数据结构5

洛谷省选计划

吾王美如画

www.luogu.com.cn

CDQ分治

P3810 【模板】三维偏序（陌上花开）

- 有 n 个元素，第 i 个元素有 a_i, b_i, c_i 三个属性，设 $f(i)$ 表示$a_j \le a_i$ 且 $b_j \le b_i$ 且 $c_j \le c_i$ 且 $j \ne i$ 的 j 的数量。
- 对于 $d \in [0, n)$，求 $f(i) = d$ 的数量
- $1 \le n \le 10^5$,$1 \le a_i, b_i, c_i \le k \le 2 \times 10^5$

- 我们先考虑二维的情况，先将 a_i 从小到大排序，之后顺序统计有多少对 $i < j\ s.t. b_i < b_j$ 。后面这部分是一个一维数点，而我们上述的排序过程实际上使得问题的维度减少了一维，我们尝试将这种做法套用在更高维上。

- 对于三维，我们同样将 a_i 从小到大排序，这时问题转化为二维数点，但当我们尝试再进一步排序降维时，发现再次排序会影响 a_i 的有序性。这时我们考虑分治，将点按 a_i 是否大于 mid 分为'左'边和'右'边。对于左（或右）边内部的贡献我们递归下去计算，现在考虑左右之间的贡献。

- 此时，我们只允许‘左’边的点产生贡献，‘右’边的点计算贡献，这样便可以将这些点按 b_i 排序使得问题变为一维数点了。

- 每层递归的代价都是$O(N * \log N)$ 总复杂度为 $O(N * \log^2 N)$

- 该方法可同理延展至更高维。

P4169 [Violet] 天使玩偶/SJY摆棋子

- 在二维平面，初始有n个点 $(0 \le x_i, y_i \le 10^6)$，要求支持两种操作： 加点 以及 查询距离某位置曼哈顿距离最小的点。操作数为m
- $n, m \le 3 \times 10^5$

- 首先观察题目不强制在线，我们考虑将操作离线，每个操作都对对应上一个时间戳。

- 接着我们发现对于原问题，曼哈顿距离中的绝对值很难处理，考虑将其分四部分解决。每次对于某个询问，只求其左下（左上、右下、右上）最近的点。此时曼哈顿距离最小转化成了横纵坐标和最大（同时满足在询问左下方的限制）。

- 我们来梳理一下目前的限制（以左下为例）：

  1. 时间戳小的加点操作向时间戳大的询问作贡献

  2. 加点操作的横纵坐标均小于询问

- 一共是三层限制，本质上便是三维偏序，将三维数点中的计数替换为求最大值即可。

P2487 [SDOI2011] 拦截导弹

现有 n 枚导弹按顺序飞来，第 i 枚的高度为 h_i ，速度为 v_i 。要求选出最长的导弹序列使得对于 $\forall i < j, h_i \geq h_j$ 且 $v_i \geq v_j$ 。当然，可能存在多种最长的序列。现求每个导弹出现在最长序列的概率。

$1 \leq n \leq 5 \times 10^4, 1 \leq h_i, v_i \leq 10^9$

- 首先考虑如何处理这个“出现在最长序列里的概率”。我们可以发现，x 出现在最长序列的概率等价于 $\frac{\text{包含 } x \text{ 的最长序列数}}{\text{总共的最长序列数}}$ 。而要求解包含 x 的最长序列数，其实只需看 x 是否能存在于最长序列中。若能，则其出现次数为：包含 x 的最长前缀数 $\times$ 包含 x 的最长后缀数。
- 这样，我们设 $\text{len}_{\text{pre}}[x]$, $cnt_{pre}[x]$ ，表示以 x 为结尾的前缀里的最长序列长度以及最长序列个数。这两项均可以 $O(n)$ 的时间复杂度进行转移：

$$\begin{cases} len_{pre}[i] = \max(\text{len}_{\text{pre}}[\text{j}]) + 1 \\ cnt_{pre}[i] = \sum_{len_{pre}[j]+1==len_{pre}[i]} cnt_{pre}[j] \end{cases}$$

