第二轮讲课

luckyleaves

绵阳乐荟城附属一中

2022 年 11 月 9 日

目录

目录

目录

众所周知，数组查询方便，但插入和删除困难（复杂度极高 $O(n)$ ）而链表恰恰相反，索引困难，而插入和删除简单。

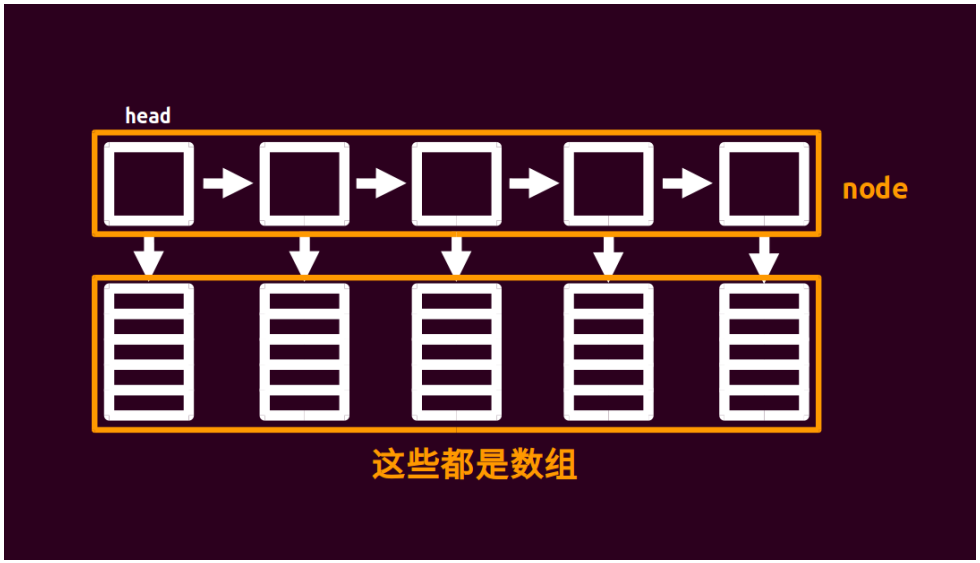
那么当我们遇到既要动态插入删除，又要快速索引元素的问题时，有没有一种数据结构能够同时快速支持这两个操作呢?

众所周知，数组查询方便，但插入和删除困难（复杂度极高 $O(n)$ ）而链表恰恰相反，索引困难，而插入和删除简单。

那么当我们遇到既要动态插入删除，又要快速索引元素的问题时，有没有一种数据结构能够同时快速支持这两个操作呢？

以我们老祖宗的强大的智慧，答案当然是有，它就是——~~序列之王 Splay~~块状链表。

听到这个名字，大概都能猜到它的主要思想：分块。
它将整个数列分成 $\sqrt{n}$ 块，每一块 $\sqrt{n}$ 个数用链表串起来，然后再用一个主链表将他们都传起来，具体的它长这样：



用 vector 实现

1. 建立块状链表。依次提取数组的前 B 元素当一个 $vector$ 添加。
2. 找到第 k 元素。记录每 vector 维护的范围，确定 k 所在区间，用 $k-l$ 即可找到（l 是这个 $vector$ 的首项）。
3. 插入。首先，我们需要知道插入的位置。为了方便默认是在给定插入位置的前面插入；直接用 $vector::insert$。为了保证时间复杂度，如果插入完导致这个 $vector$ 大于 $2B$，则在 B 处分裂这个 $vector$。具体怎么分裂：
   1. 在块状链表里找到这个 $vector$ 接下来的 $vector$ 在接下来的 $vector$ 之前插入一个新建的 $vector$。
   2. 这个新建 $vector$ 应该保存刚才插入的 $vector$ 的 B 元素后缀，也就是 $vector::end()-B$ 这个 $iterator$ 区间。我们用 B 分裂来平均分裂后的 $vector$ 长度，并同时保证没有 $vector$ 长度超过 B。
   3. 建完新的 $vector$，再删除这个区间。
4. 删除。同样找到位置，利用 $vector::erase$ 如果这个 $vector$ 的大小变为零，把这个 $vector$ 从块状链表删除。

