（10分）：对于每个i，整数i 在序列$A_0, A_1, \ldots, A_{N-1}$ 中最多出现4 次。

子任务4（10分）：$N = 16$。

子任务5（18分）：$M = 1$。

子任务6（56分）：无附加限制。

doll 机械娃娃

得分根据S 的值来计算。

$$Score = \begin{cases} 1 & S \leq N + \log_2 N \\ 0.5 + 0.4 * (\frac{2N-S}{N-log_2 N})^2 & N + \log_2 N < S \leq 2N \\ 0 & S > 2N \end{cases}$$

子任务1

因为每个整数只出现了一次。

子任务1

因为每个整数只出现了一次。

那么可以是直接将触发器按顺序接起来。

子任务2

每个整数只出现了一次或两次。

子任务2

每个整数只出现了一次或两次。

对于出现一次的触发器，直接令其出边指向下一个数的触发器。

子任务2

每个整数只出现了一次或两次。

对于出现一次的触发器，直接令其出边指向下一个数的触发器。

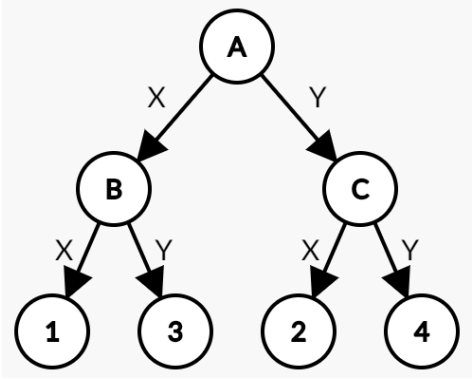
对于出现两次的触发器，令其出边指向一个开关，并且开关的X出边指向第一次出现时的后一个数，Y出边指向第二次出现时的后一个数。