其中 $i > j$ ，$h_i \le h_j$, $v_i \le v_j$ 。

- 此时我们已经得到一个 $O(N^2)$ 的做法了，但这显然不足以通过，我们考虑优化。而此时，明显比较臃肿的地方在于 $O(N)$ 的转移。我们可以发现，其实此处的限制只有 $i > j$，$h_i \le h_j$，$v_i \le v_j$。注意到我们并不关心转移实际是如何进行，那么本质上这里就变为统计一个三维空间里的立方体的信息，这恰好是cdq分治所描述的问题。
- 但值得注意的是，在dp问题中我们除了应用常规的cdq分治之外，还需保证一个位置在做贡献之前，其必须已经做完了所有的查询。即该位置的dp值必须已经计算好了，才能往后做贡献。对于这题，只需在分治到 $[l,r]$ 区间时，我们总是先递归处理 $[l,mid]$ 区间，之后再计算 $[l,mid]$ 对 $[mid+1,r]$ 区间的贡献，最后再递归处理 $[mid+1,r]$ 即可。这样就能保障在做贡献之前，我们已经处理完了该位置的所有询问从而将该位置的值全部确定下来。

P8253 [NOI Online 2022 提高组] 如何正确地排序

有一个$m \times n$的数组$a_{i,j}$。

定义 $f(i,j) = min_{k=1}^{m}(a_{k,i} + a_{k,j}) + max_{k=1}^{m}(a_{k,i} + a_{k,j})$

求$\sum_{i=1}^{\mathrm{n}} \sum_{j=1}^{m} f(i,j)$

数据范围：$m \leq 4\,, n \leq 2 \times 10^5$

- 首先max和min部分相对独立，我们此处只考虑max部分。$a_{i,j}$作一次贡献，当且仅当 $\exists k\ \ s.t. \forall t \neq i\ (a_{t,j}+a_{t,k} < a_{i,j} + a_{i,k})\ or\ (a_{t,j} + a_{t,k} == a_{i,j} + a_{i,k}\ \&\ t < i)$ （避免相同的值被多次取到）
- 只拿小于的部分考虑，通过移项得 $a_{t,j}-a_{i,j} < a_{i,k} - a_{t,k}$
- 观察到m其实相当小，这使得式子中的 i,t 可以直接暴力枚举。我们以 $i = 1$ 为例。
$\begin{cases} a_{2,j} - a_{1,j} < a_{1,k} - a_{2,k} \\ a_{3,j} - a_{1,j} < a_{1,k} - a_{3,k} \\ a_{4,j} - a_{1,j} < a_{1,k} - a_{4,k} \end{cases}$ 。对于任意 i 只存在类似的三条限制，而其描述的恰好是一个三维偏序问题，用cdq分治解决即可。

整体二分

P3332 [ZJOI2013] K大数查询

维护 n 个可重数集，集合编号从 1到 n。

这些集合初始都是空集，有 m 个操作:

- $1\ l\ r\ c$： 将 c 加入到编号在 $[l,r]$ 内的集合中
- $2\ l\ r\ c$： 查询编号在 $[l,r]$内的集合的并集中，第 c 大的数是多少

$1 \le n,m \le 5\times 10^4$

$1 \le l,r \le n$

1操作中$|c| \le n$

2操作中$1 \le c \le 2^{63}$

- 遇见第k大问题，熟练想到二分。二分mid，将问题转化为是否存在 k 个 $\le mid$ 的数在$[l,r]$范围内。这样修改和询问皆可在线段树上实现。但对于每个mid都把所有操作遍历一次的时间复杂度是不能接受的，我们发现这过程中有很多的过程是重复的，考虑将其一起操作，即整体二分。

- 考虑对于答案区间为$[l,r]$，有操作集合 op （op应保有时间顺序），取$mid=\frac{l+r}{2}$，对于操作$t \in op$

1. 若为添加数 c ， 当 $c \le mid$ 则其一定会给最终答案大于mid的那些查询作贡献，故使线段树$[t_l,t_r]$区间加一，并将该操作放入答案区间 $[l,mid]$ 中做贡献。若$c \ge mid$则其一定不对最终答案小于mid的查询做贡献，直接放入答案区间$[mid+1,r]$
2. 若为查询第c小， 当 $[t_l,t_r]$的区间和$sum \le c$，则表示该查询的最终答案一定大于mid，将 $c -= sum$ ，再将该操作放入答案区间$[mid+1,r]$，否则表明该查询的答案小于等于mid，直接将该操作放入答案区间$[l,mid]$