B 取 $\sqrt{n}$

关于主链:

其实主链没有必要用 $vector$ 强行插入，在每一块中间记录本块左右块的编号，然后直接在队尾加即可（常数小的一批，在下面的题中表现非常优秀）

例题

带插入区间 K 小值

这道题全凭各位常数。

过不了就删掉 $split$ 函数。

前言

分块之神 Dyd 曾经说过：分块就是一种套路，你把代码每一部分换成暴力，调出一部分就全部调出来了，所以分块特别好调。

蒟蒻不懂，可能这就是蒟蒻比巨佬菜的原因。

特殊分块

例题 1

Points on Plane

一句话题意：给定 n 个点对，要求对这 n 点排序，使每个点（第一个点除外）到前一个点的曼哈顿距离之和小于等于 $2.5 \cdot 10^9$。

其中 $n \leq 10^6$，每个点的横纵坐标均不超过 n 的最大范围。

Solution

提示 1：我们在讲分块，考虑将点分成几个部分试试？

Solution

提示 1：我们在讲分块，考虑将点分成几个部分试试？

将点对视作一维区间，将相邻两点的距离改转换成区间的移动距离，要使区间移动次数尽可能小，你想到了什么？

Solution

提示 1：我们在讲分块，考虑将点分成几个部分试试？

将点对视作一维区间，将相邻两点的距离改转换成区间的移动距离，要使区间移动次数尽可能小，你想到了什么？

这就是莫队的常规操作，但是交上去却发现无论怎么改变块长最后答案都比 $2.5 \cdot 10^9$ 大一点，此时还有什么优化空间？

Solution

提示 1：我们在讲分块，考虑将点分成几个部分试试？

将点对视作一维区间，将相邻两点的距离改转换成区间的移动距离，要使区间移动次数尽可能小，你想到了什么？

这就是莫队的常规操作，但是交上去却发现无论怎么改变块长最后答案都比 $2.5 \cdot 10^9$ 大一点，此时还有什么优化空间？

发现莫队跨块时总会有无效的移动，我们考虑其经典优化，对于奇数块，我们让其右端点单增，偶数块反之，则可通过这道题。

例题 2

问 1：
分块可以视作三层的 $\sqrt{n}$ 叉树，如何做到 $O(1)$ 插入 $O(\sqrt{n})$ 查询区间 k 大数。

例题 2

问 1：
分块可以视作三层的 $\sqrt{n}$ 叉树，如何做到 $O(1)$ 插入 $O(\sqrt{n})$ 查询区间 k 大数。
值域分块，维护块内数的个数和。查询时先逐块确定，然后进入块内逐个确定。

例题 2

问 1：
分块可以视作三层的 $\sqrt{n}$ 叉树，如何做到 $O(1)$ 插入 $O(\sqrt{n})$ 查询区间 k 大数。
值域分块，维护块内数的个数和。查询时先逐块确定，然后进入块内逐个确定。

问 2：
如何做到 $O(\sqrt{n})$ 插入，$O(1)$ 查询第 k 小（全局）。

例题 2

问 1：
分块可以视作三层的 $\sqrt{n}$ 叉树，如何做到 $O(1)$ 插入 $O(\sqrt{n})$ 查询区间 k 大数。
值域分块，维护块内数的个数和。查询时先逐块确定，然后进入块内逐个确定。

问 2：
如何做到 $O(\sqrt{n})$ 插入，$O(1)$ 查询第 k 小（全局）。
考虑对每个 k 维护出它在哪个块中，我们对此再分块（后文为区别称其为段），段数是 $\sqrt{n}$ 的，那么单次插入对于所有段的改变是 $\sqrt{n}$ 级的，然后我们考虑对原来每个值域块维护出前面数的个数（插入的改变肯定也是 $\sqrt{n}$ 的），直接将 k 减去后取对应下标就行了。

例题 3

给定一个长度为 n 的序列和 m 个区间。
有两种操作:

1. 将序列中某个数 x 改为 y。
2. 查询第 x 个区间到第 y 个区间和（的和）。

Solution

提示 1：和上面的问题解法有部分重叠。

Solution

提示 1：和上面的问题解法有部分重叠。

首先将区间分块，维护 $cnt[len][i]$ 表示 i 在第 len 块中覆盖了多少次，这样单点修改了之后就可以 $O(\sqrt{n})$ 求出每块的变化，对于散块的维护，发现每次查询时需要进行 $\sqrt{n}$ 次区间求和，使用 $O(\sqrt{n})$ 修改（用分块维护原序列的前缀和，单次询问区间和就是 $O(1)$），$O(1)$ 查询的分块即可。

带修改莫队

带修改莫队

带修改莫队

看了一下，大家都会。

但还是再唠叨一下。

总所周知，普通的莫队是不能支持修改操作的，但是有些题目它就是让你支持修改，而且除了莫队你还不会其他的算法，这个时候我们就需要对莫队进行改进，让其支持修改。

我们考虑对每个询问记录三个信息：$[l, r, time]$，意味着查询的起始，终点，需要完成前 $time$ 次修改操作。

根据分块的时间复杂度平衡思想，首先考虑将序列分成 len 块，考虑将 l，r 都放入块中，首先按 l 的块的编号排序，然后 l 的块的序号相同的询问再按 r 的块的编号排序，l 和 r 的块的编号都相等的再按需要修改的次数升序排序。

考虑这样做的时间复杂度，一共有 len 块，令每块长度为 $limit$（则 n 可以粗略的认为是 $len \times limit$），发现莫队的移动变成了三个部分：l 块移动，r 块移动，t 指针移动。

首先来考虑第一部分，l 块移动时（跨块时），r 移动的时间复杂度是 $\frac{n^2}{limit}$。

第二部分，r 移动时（块内移动），l 的移动的时间复杂度为 $n \cdot limit$ 的。

第三部分，l, r 移动前，$time$ 指针移动（块内），时间复杂度为 $\frac{n^2 \cdot m}{limit^2}$。（$m$ 是中间的修改操作，两两块匹配时 $time$ 指针都要移动 m 次）

将三部分的极值取 min，$limit$ 就应该取 $\sqrt[3]{n \cdot m}$，最后复杂度就是 $\sqrt[3]{n^4 \cdot m}$，当然因为平时 cnt 卡不满，所以可以看着定。

例题

P1903 国家集训队数颜色 / 维护队列

例题 2

CF940F Machine Learning

一句话题意：

给定一个序列 a，有两种操作，共有 m 次。
操作 1：查询区间 $[l, r]$ 中每个数出现次数的 mex。
操作 2：单点修改
n, m 都是 10^5

Solution

没有要求在线，考虑离线，用带修改莫队维护出当前时间戳下每个数出现的次数和每种出现次数是否都有数满足出现了这么多次。

之后我们考虑区间 mex 怎么求，考虑我们暴力做的话，最坏情况也就是每种次数都有数满足，那么区间含有的数的个数就需要达到：$1+2+3+4\cdots$，不难发现这个长度最多是 $\sqrt{n}$ 的，所以暴力求 mex 没有问题。

树分块

树分块

一般可以用其他做法碾压树分块，所以题目较少。

树上数数的不带修的强制在线的经典问题

特别的，菊花图，蒲公英可以卡掉部分题目中的树分块解法，但本人也不是很懂怎么卡。

什么是树分块

树分块就是像序列分块一样设置一个阈值 B，然后在树上随机撒 $\frac{n}{B}$ 个点，使其成为关键节点，然后使每一个点到距离其最近的关键节点的距离不超过 B，这样我们就可以首先预处理出每条到根节点的链上两两关键节点之间的信息（所以树分块支持不了大量的修改操作），每次将需要计算的两点的距离最近的关键节点的信息拿出来，其他信息暴力统计即可，复杂度：$O(B^2+qB+\frac{n^2}{B})$。