子任务3

对于出现四次的触发器，我们可以构造一个四叶节点的结构。

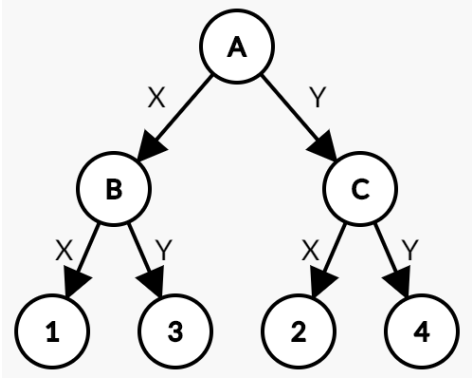
子任务3

对于出现四次的触发器，我们可以构造一个四叶节点的结构。



子任务3

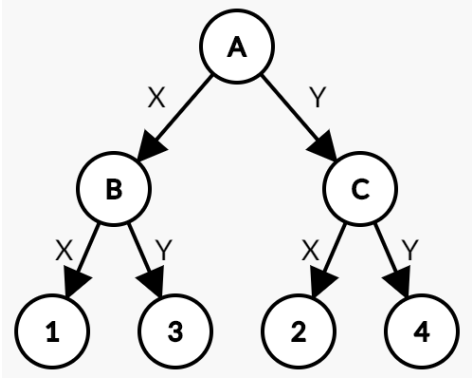
对于出现四次的触发器，我们可以构造一个四叶节点的结构。



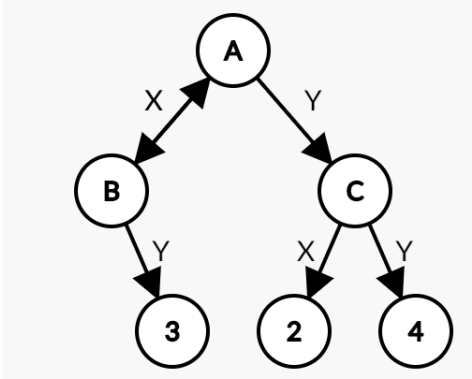
而对于出现三次的触发器，可以令前三个的某个出边变成“root”。

子任务3

对于出现四次的触发器，我们可以构造一个四叶节点的结构。



而对于出现三次的触发器，可以令前三个的某个出边变成“root”。



子任务4

可以发现根本没有必要对于每个触发器，建立一颗子树。

子任务4

可以发现根本没有必要对于每个触发器，建立一颗子树。

那么我们将其合并成一颗，对于$n = 16$的情况只需要一颗5层的满二叉树即可（15个开关）。

子任务5

我们直接建立一颗足够大的满二叉树，然后把部分叶子节点变成“root”。

子任务5

我们直接建立一颗足够大的满二叉树，然后把部分叶子节点变成“root”。这个可以获得一部分的分。

子任务5

我们直接建立一颗足够大的满二叉树，然后把部分叶子节点变成“root”。这个可以获得一部分的分。

考虑优化，我们可以是有技巧的选择变成“root”的叶子节点。那么对于子树中全是这种节点的边可以直接指向“root”。

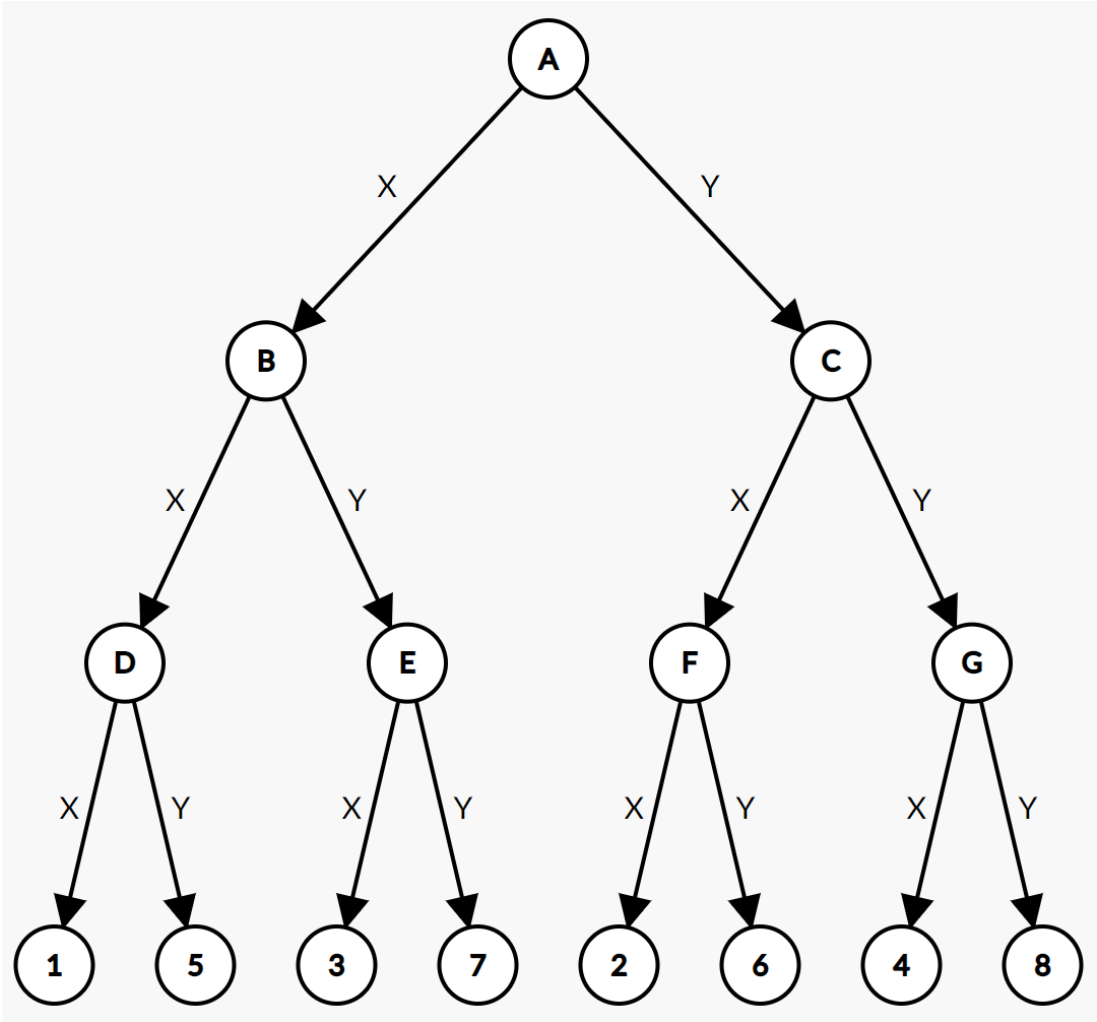
子任务5

我们直接建立一颗足够大的满二叉树，然后把部分叶子节点变成“root”。这个可以获得一部分的分。

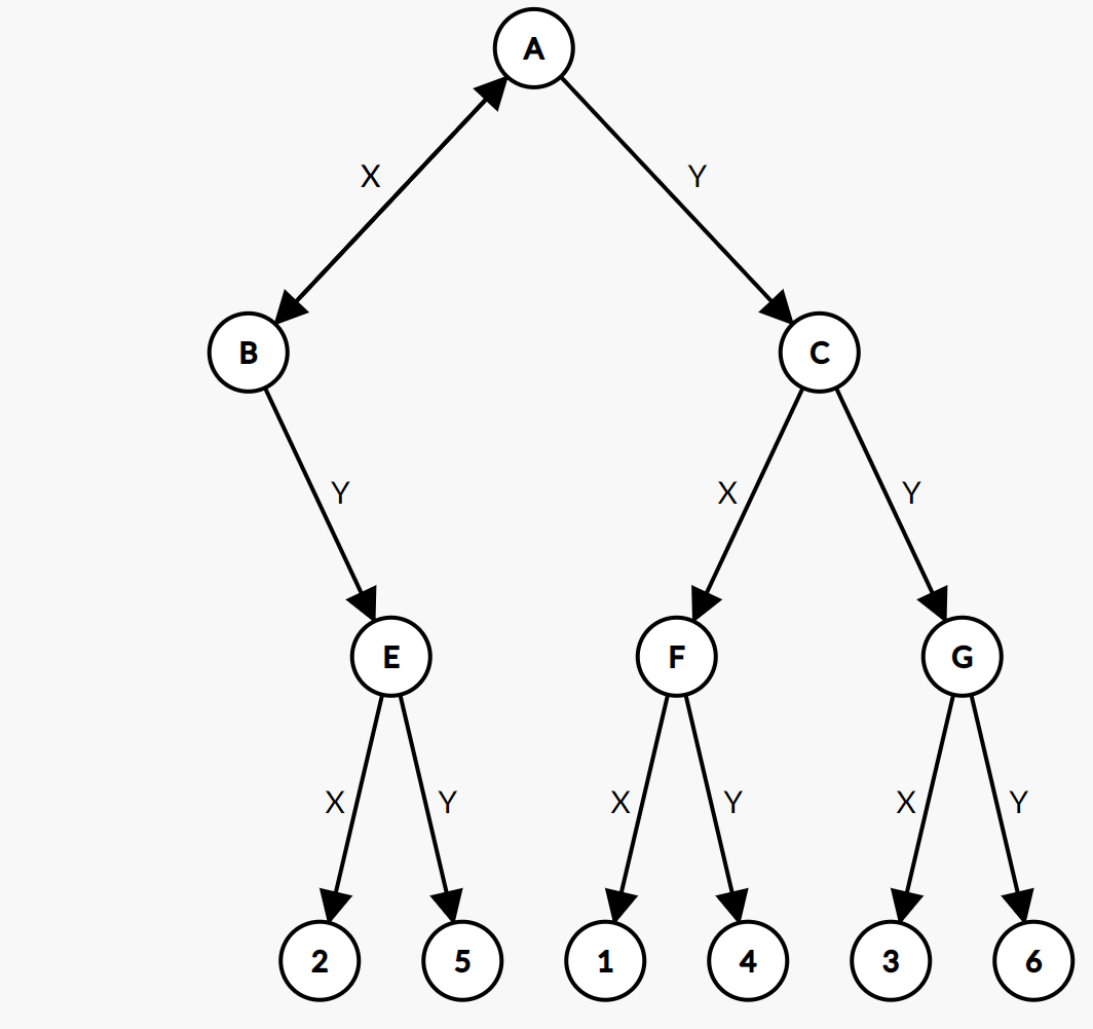
考虑优化，我们可以是有技巧的选择变成“root”的叶子节点。那么对于子树中全是这种节点的边可以直接指向“root”。

一种处理方法，也是最简单的方法是，直接令所有这种叶子节点是最靠左的节点。

子任务5



子任务5



子任务6

我们只需要结合子任务4和子任务5的思路就可以了。

highway 高速公路收费

一张N 个点M 条边的无向连通图，每条边的权值为A 或B 中的一个。

你能做的操作是给每条边设定权值后能够知道S 到T 的最短路长度。你知道A, B 的值，但不知道S, T 。

你的任务是进行尽量少的操作次数确定S, B 的值。

$2 \leq N \leq 90000, 1 \leq M \leq 130000, 1 \leq A < B \leq 1000000000$

highway 高速公路收费

子任务1（5分）：S 或T 中有一个是0 ，$N \leq 100, M = N - 1$。

子任务2（7分）：S 或T 中有一个是0 ，$M = N - 1$ 。操作次数不超过60 。

子任务3（6分）：$M = N - 1$ ，图为一条链。操作次数不超过60 。

子任务4（33分）：$M = N - 1$ 。操作次数不超过60 。

子任务5（18分）：$A = 1, B = 2$ 。操作次数不超过52 。

子任务6（31分）：没有附加限制。操作次数不超过52 可得21分，操作次数不超过50 可得31 分。

子任务1

对于每一个点，都测试一下是否为T。

子任务2

以S为起点建立bfs序，将所有点按照到bfs序排序，使用二分来判断。

子任务2

以S为起点建立bfs序，将所有点按照到bfs序排序，使用二分来判断。

具体的方法就是将小于二分位置的点与父亲的边的权值改为B，那么如果最短路长度为$B*$最短路边数，那么说明t是在小于二分位置的一侧，反之则反。

子任务3

同样还是二分，一次二分便可以知道一个点的位置。

子任务4

我们从任意一点为起点建立bfs序，那么一次二分就可以找到S, T中bfs序较大的那个点。变为子任务2的情况，再进行一次二分即可。

子任务4

我们从任意一点为起点建立bfs序，那么一次二分就可以找到S, T中bfs序较大的那个点。变为子任务2的情况，再进行一次二分即可。

或者是，先利用二分找到最短路上的一条边，然后将这条边删去，变成两个子任务2的情况。

子任务5

对于V 的一个子集V'，如果我们将V'与$V \setminus V'$之间的边设为1，其他边设为2。那么，我们可以通过最短路的奇偶性来判定S, T是否在同一个集合内。

子任务5

对于V 的一个子集V'，如果我们将V'与$V \setminus V'$之间的边设为1，其他边设为2。那么，我们可以通过最短路的奇偶性来判定S, T是否在同一个集合内。