- 该递归最多log层，每层都是一共枚举m个操作同时维护线段树，总复杂度$O(mlog^2n)$

P3242 [HNOI2015] 接水果

给定一颗n个节点的树。树上有p个盘子，q个水果。盘子和水果都是树上的一条路径。且每个盘子有一个权值。

一个盘子能接住一个水果，当且仅当**盘子的路径是水果的路径的子路径。**

问对于第i个水果，能接住它的所有盘子中，权值第k_i小的盘子。

$1 \leq n, p, q \leq 4 \times 10^4$

- 考虑记节i点时间戳为dfn_i，则 $u \to v$ 包含 $x \to y$ 可表示为

$dfn_x \le dfn_u \le dfn_x + size_x - 1 , dfn_y \le dfn_v \le dfn_y + size_y - 1$（$x,y$不为祖先关系）

$dfn_u \le dfn_x$ or $dfn_x + size_x - 1 \le dfn_u , dfn_y \le dfn_v \le dfn_y + size_y - 1$（$x$为$y$祖先）

- 可以将这个问题转化到二维平面上，每个盘子对应平面上一矩形（或两不相交矩形的并），每个水果即为查询所有覆盖平面上一点的矩形中权值第k小的序号。

- 考虑扫描线的思路，将矩形差分，问题转化为维护多个可重集，支持区间添加一个数，区间减少一个数，查询某个可重集中的第k小。这部分就是之前讲过的k大数查询。

线段树合并

P4556 [Vani有约会] 雨天的尾巴 /【模板】线段树合并

给定一个n个节点的树，每个节点上都有一个可重集，m次操作均为向树上一路径(u, v)上的所有可重集同时添加一个数c，所有操作后询问每个可重集中个数最多的数。

$n, m \le 10^5, 1 \le c \le 10^5$

- 考虑链上的情况，我们可以将修改差分，维护一个权值线段树，从左到右进行加减操作，同时查询全局最大值对应id。

- 考虑树上的情况，我们如法炮制，进行树上差分。根据树上差分的做法，我们应当从每个叶子向上跳，并在两两的lca处合并信息。这里合并权值线段树的信息便需要线段树合并。

- 考虑我们现在要将a,b两棵动态开点线段树进行合并，目前正在合并u,v节点及其子树

  1. 若u,v都为叶子，则直接合并信息并返回。

  2. 若u,v左子树都非空，则递归进二者左子树继续合并。

  3. 若u,v左子树有某一个非空，则将其直接作为合并后的左子树。

  4. 若u,v左子树均为空，则合并后左子树也为空。

  右子树同理

- 通过该合并方式，任意棵共有n个叶子的动态开点权值线段树合并为一棵的时间复杂度为$O(n\log k)$（k为值域大小）

- 这个过程的正确性很显然，我们来验证时间复杂度

- 考虑这个过程中，我们的每次合并实际上是访问了这两棵线段树所有共有的节点，考虑新加入一个树有t个叶子（$\sum t = n$），则其至多有 $t\log k$ 个节点，则这次合并的复杂度$O(t\log k)$，总复杂度 $O(n\log k)$

P6773 [NOI2020] 命运

给定一棵树 $T=(V,E)$ 和点对集合 $Q\subseteq V\times V$，满足对于所有 $(u,v)\in Q$，都有 $u\neq v$，并且 u 是 v 在树 T 上的祖先。其中 V 和 E 分别代表树 T 的结点集和边集。

求有多少个不同的函数 $f：E\to\{0,1\}$（将每条边 $e\in E$ 的 $f(e)$ 值置为 0 或 1），满足对于任何 $(u,v)\in Q$，都存在 u 到 v 路径上的一条边 e 使得 $f(e)=1$。由于答案可能非常大，你只需要输出结果对 998,244,353（一个素数）取模的结果。