撒点

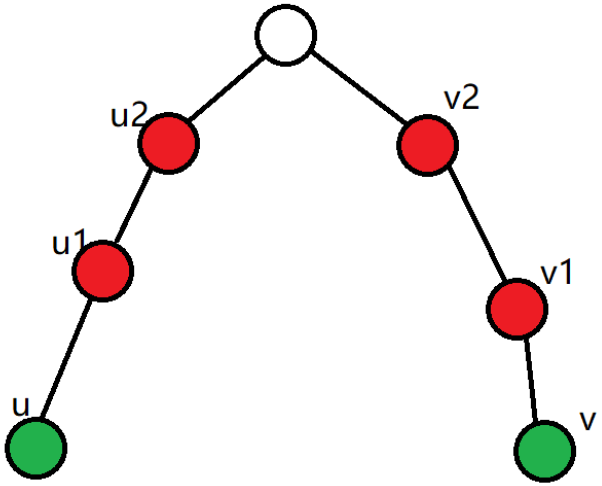
关于树分块的撒点，首先我们将所有的非关键节点入队，每次取出其中深度最大的节点，若他的节点的 $1 \sim B$ 级祖先都没有关键节点，就将它的 S 级祖先标记为关键节点，由上述方法可知每一次标记至少能控制 B 个节点，这些节点都不会被标记，且从这个节点出发最大移动距离最多为 B，满足了我们的需要。

维护信息

接下来，考虑对一条到根的路径上的所有关键点用 $bitset$ 维护出他们两两之间的颜色，处理的时候我们可以使用递推的方式即：$b_{i,j} = b_{i,j-1} or b_{i+1,j-1}$，所以我们可以用最多 $O(n)$ 的时间统计出每相邻关键节点的答案，其他的答案再两两统计，由于点对最多有 $\frac{n^2}{B^2}$，所以预处理的复杂度为 $O(\frac{n^2}{B} + \frac{n^3}{wB^2}) \approx O(n\sqrt{n} + \frac{n^2}{w})$。

计算答案

之后就简单了，将答案拆成四部分即可：



其中红色的节点表示关键节点，绿色节点表示查询节点，查询节点到最深的关键节点暴力做，最高的关键节点之间也暴力做，那么最终复杂度为：$O((n+m)\sqrt{n}+\frac{n^2+nm}{w})$。~~(不知到为什么被 SPOJ 卡掉了)~~

例题

P6177 Count on a tree II/[模板] 树分块

P3603 雪辉

树上莫队

简介

我们回顾一下树分块的那道题，我们发现这样的做法及其丑陋(但支持强制在线这一点就很强)，我们想一想如果不是强制在线的话我们应该怎么搞，考虑到一般莫队只能处理序列问题，思考一棵树如何转成序列，我们使用一个叫欧拉序的东西，具体构造方法:

欧拉序

对这棵树进行 dfs，将任何第一次访问的节点加入序列，当节点回溯时再将这个节点加入序列。

我们回过头来重新观察这道题，设 i 节点在序列中第一次出现是 $st[i]$，第二次出现是 $ed[i]$，那么对于对于任意询问节点 x，y，设 $st[x] < st[y]$，分类讨论可以得到。

1. $lca(x, y) = x$，画图可以得到答案就是序列中 $ed[x] \sim st[y]$ 之间所有只出现过一次的数所包含的不同种类（所以第一次出现就增加第二次出现就减少）。
2. $lca(x, y) \neq x$，不难发现此时 y 不可能是 x 的父亲 （不然的话为什么 st[y] 不在 st[x] 前面） ，所以画个图分析，此时的 $ed[x] \sim st[y]$ 没有包含 $lca(x, y)$，特判即可。

拓展题

还有一道和这个题一摸一样的拓展版（多了修改，~~但无伤大雅~~）：
[WC2013] 糖果公园，快去切掉吧。

Count on a tree

COT - Count on a tree

Count on a tree

比较板，但是题解很有意思的一道题。
题解说到这样一段话：
当数据结构之间互相兼容时，往往会出现以下几种情况。

1. 根号套根号，复杂度往往也是根号的（值域分块套区间分块）。
2. log 套 log，复杂度往往是多个 log 相乘的。
3. 根号套 log，复杂度往往是 $\log n \cdot \sqrt{n}$ 或者 $\sqrt{n \log n}$ 的。