那么这样我们便可以知道S *xor* T，然后便可以知道S, T。

子任务6 - 21分

我们以任意一个点为起点，建立bfs树并进行二分，判定条件改为最短路是否发生改变，这样便可以利用一次二分找到最短路上的一个点v。

子任务6 - 21分

我们以任意一个点为起点，建立bfs树并进行二分，判定条件改为最短路是否发生改变，这样便可以利用一次二分找到最短路上的一个点v。

然后以v为起点，建立bfs树，若把所有的非树边定为B，那么最短路一定是全选树边。

子任务6 - 21分

我们以任意一个点为起点，建立bfs树并进行二分，判定条件改为最短路是否发生改变，这样便可以利用一次二分找到最短路上的一个点v。

然后以v为起点，建立bfs树，若把所有的非树边定为B，那么最短路一定是全选树边。

问题转化为子任务4的情况，但只能使用第一种方法。

子任务6

我们可以是先用二分找到一条在最短路上的边(u, v)。

子任务6

我们可以是先用二分找到一条在最短路上的边(u, v)。

我们不妨假设路径是$S \to u \to v \to T$，那么S一定在到u距离小于到v距离的集合中，T则反之。

子任务6

我们可以是先用二分找到一条在最短路上的边(u, v)。

我们不妨假设路径是$S \to u \to v \to T$，那么S一定在到u距离小于到v距离的集合中，T则反之。

那么对于每一个集合，我们以u或v建立bfs树，将非树边定为B之后，问题就转化子任务2的情况了。

meeting 会议

N 座山排成一行，第i 座山的高度是H_i ，每座山上恰有一个人。

有Q 个会议，每个会议需要第L_j 座山到第R_j 座山之间的人参加。对于一个会议需要选择一座山$x(L_j \leq x \leq R_j)$ 来举办。会议的成本是所有参会者所在山与x之间山的高度最大值之和。要求最小化成本。

每次会议后参会者会回到自己的山，即询问之间独立。

$1 \leq N \leq 750000, 1 \leq Q \leq 750000$

子任务1（4分）：$N \leq 3000, Q \leq 10$ 。

子任务2（15分）：$N \leq 5000, Q \leq 5000$ 。

子任务3（17分）：$N \leq 100000, Q \leq 100000, H_i \leq 2$ 。

子任务4（24分）：$N \leq 100000, Q \leq 100000, H_i \leq 20$ 。

子任务5（40分）：没有附加限制。

子任务1

直接暴力枚举x的取值并计算成本。

复杂度$O(N^2Q)$。

子任务2

可以发现计算成本的过程可以使用单调队列来优化。

复杂度$O(NQ)$。

子任务3

如果高度只有1和2，那么显然x一定是在最长的一段1中，线段树维护即可。

复杂度$O(N + Q \log N)$。

子任务4

高度≤ 20，我们可以是预处理出两侧高度相同且中间均小于两侧的这样的区间的信息（x的位置以及成本）。

子任务4

高度≤ 20，我们可以是预处理出两侧高度相同且中间均小于两侧的这样的区间的信息（x的位置以及成本）。

有了这个之后，那么每次的询问就是$O(H \log n)$或$O(H)$的了。

子任务4

高度≤ 20，我们可以是预处理出两侧高度相同且中间均小于两侧的这样的区间的信息（x的位置以及成本）。

有了这个之后，那么每次的询问就是$O(H \log n)$或$O(H)$的了。

复杂度$O(H(N + Q \log N))$或$O(H(N + Q))$。

子任务5

为了方便讨论，我们不妨假设所有高度均不相同。

子任务5

为了方便讨论，我们不妨假设所有高度均不相同。

会议点x的情况有三种：

子任务5

为了方便讨论，我们不妨假设所有高度均不相同。

会议点x的情况有三种：

- x就在区间最大值处，直接计算就好。

子任务5

为了方便讨论，我们不妨假设所有高度均不相同。

会议点x的情况有三种：

- x就在区间最大值处，直接计算就好。
- x在区间最大值左边。
- x在区间最大值右边。

显然后两种情况是等价的，我们下面只考虑x在区间最大值右边的情况。

子任务5

定义$Cost(l, r)$为区间$[l, r]$的最小成本。

子任务5

定义$Cost(l, r)$为区间$[l, r]$的最小成本。

定义$L(x)$为x左边第一个大于它的右边的位置，即满足$argmax_{y \le i \le x}(H_i) = x$的最小的$y$。

同样定义$R(x)$为x右边第一个大于它的左边的位置，即满足$argmax_{x \le i \le y}(H_i) = x$的最大的$y$。

子任务5

定义$Cost(l, r)$为区间$[l, r]$的最小成本。

定义$L(x)$为x左边第一个大于它的右边的位置，即满足$argmax_{y \le i \le x}(H_i) = x$的最小的$y$。

同样定义$R(x)$为x右边第一个大于它的左边的位置，即满足$argmax_{x \le i \le y}(H_i) = x$的最大的$y$。

定义$S(x)$为一个长为$R(x) - L(x) + 1$的数组，其中第i个元素的值为$Cost(L(x), L(x) + i - 1)$。

子任务5

定义$Cost(l,r)$为区间$[l,r]$的最小成本。

定义$L(x)$为x左边第一个大于它的右边的位置，即满足$argmax_{y\le i\le x}(H_i)=x$的最小的$y$。

同样定义$R(x)$为x右边第一个大于它的左边的位置，即满足$argmax_{x\le i\le y}(H_i)=x$的最大的$y$。

定义$S(x)$为一个长为$R(x)-L(x)+1$的数组，其中第i个元素的值为$Cost(L(x),L(x)+i-1)$。

那么有了S之后就可以找到区间最大值右边部分的最大值，然后利用它的S来进行回答了。

子任务5

我们对于H_i构建笛卡尔树，那么考虑如何用$S(Lson(x))$和$S(Rson(x))$来计算$S(x)$。

子任务5

我们对于H_i构建笛卡尔树，那么考虑如何用$S(Lson(x))$和$S(Rson(x))$来计算$S(x)$。

若$i < x$，$Cost(L(x), i) = Cost(L(Lson(x)), i)$。

子任务5

我们对于H_i构建笛卡尔树，那么考虑如何用$S(Lson(x))$和$S(Rson(x))$来计算$S(x)$。

若$i < x$，$Cost(L(x), i) = Cost(L(Lson(x)), i)$。

若$i = x$，$Cost(L(x), i) = Cost(L(x), x-1) + H_x$。

子任务5

我们对于H_i构建笛卡尔树，那么考虑如何用$S(Lson(x))$和$S(Rson(x))$来计算$S(x)$。

若$i < x$，$Cost(L(x), i) = Cost(L(Lson(x)), i)$。

若$i = x$，$Cost(L(x), i) = Cost(L(x), x-1) + H_x$。

若$i > x$，$Cost(L(x), i) =$

$\min(Cost(L(x), x-1)+(i-x+1)*H_x, Cost(x+1, i)+(x-L(x)+1)*H_x)$。

子任务5

可以发现，

$$(Cost(L(x),x-1))+(i-x+1)*H_x-(Cost(x+1,i)+(x-L(x)+1)*H_x)\leq$$

$$(Cost(L(x),x-1))+((i+1)-x+1)*H_x-(Cost(x+1,i+1)+(x-L(x)+1)*H_x)$$

子任务5

可以发现，

$$(Cost(L(x),x-1))+(i-x+1)*H_x-(Cost(x+1,i)+(x-L(x)+1)*H_x)\leq$$

$$(Cost(L(x),x-1))+((i+1)-x+1)*H_x-(Cost(x+1,i+1)+(x-L(x)+1)*H_x)$$

这是因为$Cost(x+1,i+1)-Cost(x+1,i)\leq \max_{x+1\leq j\leq i+1}(H_j)\leq H_x$。

子任务5

所以对于$i > x$的部分，存在一个p使得：

子任务5

所以对于$i > x$的部分，存在一个p使得：

当$i \leq p$时，$Cost(L(x), i) = Cost(L(x), x-1) + (i - x + 1) * H_x$。

子任务5

所以对于$i > x$的部分，存在一个p使得：

当$i \leq p$时，$Cost(L(x), i) = Cost(L(x), x-1) + (i - x + 1) * H_x$。

当$i > p$时，$Cost(L(x), i) = Cost(x+1, i) + (x - L(x) + 1) * H_x$。

子任务5

所以对于$i > x$的部分，存在一个p使得：

当$i \leq p$时，$Cost(L(x), i) = Cost(L(x), x-1) + (i - x + 1) * H_x$。

当$i > p$时，$Cost(L(x), i) = Cost(x+1, i) + (x - L(x) + 1) * H_x$。

我们可以使用二分求出p的值，然后左边就是一个等差数列，右边是$S(Rson(x))$的一段加上定值，函数式线段树都能很好的维护这些东西。

子任务5

所以对于$i > x$的部分，存在一个p使得：

当$i \leq p$时，$Cost(L(x), i) = Cost(L(x), x-1) + (i - x + 1) * H_x$。

当$i > p$时，$Cost(L(x), i) = Cost(x+1, i) + (x - L(x) + 1) * H_x$。

我们可以使用二分求出p的值，然后左边就是一个等差数列，右边是$S(Rson(x))$的一段加上定值，函数式线段树都能很好的维护这些东西。

总复杂度$O((N+Q)\log N)$。

Korea 2017 (采摘贝类)