$1\leq |V|,|Q|\leq 5\times 10^5$

- 首先考虑若 (u_i, v_i) 包含 (u_j, v_j)，则其限制作用严格弱于后者，可被删除。去除后所有限制互不包含。

- 感性的理解，各个限制从树的枝叶向上延伸，之后汇聚到主干。那我们将限制分两部分考虑，一部分是其分散在子树内（这部分我们总是通过将问题归纳进子树就解决了），一部分是其从某节点开始共同延伸向祖先。

- 注意到，对于从同一个节点延伸向上的没有被满足的限制中，只有u_i深度最深的那个限制有效。由此我们考虑设出状态 $f_{u,d}$，表示u子树内没被满足的限制中的最深的深度。

- 将v的信息向父亲u合并时，分别考虑$f\big((u,v)\big)=0\ or\ 1$，有$f'_{u,d}=f_{u,d}*\sum_{i=0}^{d}f_{v,i}+f_{v,d}\sum_{i=0}^{d-1}f_{u,i}+\ f_{u,d}*\sum_{i=0}^{deep_u}f_{v,i}$，可以得到$n^2$的做法。
- 观察转移的过程，当且仅当$f_{u,d}$, $f_{v,d}$不同时为0，$f'_{u,d}$才不为0。而由于一共只有m条限制，最开始一共也只会存在m个不为0的$f_{v,d}$。这强烈启发我们去思考是否能通过线段树合并的方式优化。
- 首先对d建线段树T_u存储$f_{u,d}$与其区间和，考虑对于某次合并过程。$\sum_{i=0}^{deep_u}f_{v,i}$其实是一定值，将其记为$p$。考虑线段树合并的过程，我们优先处理完左子树，这样可以得出$\sum_{i=0}^{d}f_{v,i}$与$\sum_{i=0}^{d-1}f_{u,i}$。接着，若T_v该子树为空，则直接将T_u该子树区间乘$\sum_{i=0}^{d}f_{v,i}+p$。若T_u该子树为空，则将T_v该子树区间乘$\sum_{i=0}^{d-1}f_{u,i}$。否则递归进左右子树计算。该时间复杂度分析同线段树合并一致。

K-D Tree

P4148 简单题

你有一个$N \times N$的棋盘，每个格子内有一个整数，初始时的时候全部为 0，现在需要维护两种操作：

- $1\ x\ y\ A$：将x, y里的数字加上A
- $2\ x_1\ y_1\ x_2\ y_2$：求x_1, y_1, x_2, y_2这个矩形内的数字和

强制在线

$1 \le N \le 5 \times 10^5$，操作数不超过$2 \times 10^5$，内存限制20MB

- K-D Tree 的是一种用于存储k维空间信息的二叉树，其每一个节点代表空间内一点，每个节点的左右子树都是两个不相交的k维空间。

- 建树过程中，我们每次轮流选择一个维度进行分割，将对应这个维度的中位数的点放在该节点，按剩余点该维是否大于中位数将其分入左右子树。

- 这种建树方式保证了树高是log级别，同时在二维查询一个矩形的时间复杂度为$O(\sqrt{n})$。

- 这题要求强制在线，无法使用CDQ分治，同时内存限制使得无法使用树套树，考虑使用K-D Tree。

- 查询矩形和。直接对于当前节点，看左右子矩形与查询矩形的关系，若部分相交则递归进子树继续查询。

- 修改。由于K-D Tree是不可旋转的，添加节点导致不平衡只能通过重构的方式解决。

P2093 [国家集训队] JZPFAR

给定 n 个点在二维平面上，有 m 次询问，每次询问给定一点 (p_x, p_y) 与一整数 k，输出 n 个点中离 (px, py) 的距离第 k 大的点的标号。

$1 \le n \le 10^5$, $1 \le m \le 10^4$, $1 \le k \le 20$, $-10^9 \le x_i, y_i \le 10^9$, 询问**点坐标在某范围内随机分布**

- K-D Tree查询最近（最远）点需要基于随机性，有时可能可用于骗分。考虑对于二维矩形，一点到矩形内的点最小（最大）距离，可用矩形的边框来确定范围。（最小距离大于等于查询点到矩形边的距离，最大距离小于等于查询点到矩形顶点的距离），如果该矩形内可能存在比当前答案更优的点，我们就暴力递归进该矩形查询。