Count on a tree

首先我们先按照树上莫队的思路将整颗树转化成一个序列，然后我们发现问题就转化成一个区间 k 大值问题（记得对于 lca 要特判）。

比较显然的
思路是用莫队，然后于此同时维护一颗平衡树，复杂度 $m\sqrt{n}\log n$。

Count on a tree

然后就会发现打死也过不去，根据前面的规律，我们希望能够再平衡一下复杂度，使其降到 $\sqrt{n}$。

首先莫队复杂度就是 $m\sqrt{n}$ 的了，所以我们剩下的这个就应该做到 $O(1)$ 插入。

发现值域分块可以做到，而且查询也是 $\sqrt{n}$ 的，非常平衡。

Count on a tree

以下的东西就扯远了，但是因为有双倍经验，但是那道双倍经验要求强制在线，所以我们把另一种做法也补充一下。

考虑树上差分 $+$ 主席树。发现主席树统计的形式是：$val_r - val_{l-1}$，不难发现这里的贡献形式为：$val_u + val_v - val_{lca(u,v)} - valfa_{lca(u,v)}$，然后就可以做了。

基本就是个板子题（但是可以卡空间）。

”在线莫队”

说是说莫队，但实际上主题是序列分块。

当一个题就是个莫队板子但它却要求你强制在线的时候（且空间足够），我们一般这样思考（有更优的做法当我没说）。
考虑回滚莫队的过程，当我们从前一个询问的区间向当前区间移动时，我们发现这过程就是将我们已经算出来的区间扩大成更大的区间，我们尝试在这个数列上撒很多关键点（设有 B 个），预处理出两两关键点的信息并将其储存起来（这部分时间复杂度是 $n\sqrt{B^2}$ 的）。

之后我们想要查询一段区间就可以找到它内部最大的已经被计算过的区间，然后通过已经算出来的信息与剩下散块的信息暴力合并即可，不难发现此时的复杂度为 mB 的。

再根号平衡一手，复杂度就是 $(n+m)\sqrt{n}$ 的了。

个人没有发现如何规避没有包含任何已经处理的区间的处理办法，只能将其特判掉，如果大家有比较好的办法可以上来分享一下。

例题

P4135 作诗

例题 2

作诗加强版，现在卡空间了，空间限制：$64MB$，请各位神仙各显神通。

Solution

提示 1：维护增量

Solution

提示 1：维护增量
提示 2：使用 $bitset$，但是怎么判断这个数是否曾经出现过?

Solution

提示 1：维护增量
提示 2：使用 $bitset$，但是怎么判断这个数是否曾经出现过?
对于每种数建立一个 $vector$，同时记录当前数在 $vector$ 中的位置，计算的时候查一下前驱后继即可，时间复杂度 $n\sqrt{n}$，空间复杂度：$\frac{n\sqrt{n}}{w}$。

例题 3

P5048 [Ynoi2019 模拟赛] Yuno loves sqrt technology III

例题 3

P5048 [Ynoi2019 模拟赛] Yuno loves sqrt technology III

跟上一道是一个套路。

Solution

首先我们先用 $n\sqrt{n}$ 的时间维护出两两块的答案。

然后我们考虑散块对答案的贡献，依旧考虑在线莫队，发现卡空间，我们依旧考虑维护增量，先求出每个整块的答案，然后考虑散块中每一个值 a_i，如果它在左散块，就判断其右边且在范围内是否存在等于 ans 个这种数，如果有，就更新 ans 对此的，由于我们已经把相同数都加入一个 $vector$，所以我们直接在这个 $vector$ 上对下标直接做加法即可，右散块同理，时间复杂度 $n\sqrt{n}$，空间复杂度 n。

倍增分块 + 底层线段树 + 底层分块

前言

~~怀疑赛事怎么可能有人写出来~~

$5e5$ 的神仙分块。

题目

一个数列，实现如下两种操作。
1. 将 [l,r] 区间中所有 >x 的数减去 x。
2. 询问区间 [l,r] 的和，最小值，最大值。
值域 1e9，序列长度 5e5