给出一个$N * N$ 的矩阵，每个格子能够收集$A_{i,j}$ 个贝类。

定义$B_{i,j}$ 为从(i, j) 出发，只能向上向左走能够收集的贝类最大值。求$\sum B_{i,j}$ 。

同时还有N 次修改操作，每次是将某个$A_{i,j}$ 加1 或减1 ，并求新的$\sum B_{i,j}$ 。

$N \leq 1500$。

Korea 2017 (采摘贝类)

对于一次修改，可以发现修改的范围一定是一个连通区域，并且左边界一定不会向左移动，上边界也不会向上移动。

Korea 2017 (采摘贝类)

对于一次修改，可以发现修改的范围一定是一个连通区域，并且左边界一定不会向左移动，上边界也不会向上移动。

那么我们只需要求出修改的范围的边界即可，然后用线段树来维护。

Korea 2017 (采摘贝类)

对于一次修改，可以发现修改的范围一定是一个连通区域，并且左边界一定不会向左移动，上边界也不会向上移动。

那么我们只需要求出修改的范围的边界即可，然后用线段树来维护。

时间复杂度$O(N^2 \log n)$。

USA 2018 Train Tracking

有一辆N 节车厢的列车，每节车厢上有一个数字c_0 。列车只被你从前往后观察两次，同时你只有5500个能保存int范围内数的笔记本。你需要回答出按顺序回答出每相邻k 个最小值。

具体来说，你需要实现一个函数helpBessie(ID)，表示你当前看到的这节车厢的编号。在函数中你能够对于笔记本进行操作，进行一次回答，查询n, k的值，当前列车的数值，当前是第几次观察。

$n \leq 10^6$

USA 2018 Train Tracking

令$f(i)$表示$[i, i+k)$的最小值的位置。显然有$f(i) \leq f(i+1)$。

USA 2018 Train Tracking

令$f(i)$表示$[i, i+k)$的最小值的位置。显然有$f(i) \leq f(i+1)$。

我们第一次观察时，我们想办法求出$f(0), f(M), f(2M), \ldots, f(N-M)$的值，其中$M = O(\sqrt{N})$。

USA 2018 Train Tracking

令$f(i)$表示$[i, i+k)$的最小值的位置。显然有$f(i) \leq f(i+1)$。

我们第一次观察时，我们想办法求出$f(0), f(M), f(2M), \ldots, f(N-M)$的值，其中$M = O(\sqrt{N})$。

具体的方法是，我们使用单调队列，因为$[iM, iM+k)$是由若干个整块和最后一小截组成，所以在单调队列中，同一块的元素就只需要保留最小的一个即可。那么这样使用的空间便是$O(\sqrt{N})$的。

USA 2018 Train Tracking

在第二次观察的时候，我们需要求出所有的f。

因为$f(iM) \leq f(iM+1) \leq f(iM+2) \leq \cdots \leq f((i+1)M)$。

USA 2018 Train Tracking

在第二次观察的时候，我们需要求出所有的f。

因为$f(iM) \leq f(iM+1) \leq f(iM+2) \leq \cdots \leq f((i+1)M)$。

所以我们只需要考虑在$[f(iM), f((i+1)M)]$中的元素。

USA 2018 Train Tracking

在第二次观察的时候，我们需要求出所有的f。

因为$f(iM) \leq f(iM+1) \leq f(iM+2) \leq \cdots \leq f((i+1)M)$。

所以我们只需要考虑在$[f(iM), f((i+1)M)]$中的元素。

而其中只有$O(M)$个元素是只会影响部分点，其他的元素会影响这一段内的所有f。所以只要用单调队列维护那$O(M)$个值即可。

USA 2018 Train Tracking

在第二次观察的时候，我们需要求出所有的f。

因为$f(iM) \leq f(iM+1) \leq f(iM+2) \leq \cdots \leq f((i+1)M)$。

所以我们只需要考虑在$[f(iM), f((i+1)M)]$中的元素。

而其中只有$O(M)$个元素是只会影响部分点，其他的元素会影响这一段内的所有f。所以只要用单调队列维护那$O(M)$个值即可。

总时间复杂度$O(N)$，空间复杂度$O(\sqrt{N})$。

Japan 2018 Airline Route Map

你需要实现两个程序。

你的第一个程序，会得到一张n 个点m 条边的无向图。你需要给出一张n' 个点的无向图。

然后交互库会将会将这个图的随机重标号（包括点和边）并交个第二个程序。

你的第二个程序需要将图还原出来。

子任务1（22分）：$m \leq 10$。

子任务2（15分）：$n \leq 40$。

子任务2（63分）：$n \leq 1000$。要求$n' - n \leq 12$。若$n' - n \leq 20$可获得至少33分。

Japan 2018 Airline Route Map

我们显然是只能够是保留所有原图的边，然后利用额外的12个点来确定原图的每个点的编号，从而来还原。

Japan 2018 Airline Route Map

我们显然是只能够是保留所有原图的边，然后利用额外的12个点来确定原图的每个点的编号，从而来还原。

很显然的我们可以利用10个点来标记每一个点的二进制位，例如如果第i位为1则与额外i号点连边。那么我们便要想办法识别出这10个点。

Japan 2018 Airline Route Map

我们显然是只能够是保留所有原图的边，然后利用额外的12个点来确定原图的每个点的编号，从而来还原。

很显然的我们可以利用10个点来标记每一个点的二进制位，例如如果第i位为1则与额外i号点连边。那么我们便要想办法识别出这10个点。

我们显然只能够在度数上做文章。可以注意到，因为$n < 1023$，所以不存在一个点连接了所有点。我们可以令额外11号点向除了额外12号的其他点连边。那么度数最大的点就是额外11号点，唯一与它没有边的便是额外12号点。

Japan 2018 Airline Route Map

我们显然是只能够是保留所有原图的边，然后利用额外的12个点来确定原图的每个点的编号，从而来还原。

很显然的我们可以利用10个点来标记每一个点的二进制位，例如如果第i位为1则与额外i号点连边。那么我们便要想办法识别出这10个点。

我们显然只能够在度数上做文章。可以注意到，因为$n < 1023$，所以不存在一个点连接了所有点。我们可以令额外11号点向除了额外12号的其他点连边。那么度数最大的点就是额外11号点，唯一与它没有边的便是额外12号点。

然后令额外12号点只向额外$1 \sim 10$号点连边，那么这样就可以找出这10个点。而如何识别它们的编号呢？

Japan 2018 Airline Route Map

我们显然是只能够是保留所有原图的边，然后利用额外的12个点来确定原图的每个点的编号，从而来还原。

很显然的我们可以利用10个点来标记每一个点的二进制位，例如如果第i位为1则与额外i号点连边。那么我们便要想办法识别出这10个点。

我们显然只能够在度数上做文章。可以注意到，因为$n < 1023$，所以不存在一个点连接了所有点。我们可以令额外11号点向除了额外12号的其他点连边。那么度数最大的点就是额外11号点，唯一与它没有边的便是额外12号点。