- 对于本题，第k大的k其实很小，考虑直接维护一个大小为k的小根堆，一开始里面全是-INF，若左右子矩形可能的最大距离大于堆顶则递归进去搜索。同时在节点检查该节点的距离是否大于堆顶，若是则弹出堆顶并将当前距离入堆。

P5471 [NOI2019] 弹跳

给定平面内n点以及m组边，每组边为$p_i, t_i, L_i, R_i, D_i, U_i$，表示由$p_i$向$L_i, R_i, D_i, U_i$范围内所有点连一条长为$t_i$的边。求从1号点到其余点的最短路

$n \le 7 \times 10^4, m \le 1.5 \times 10^5$ 内存 128MB

- 考虑类似线段树优化建图，这题我们可以采用K-D Tree优化建图。若是直接建图，由K-D Tree的矩形查询复杂度可知每组边都需要连$\sqrt{n}$条边，总共$m\sqrt{n}$条边，总复杂度$O(m\sqrt{n}\log n)$，会被卡掉

- 考虑堆优化*Dijkstra*的过程，松弛过程其实本质是子树对某数取min，接着就是找全局最小值点继续该过程。我们完全可以不把图显式的建出来，直接利用**K-D Tree**优化*Dijkstra*。

- 我们在节点记录该点最短路，子树内最短路及对应id，同时可以类比线段树在K-D Tree多维护一个tag，表示子树应对tag取min，在查询或修改的时候push down即可。
- 注意一个点被visit后应当在之后被忽略。

线段树分治&二进制分组

P5227 [AHOI2013] 连通图

给定一个无向连通图(V,E)和k个小集合S_i，每个小集合包含一些边，对于每个集合，你需要确定将集合中的边删掉后改图是否保持联通。集合间的询问相互独立

定义一个图为联通的当且仅当对于任意的两个顶点，都存在一条路径连接它们

$1 \le |V|, k \le 10^5$

$1 \le |E| \le 2 \times 10^5$

$1 \le |S_i| \le 4$

- 线段树分治的基本思路是当有一些问题中，删除操作难以处理，但撤销操作可以通过某些可接受的复杂度得到时，我们通过将操作离线，把每个操作视作在一段时间轴上起作用。我们在时间轴上建立线段树，类比线段树的区间加，我们可以将每个操作分为至多分为log块。类似打tag，我们在线段树的每个节点开一个vector用于存储应进行的操作。这样我们直接遍历整棵树，在进入节点时做对应操作，离开节点时撤销这些操作，在每个叶子便是对应时刻的状态。

- 对于这题，维护带删除的图连通性非常困难，但我们可以维护可撤销并查集。这样把每条边视作在一定时间范围上的加边操作，使用线段树分治即可。

CF710F String Set Queries

维护一个字符串集合，支持三种操作：

1.加字符串

2.删字符串

3.查询集合中的所有字符串在给出的模板串中出现的次数

操作数 $m \leq 3 \times 10^5$，输入字符串总长度 $\sum |s_i| \leq 3 \times 10^5$。

本题强制在线

- 二进制分组主要是用于一些静态的数据结构，由于题目要求支持动态修改，对于现有的n条数据，我们按其二进制将其分为多组二的幂次大小的数据，比如 $6=(110)_2$ 那么我们就分为一组规模为4，另一组为2。在每组内部独立地建立数据结构维护信息。当加入一条新信息时，类似2048的过程，我们从规模小的部分一路合并，如果存在规模为1的组，就合成一个规模为2，如果同时有规模为2的数据组，就合成规模为4的，以此类推。这样做正确性显然，考虑时间复杂度。无论如何，每个数据最多被合并log次，设构建一个包含n个数据需要$O(nq)$，那么使用二进制分组的复杂度为$O(nq\log n)$
- 考虑本题，如果没有修改，我们只需要建立ac自动机与fail树，在fail树上查询即可。（参考 P5357 【模板】AC 自动机）。但当我们需要修改时，fail树无法动态维护，使用二进制分组即可。这里删除操作可以用两组ac自动机维护，一组维护添加操作，一组维护删除。每次查询时用在前者查出的答案减去在后者查出的答案即可。