然后令额外12号点只向额外$1 \sim 10$号点连边，那么这样就可以找出这10个点。而如何识别它们的编号呢？

一种方法是从1到10号依次连成一条链，然后令1号点向除了10号点以外的点连边。

Japan 2018 Security Gate

一个括号序列被称为好的当且仅当将某一连续子段取反后是一个合法的括号序列。现在有一个带有通配符的括号序列，求有多少个好的括号序列能够匹配上。

$n \leq 300$。

子任务1（4分）：$n \leq 100$，通配符数量不超过4 。

子任务2（8分）：$n \leq 100$，通配符数量不超过12 。

子任务3（18分）：$n \leq 100$，通配符数量不超过20 。

子任务4（43分）：$n \leq 100$。

子任务5（27分）：没有追加限制。

Japan 2018 Security Gate

合法的序列可以分成以下三种：

- 本身就是合法括号序列。
- 只有从某一侧开始的前缀和是非法的。
- 从两侧开始的前缀和都是非法的。

Japan 2018 Security Gate

本身就是合法括号序列

直接$O(n^2)$DP即可。

Japan 2018 Security Gate

只有从某一侧开始的前缀和是非法的

不妨假设从左开始会导致不合法。令S_i为前缀和，令A为前缀和第一次$= -1$之前的最大值，B为S_n。

Japan 2018 Security Gate

只有从某一侧开始的前缀和是非法的

不妨假设从左开始会导致不合法。令S_i为前缀和，令A为前缀和第一次$= -1$之前的最大值，B为S_n。

最优的情况下，取反的子段的左端点一定是A的位置，右端点就是前缀和为$A + \frac{B}{2}$的位置。

Japan 2018 Security Gate

只有从某一侧开始的前缀和是非法的

不妨假设从左开始会导致不合法。令S_i为前缀和，令A为前缀和第一次$= -1$之前的最大值，B为S_n。

最优的情况下，取反的子段的左端点一定是A的位置，右端点就是前缀和为$A + \frac{B}{2}$的位置。

若要使得取反后前缀和≥ 0，那么取反区间的所有值都要$\leq 2A$。

Japan 2018 Security Gate

只有从某一侧开始的前缀和是非法的

不妨假设从左开始会导致不合法。令S_i为前缀和，令A为前缀和第一次$= -1$之前的最大值，B为S_n。

最优的情况下，取反的子段的左端点一定是A的位置，右端点就是前缀和为$A + \frac{B}{2}$的位置。

若要使得取反后前缀和≥ 0，那么取反区间的所有值都要$\leq 2A$。

我们可以是枚举A和B的值，再进行$O(n^2)$的DP即可。

Japan 2018 Security Gate

只有从某一侧开始的前缀和是非法的

考虑优化，我把序列分成两个阶段，第一次小于0之前和之后。

Japan 2018 Security Gate

只有从某一侧开始的前缀和是非法的

考虑优化，我把序列分成两个阶段，第一次小于0之前和之后。

对于前面的部分限制只与A有关。对于后面的部分，如果我们将所有的S都减B，那么反转子段的右端点的前缀和就应该是$A-\frac{B}{2}$，并且取反区间的限制变为了$\leq 2A-B$。这都只与$A-\frac{B}{2}$有关，所以可以更加这一值来预处理。

Japan 2018 Security Gate

只有从某一侧开始的前缀和是非法的

考虑优化，我把序列分成两个阶段，第一次小于0之前和之后。

对于前面的部分限制只与A有关。对于后面的部分，如果我们将所有的S都减B，那么反转子段的右端点的前缀和就应该是$A-\frac{B}{2}$，并且取反区间的限制变为了$\leq 2A-B$。这都只与$A-\frac{B}{2}$有关，所以可以更加这一值来预处理。

复杂度$O(n^3)$。

Janpan 2018 Security Gate

从两侧开始的前缀和都是非法的

令C为S_n，A为前缀和第一次$= -1$之前的最大值，B为前缀和最后一次$= C - 1$之后的最大值。

Janpan 2018 Security Gate

从两侧开始的前缀和都是非法的

令C为S_n，A为前缀和第一次$= -1$之前的最大值，B为前缀和最后一次$= C-1$之后的最大值。

如果$A + \frac{C}{2} \leq B$，那么最优情况下，取反子段的左端点一定是A的位置，而右端点需要是最后一次$= C-1$之后的等于$A + \frac{C}{2}$的位置。

Janpan 2018 Security Gate

从两侧开始的前缀和都是非法的

令C为S_n，A为前缀和第一次$= -1$之前的最大值，B为前缀和最后一次$= C-1$之后的最大值。

如果$A + \frac{C}{2} \leq B$，那么最优情况下，取反子段的左端点一定是A的位置，而右端点需要是最后一次$= C-1$之后的等于$A + \frac{C}{2}$的位置。

对于这个可以是按照最后一次$= C-1$的位置分段，前一段只与A有关，后一段只与$B-C$有关。

Janpan 2018 Security Gate

从两侧开始的前缀和都是非法的

令C为S_n，A为前缀和第一次$= -1$之前的最大值，B为前缀和最后一次$= C-1$之后的最大值。

如果$A + \frac{C}{2} \leq B$，那么最优情况下，取反子段的左端点一定是A的位置，而右端点需要是最后一次$= C-1$之后的等于$A + \frac{C}{2}$的位置。

对于这个可以是按照最后一次$= C-1$的位置分段，前一段只与A有关，后一段只与$B-C$有关。

然后对于$A + \frac{C}{2} > B$的情况，倒过来再做一次就好了。

Japan 2017 Cultivation

在一个$R*C$的网格上，初始时有N个格子种有草。每一年结束时，你可以决定一场风，使得所有种有草的地方会沿着风的方向传播一格。问最少多少年能使得所有格子上都有草。

$R, C \leq 10^9, N \leq 300$

子任务1（5分）：$R, C \leq 4$。

子任务2（10分）：$R, C \leq 40$。

子任务3（15分）：$R \leq 40$。

子任务4（30分）：$N \leq 25$。

子任务5（20分）：$N \leq 100$。

子任务6（20分）：没有追加限制。

Japan 2017 Cultivation

子任务2

我们首先可以很显然的得到一个naive的想法。

Japan 2017 Cultivation

子任务2

我们首先可以很显然的得到一个naive的想法。

枚举一共吹了a场东风，b场南风，c场西风，d场北风。然后维护一下矩阵并。

Japan 2017 Cultivation

子任务2

我们首先可以很显然的得到一个naive的想法。

枚举一共吹了a场东风，b场南风，c场西风，d场北风。然后维护一下矩阵并。

复杂度$O(R^3C^3)$或$O(R^2C^2N\log n)$。

Japan 2017 Cultivation

子任务3

如果我们先决定了南北风的场次，那么对于每一行我们可以得到一个对于东西风的限制，即东风至少a场，西风至少b场，合计至少c场。

Japan 2017 Cultivation

子任务3

如果我们先决定了南北风的场次，那么对于每一行我们可以得到一个对于东西风的限制，即东风至少a场，西风至少b场，合计至少c场。

复杂度$O(R^3N)$。

Japan 2017 Cultivation

子任务4

下面改成先决定东西风。可以发现说，如果西风的场数不是某个$x_i - 1$的话，那么将西风场数-1，东风场数$+1$，同样还是一组合法的解。那么西风的可能情况就只有$O(N)$种。

Japan 2017 Cultivation

子任务4

下面改成先决定东西风。可以发现说，如果西风的场数不是某个$x_i - 1$的话，那么将西风场数-1，东风场数$+1$，同样还是一组合法的解。那么西风的可能情况就只有$O(N)$种。

然后东风的场数一定会是某个$x_i - x_j - l - 1$或$C - x_i$。一共只有$O(N^2)$种情况。

Japan 2017 Cultivation

子任务4

下面改成先决定东西风。可以发现说，如果西风的场数不是某个$x_i - 1$的话，那么将西风场数-1，东风场数$+1$，同样还是一组合法的解。那么西风的可能情况就只有$O(N)$种。

然后东风的场数一定会是某个$x_i - x_j - l - 1$或$C - x_i$。一共只有$O(N^2)$种情况。

因为决定了东西风后，本质不同的列只有$O(N)$种，直接计算即可。

Japan 2017 Cultivation

子任务4

下面改成先决定东西风。可以发现说，如果西风的场数不是某个$x_i - 1$的话，那么将西风场数-1，东风场数$+1$，同样还是一组合法的解。那么西风的可能情况就只有$O(N)$种。

然后东风的场数一定会是某个$x_i - x_j - l - 1$或$C - x_i$。一共只有$O(N^2)$种情况。

因为决定了东西风后，本质不同的列只有$O(N)$种，直接计算即可。

复杂度$O(N^5)$。

Japan 2017 Cultivation

子任务5

可以发现说东风+西风的场数只有$O(N^2)$种情况。

Japan 2017 Cultivation

子任务5

可以发现说东风+西风的场数只有$O(N^2)$种情况。

刮a场东风、b场西风的情况可以看成是，刮$a+b$场东风，然后网格向东移动b格。

Japan 2017 Cultivation

子任务5

可以发现说东风+西风的场数只有$O(N^2)$种情况。

刮a场东风、b场西风的情况可以看成是，刮$a+b$场东风，然后网格向东移动b格。

那么这样就只需要枚举东风+西风的场数，然后用单调队列来维护网格的移动。

Japan 2017 Cultivation

子任务5

可以发现说东风+西风的场数只有$O(N^2)$种情况。

刮a场东风、b场西风的情况可以看成是，刮$a+b$场东风，然后网格向东移动b格。

那么这样就只需要枚举东风+西风的场数，然后用单调队列来维护网格的移动。

复杂度$O(N^4)$。

Japan 2017 Cultivation

子任务6

我们现在的瓶颈在于我们每次需要重新计算对于南北风的限制。

Japan 2017 Cultivation

子任务6

我们现在的瓶颈在于我们每次需要重新计算对于南北风的限制。

可以发现，当东风+西风的场数递增的时候，每一段中的被覆盖的行不会减少，所以总共只会有$O(N^2)$次修改。

Japan 2017 Cultivation

子任务6

我们现在的瓶颈在于我们每次需要重新计算对于南北风的限制。

可以发现，当东风+西风的场数递增的时候，每一段中的被覆盖的行不会减少，所以总共只会有$O(N^2)$次修改。

复杂度$O(N^3)$。

Japan 2017 Sparklers

一共n 个人站在数轴上，每个人拥有一只能够燃烧T 秒的烟火。最初只有第k 个人的烟火是点燃的，当两个人处于同一位置时可以将火焰传递过去。同时，所有人的移动速度不能超过s 。

现在想要知道如果要让火焰燃烧恰好nT 秒，s 的最小可行值是多少。

$n \leq 100000$ 。

子任务1（30分）：$N \leq 20$ 。

子任务2（20分）：$N \leq 1000$ 。

子任务3（50分）：没有追加限制。

Japan 2017 Sparklers

答案具有可二分性，那么我们就只需要考虑如何检验是否合法。

Japan 2017 Sparklers

答案具有可二分性，那么我们就只需要考虑如何检验是否合法。

可以观察发现，不管拿着点燃的烟火的人怎么走，其他的人都是向着他移动的。

Japan 2017 Sparklers

答案具有可二分性，那么我们就只需要考虑如何检验是否合法。

可以观察发现，不管拿着点燃的烟火的人怎么走，其他的人都是向着他移动的。

那么任意情况下，能聚到一起的人肯定是一段区间。

Japan 2017 Sparklers

答案具有可二分性，那么我们就只需要考虑如何检验是否合法。

可以观察发现，不管拿着点燃的烟火的人怎么走，其他的人都是向着他移动的。

那么任意情况下，能聚到一起的人肯定是一段区间。

我们用$f[l][r]$表示$[l,r]$内的人，能否在燃烧的时间内聚在一起。

Japan 2017 Sparklers

答案具有可二分性，那么我们就只需要考虑如何检验是否合法。

可以观察发现，不管拿着点燃的烟火的人怎么走，其他的人都是向着他移动的。

那么任意情况下，能聚到一起的人肯定是一段区间。

我们用$f[l][r]$表示$[l,r]$内的人，能否在燃烧的时间内聚在一起。

可以发现如果只观察l和r两个的移动，他们在这段时间内一定是向着对方移动的。那么如果可行的话，就必定需要满足$(x[r]-x[l])/2s \leq (r-l)t$。同时还需要满足$[l,r-1]$或$[l+1,r]$有一个满足。

Japan 2017 Sparklers

答案具有可二分性，那么我们就只需要考虑如何检验是否合法。

可以观察发现，不管拿着点燃的烟火的人怎么走，其他的人都是向着他移动的。

那么任意情况下，能聚到一起的人肯定是一段区间。

我们用$f[l][r]$表示$[l,r]$内的人，能否在燃烧的时间内聚在一起。

可以发现如果只观察l和r两个的移动，他们在这段时间内一定是向着对方移动的。那么如果可行的话，就必定需要满足$(x[r]-x[l])/2s \le (r-l)t$。同时还需要满足$[l,r-1]$或$[l+1,r]$有一个满足。

相当于就是判断是否存在一个$[k,k]$到$[1,n]$的方案使得每个区间都满足条件。

Japan 2017 Sparklers

$$(x[r]-x[l])/2s \le (r-l)t \Leftrightarrow x[r]-2str \le x[l]-2str$$

令$y[i]=x[i]-2sti$。

Japan 2017 Sparklers

$$(x[r]-x[l])/2s \le (r-l)t \Leftrightarrow x[r]-2str \le x[l]-2str$$

令$y[i]=x[i]-2sti$。

我们考虑贪心的移动，即当前为$[l,r]$，若存在一个$l'<l$满足$\forall l' \le i < i, y[i] \ge y[r]$和$y[l'] > y[l]$，那么可以移动至$[l',r]$。$r$的移动也类似。

Japan 2017 Sparklers

$$(x[r]-x[l])/2s \le (r-l)t \Leftrightarrow x[r]-2str \le x[l]-2str$$

令$y[i]=x[i]-2sti$。

我们考虑贪心的移动，即当前为$[l,r]$，若存在一个$l'<l$满足$\forall l' \le i < i, y[i] \ge y[r]$和$y[l'] > y[l]$，那么可以移动至$[l',r]$。$r$的移动也类似。

若令L为$1 \sim k$中y最大的下标，令R为$k \sim n$中y最小的下标。根据这一个贪心思想，我们只能移动到$[L,R]$。

Japan 2017 Sparklers

$$(x[r]-x[l])/2s \le (r-l)t \Leftrightarrow x[r]-2str \le x[l]-2str$$

令$y[i]=x[i]-2sti$。

我们考虑贪心的移动，即当前为$[l,r]$，若存在一个$l'<l$满足$\forall l' \le i < i, y[i] \ge y[r]$和$y[l'] > y[l]$，那么可以移动至$[l',r]$。$r$的移动也类似。

若令L为$1 \sim k$中y最大的下标，令R为$k \sim n$中y最小的下标。根据这一个贪心思想，我们只能移动到$[L,R]$。

因为移动是可逆的，所以只要$[1,n]$能够移动到$[L,R]$就行。

Japan 2017 Arranging Tickets

在一个环形的铁路上有n 个站，一份套票包含1 到n 这n 张票，其中编号为i 的票能从i 到$i+1$ 或从$i+1$ 到i 。票只能整套购买。

现在有m 批人想要乘坐铁路，第i 批人是想从A_i 到B_i 的C_i 个人。

求问最少买多少份套票能够满足所有人的条件。

$N \leq 200000, M \leq 100000$

子任务1（10分）：$N, M \leq 20, C_i = 1$ 。

子任务2（35分）：$N, M \leq 300, C_i = 1$ 。

子任务3（20分）：$N, M \leq 3000, C_i = 1$ 。

子任务4（20分）：$C_i = 1$ 。

子任务5（15分）：没有追加限制。

Japan 2017 Arranging Tickets

子任务2

显然答案具有可二分性。那我们只需要考虑ans是否合法。

Japan 2017 Arranging Tickets

子任务2

显然答案具有可二分性。那我们只需要考虑ans是否合法。

我们断环为链，然后所以的人都是一个$[A_i, B_i)$的区间，并且可以将其翻转为$[1, A_i) \bigcup [B_i, n]$。

Japan 2017 Arranging Tickets

子任务2

显然答案具有可二分性。那我们只需要考虑ans是否合法。

我们断环为链，然后所以的人都是一个$[A_i, B_i)$的区间，并且可以将其翻转为$[1, A_i) \bigcup [B_i, n]$。

可以观察发现，所有翻转的区间一定是交集非空。

Japan 2017 Arranging Tickets

子任务2

显然答案具有可二分性。那我们只需要考虑ans是否合法。

我们断环为链，然后所以的人都是一个$[A_i, B_i)$的区间，并且可以将其翻转为$[1, A_i) \bigcup [B_i, n]$。

可以观察发现，所有翻转的区间一定是交集非空。（取两个不交的区间，将其反转后会更劣。）

Japan 2017 Arranging Tickets

子任务2

显然答案具有可二分性。那我们只需要考虑ans是否合法。

我们断环为链，然后所以的人都是一个$[A_i, B_i)$的区间，并且可以将其翻转为$[1, A_i) \bigcup [B_i, n]$。

可以观察发现，所有翻转的区间一定是交集非空。（取两个不交的区间，将其反转后会更劣。）

那么我们可以是枚举交点t和需要翻转的区间个数cnt，然后我们便可以进行贪心了。

Japan 2017 Arranging Tickets

子任务2

贪心的方法是维护一个以右端点为关键字的大根堆，令i从1枚举到t，然后：

- 将所有左端点是i且跨过了t的区间加入堆。
- 如果当前已经翻转过的区间个数还没有达到这个位置合法所需要的次数，那么便需要将堆顶元素翻转，如果堆为空的话则不合法。

最后再检验这一方法是否合法。

Japan 2017 Arranging Tickets

子任务2

贪心的方法是维护一个以右端点为关键字的大根堆，令i从1枚举到t，然后：

- 将所有左端点是i且跨过了t的区间加入堆。
- 如果当前已经翻转过的区间个数还没有达到这个位置合法所需要的次数，那么便需要将堆顶元素翻转，如果堆为空的话则不合法。

最后再检验这一方法是否合法。

注意到我们是按照左端点从小到大的顺序贪心，因此后面的翻转决策可能会导致前面的位置被覆盖次数增加，所以我们不可以单纯按照当前位置初始覆盖次数和ans来决定选取的区间个数。

Japan 2017 Arranging Tickets

子任务2

贪心的方法是维护一个以右端点为关键字的大根堆，令i从1枚举到t，然后：

- 将所有左端点是i且跨过了t的区间加入堆。
- 如果当前已经翻转过的区间个数还没有达到这个位置合法所需要的次数，那么便需要将堆顶元素翻转，如果堆为空的话则不合法。

最后再检验这一方法是否合法。

注意到我们是按照左端点从小到大的顺序贪心，因此后面的翻转决策可能会导致前面的位置被覆盖次数增加，所以我们不可以单纯按照当前位置初始覆盖次数和ans来决定选取的区间个数。

假设之前的翻转次数是y，还有T次翻转机会。那么当前所需要翻转的区间个数x需要满足$y - x + T - x < ans$。

Japan 2017 Arranging Tickets

子任务2

贪心的方法是维护一个以右端点为关键字的大根堆，令i从1枚举到t，然后：

- 将所有左端点是i且跨过了t的区间加入堆。
- 如果当前已经翻转过的区间个数还没有达到这个位置合法所需要的次数，那么便需要将堆顶元素翻转，如果堆为空的话则不合法。

最后再检验这一方法是否合法。

注意到我们是按照左端点从小到大的顺序贪心，因此后面的翻转决策可能会导致前面的位置被覆盖次数增加，所以我们不可以单纯按照当前位置初始覆盖次数和ans来决定选取的区间个数。

假设之前的翻转次数是y，还有T次翻转机会。那么当前所需要翻转的区间个数x需要满足$y - x + T - x < ans$。

复杂度$O(mn^2 \log^2 n)$。

Japan 2017 Arranging Tickets

子任务3

令b_i为翻转后，每个位置被覆盖的次数。假设一种最优方案的交集是$[l, r)$，我们令其中的最大值是b_t。

Japan 2017 Arranging Tickets

子任务3

令b_i为翻转后，每个位置被覆盖的次数。假设一种最优方案的交集是$[l, r)$，我们令其中的最大值是b_t。

可以证明一定存在一组最优方案使得$b_t = maxb_i$或$b_t = maxb_i - 1$。

Japan 2017 Arranging Tickets

子任务3

令b_i为翻转后，每个位置被覆盖的次数。假设一种最优方案的交集是$[l, r)$，我们令其中的最大值是b_t。

可以证明一定存在一组最优方案使得$b_t = maxb_i$或$b_t = maxb_i - 1$。

证明：$b_t \leq maxb_i - 2$，那么我们选取翻转的区间中左端点最大和右端点最小的两个区间（可能是同一个），并取消他们的翻转，那么同样还是合法的。不断指向这一过程，一定可以使得$b_t \geq maxb_i - 1$。

Japan 2017 Arranging Tickets

子任务3

令b_i为翻转后，每个位置被覆盖的次数。假设一种最优方案的交集是$[l, r)$，我们令其中的最大值是b_t。

可以证明一定存在一组最优方案使得$b_t = maxb_i$或$b_t = maxb_i - 1$。

证明：$b_t \leq maxb_i - 2$，那么我们选取翻转的区间中左端点最大和右端点最小的两个区间（可能是同一个），并取消他们的翻转，那么同样还是合法的。不断指向这一过程，一定可以使得$b_t \geq maxb_i - 1$。

又由于$maxb_i$一定是ans，所以cnt的取值只有t的初始被覆盖次数$-0/1$。

Japan 2017 Arranging Tickets

子任务5

令a_i为初始的被覆盖次数，可以证明$a_t = max a_i$。

Japan 2017 Arranging Tickets

子任务5

令a_i为初始的被覆盖次数，可以证明$a_t = max a_i$。

证明：若$a_t \neq max a_i$，那么存在一个不在交集的i满足$a_i \geq a_t + 1$。那么一定有$b_i - a_i \geq b_t - a_t + 1$，所以可以得到$b_t + 2 \leq b_i$，与之前的假设矛盾。

Japan 2017 Arranging Tickets

子任务5

令a_i为初始的被覆盖次数，可以证明$a_t = max a_i$。

证明：若$a_t \neq max a_i$，那么存在一个不在交集的i满足$a_i \geq a_t + 1$。那么一定有$b_i - a_i \geq b_t - a_t + 1$，所以可以得到$b_t + 2 \leq b_i$，与之前的假设矛盾。

如果有多个$max a_i$，我们也只需要选取第一个和最后一个即可。（同样考虑反证法）

Japan 2017 Arranging Tickets

子任务5

令a_i为初始的被覆盖次数，可以证明$a_t = max a_i$。

证明：若$a_t \neq max a_i$，那么存在一个不在交集的i满足$a_i \geq a_t + 1$。那么一定有$b_i - a_i \geq b_t - a_t + 1$，所以可以得到$b_t + 2 \leq b_i$，与之前的假设矛盾。

如果有多个$max a_i$，我们也只需要选取第一个和最后一个即可。（同样考虑反证法）

复杂度$O(n \log^2 n)$。

Japan 2017 Natural Park

现在有一个n 个点的图，并且每个点只有不超过7 个相邻节点。

你可以通过不超过45000次询问，每次询问为A 和B 能否只通过给定点连通，来求出图中的所有边。

$n \leq 1400, m \leq 1500$。

子任务1（10分）：$N \leq 250$ 。

子任务2（35分）：图是一条链。

子任务3（20分）：图是一个直径不超过8的树。

子任务4（20分）：图是树。

子任务5（15分）：没有追加限制。

Japan 2017 Natural Park

子任务2

对于链的情况，我们考虑如何从一段已知的链，扩展到一个点x。

Japan 2017 Natural Park

子任务2

对于链的情况，我们考虑如何从一段已知的链，扩展到一个点x。

首先可以是使用一次询问找到较近的一端点y。

Japan 2017 Natural Park

子任务2

对于链的情况，我们考虑如何从一段已知的链，扩展到一个点x。

首先可以是使用一次询问找到较近的一端点y。

然后利用二分便可以找到$x \to y$ 路径上的编号最小点z。递归$x \to z$，$z \to y$即可。

Japan 2017 Natural Park

子任务2

对于链的情况，我们考虑如何从一段已知的链，扩展到一个点x。

首先可以是使用一次询问找到较近的一端点y。

然后利用二分便可以找到$x \to y$ 路径上的编号最小点z。递归$x \to z$，$z \to y$即可。

复杂度$O(n \log n)$。

Japan 2017 Natural Park

子任务4

对于树的情况，我们令y为已知树上的一个点。

Japan 2017 Natural Park

子任务4

对于树的情况，我们令y为已知树上的一个点。

我们同样使用二分就可以找到直接与树相邻的点z，然后再利用bfs序就可以二分得出与z相邻的树上的点。

Japan 2017 Natural Park

子任务4

对于树的情况，我们令y为已知树上的一个点。

我们同样使用二分就可以找到直接与树相邻的点z，然后再利用bfs序就可以二分得出与z相邻的树上的点。

次数$2n \log n$。

Japan 2017 Natural Park

对于图的情况，我们同样还是找到直接与已知图相邻的点z。

Japan 2017 Natural Park

对于图的情况，我们同样还是找到直接与已知图相邻的点z。

然后找到一个已知图上与z相邻的点。

Japan 2017 Natural Park

对于图的情况，我们同样还是找到直接与已知图相邻的点z。

然后找到一个已知图上与z相邻的点。找剩下的点，可以是将找到的删去，然后再对于每个联通块不断的调用。

Japan 2017 Natural Park

对于图的情况，我们同样还是找到直接与已知图相邻的点z。

然后找到一个已知图上与z相邻的点。找剩下的点，可以是将找到的删去，然后再对于每个联通块不断的调用。

总询问次数$7m + (n + m) \log n$。

Russia 2018 Квантовая телепортация

平面上有k 个点，坐标为(x_i, y_i) 。从第i 个点到第j 个点的代价是$2^{max(|x_i-x_j|,|y_i-y_j|)}$。求点1 到点$n$ 的最小代价路径。

$2 \leq k \leq 10000, 1 \leq x_i, y_i \leq 10000$。

子任务1（21分）：$k, x_i, y_i \leq 20$。

子任务2（13分）：$n, x_i, y_i \leq 500$。

子任务3（33分）：每一行每一列只有一个点。

子任务4（33分）：没有附加限制。

Russia 2018 Квантовая телепортация

对于每个点，可以发现，以其为中心画一个十字，那么只需要向每一个块内距离最近的一些点连边即可。（证明？）

Russia 2018 Квантовая телепортация

对于每个点，可以发现，以其为中心画一个十字，那么只需要向每一个块内距离最近的一些点连边即可。（证明？）

因为连边的点都在一条线上，所以需要用线段树来优化连边。

Russia 2018 Квантовая телепортация

对于每个点，可以发现，以其为中心画一个十字，那么只需要向每一个块内距离最近的一些点连边即可。（证明？）

因为连边的点都在一条线上，所以需要用线段树来优化连边。

然后对于距离的维护可以使用函数式线段树。

Russia 2017 Тигры

平面上有n 个点，一共有q 次询问。

每个询问是交互库知道另一个在平面上的点P 。你每次可以选择一个由n 个点中的若干个点组成的凸多边形，询问P 是不是在这个凸多边形中。你的目标是找到一个包含P 的极小凸多边形，即多边形中不含其他的点。

$n \leq 2500, q \leq 2000$ ，询问次数不超过40 次。

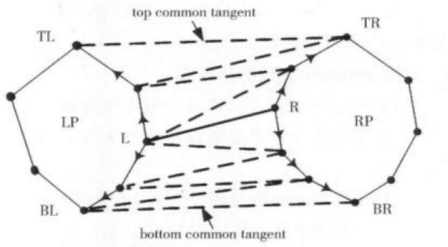
Russia 2017 Тигры

我们考虑Delauney三角剖分的分治算法。

Russia 2017 Тигры

我们考虑Delauney三角剖分的分治算法。

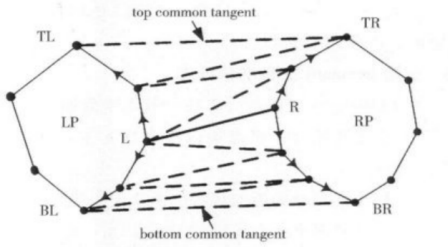
它的思想是每次将点集按照x坐标分成两个集合，分治下去，再在两个凸包直接进行三角剖分。



Russia 2017 Тигры

我们考虑Delauney三角剖分的分治算法。

它的思想是每次将点集按照x坐标分成两个集合，分治下去，再在两个凸包直接进行三角剖分。



可以发现，如果点在两边的凸包中时，我们便选择递归下去。如果在中间部分时，可以发现三角剖分给了我们一个极好的二分结构，因为任意两条线直接的所有三角形都可以用一个凸多边形覆盖，且不包含其他三角形。

我们直接利用这一结构二分即可。

Russia 2017 Траекторияобучения

在两个学期中分别开了n 和m 门课，每门课上完后可以得到一定的收益。同一学期内的课互不相同，不同学期的同一门课不能都上（收益可能不同）。每个学期上的课必须要是一段连续的区间。求最大收益。

$n, m \leq 500000$。

60%:$n, m \leq 10000$。

Russia 2017 Траекторияобучения

如果只在一个学期上课，那么显然是全部选择。

Russia 2017 Траекторияобучения

如果只在一个学期上课，那么显然是全部选择。

如果要在两个学期都上课的话，那么必然有一个学期的区间需要跨过收益的中位数。

Russia 2017 Траекторияобучения

如果只在一个学期上课，那么显然是全部选择。

如果要在两个学期都上课的话，那么必然有一个学期的区间需要跨过收益的中位数。

不妨假设强制第一个学期必须跨过。我们令$f[l][r]$为第二个学期的区间为$[l, r]$的最大值。

Russia 2017 Траекторияобучения

如果只在一个学期上课，那么显然是全部选择。

如果要在两个学期都上课的话，那么必然有一个学期的区间需要跨过收益的中位数。

不妨假设强制第一个学期必须跨过。我们令$f[l][r]$为第二个学期的区间为$[l,r]$的最大值。

因为第一个学期必须跨过中位数，所以第一个学期对于$f[l][r]$的贡献便十分容易计算。即一定是从中间向两边能选就选。

Russia 2017 Траекторияобучения

如果只在一个学期上课，那么显然是全部选择。

如果要在两个学期都上课的话，那么必然有一个学期的区间需要跨过收益的中位数。

不妨假设强制第一个学期必须跨过。我们令$f[l][r]$为第二个学期的区间为$[l,r]$的最大值。

因为第一个学期必须跨过中位数，所以第一个学期对于$f[l][r]$的贡献便十分容易计算。即一定是从中间向两边能选就选。

那么从小到大枚举l，然后用线段树来维护$f[l][r]$的最大值即可。