一看这题的第二问，哈，我会，线段树。

一看这题的第二问，哈，我会，线段树。
冷静下来看第一问，*?!@!?><。

一看这题的第二问，哈，我会，线段树。
冷静下来看第一问，*?!@!?><。
毕竟是 lxl 的题，我们首先得往分块上想。

仔细思考之后就会发现这题必须要用数据结构维护值域和序列。

听说这题可以用分块套 Splay（不愧是序列之王）
~~只要你没有常数~~。

std

因为值域 10^9，考虑一种神奇的分块：倍增分块，这是一种什么样的分块呢，就是将值在 $[B^i, B^{i+1})$ 的数分在一起。

思考为什么考虑这种分块：

std

因为值域 10^9，考虑一种神奇的分块：倍增分块，这是一种什么样的分块呢，就是将值在 $[B^i, B^{i+1})$ 的数分在一起。

思考为什么考虑这种分块：

我们使用线段树暴力的时候，一定会先建一课序列线段树。

std

因为值域 10^9，考虑一种神奇的分块：倍增分块，这是一种什么样的分块呢，就是将值在 $[B^i, B^{i+1})$ 的数分在一起。

思考为什么考虑这种分块：

我们使用线段树暴力的时候，一定会先建一课序列线段树。

1. 如果区间的最大值 $\leq x$ 直接跳过。
2. 如果区间的最小值 $> x$ 打上懒标记之后返回。
3. 递归进入子区间。

std

因为值域 10^9，考虑一种神奇的分块：倍增分块，这是一种什么样的分块呢，就是将值在 $[B^i, B^{i+1})$ 的数分在一起。

思考为什么考虑这种分块：

我们使用线段树暴力的时候，一定会先建一课序列线段树。

1. 如果区间的最大值 $\leq x$ 直接跳过。
2. 如果区间的最小值 $> x$ 打上懒标记之后返回。
3. 递归进入子区间。

仔细想一想，如果序列是一大一小交替排布，每一次更改就是大常数的 $O(n)$，跑不过暴力。

std

我们假设已经将值域分成了 cnt 个块，每个块里都有一个序列线段树。

这样整体上看最坏复杂度为 $n \times cnt \times \log n$

std

我们假设已经将值域分成了 cnt 个块，每个块里都有一个序列线段树。

这样整体上看最坏复杂度为 $n \times cnt \times \log n$

但是思考分块的作用。

std

我们假设已经将值域分成了 cnt 个块，每个块里都有一个序列线段树。

这样整体上看最坏复杂度为 $n \times cnt \times \log n$

但是思考分块的作用。

1. 如果一个块上界 $\leq x$ 直接跳过。
2. 如果一个块的下届 $> x$，这一部分将被重构，那么这一部分复杂度是：$m \times cnt \times \log n$ 的。
3. 不然就递归。

std

我们假设已经将值域分成了 cnt 个块，每个块里都有一个序列线段树。

这样整体上看最坏复杂度为 $n \times cnt \times \log n$

但是思考分块的作用。

1. 如果一个块上界 $\leq x$ 直接跳过。
2. 如果一个块的下届 $> x$，这一部分将被重构，那么这一部分复杂度是：$m \times cnt \times \log n$ 的。
3. 不然就递归。

我们发现这样做了之后前两种情况的复杂度之和块的数量有关，但是，第三种情况的最差复杂度是线性的。

std

我们假设情况三的块长为 len，由上我们发现每次修改最多会有一块成为情况三，所以最多就会操作 $cnt \times \frac{len}{x}$ 次，我们将这式子拆开分析，首先如果每个块都已经因为上面的操作变成同一个值域了的话，显然最多会有 cnt 块需要递归，然后每一块内我们最多递归 $\frac{len}{x}$ 次（之后就减为 1 了）。

考虑最叼钻的情况，即 $\forall x = l$ 的情况，l 就是当前块的左边界，我们就要让 $n \times \log n \times cnt \times \frac{len}{l}$ 最小，考虑固定 $\frac{len}{l}$，$cnt = \log_B^{10^9}$，最终的复杂度：

$$O((n+m) \times \log_B 10^9 \times \log n + n \times B \times \log n \times \log_B^{10^9})$$

std

~~打开几何画板，但是并没有画出来~~，B 的理论最优为 16，实测的话：应该取 $2 \sim 4$。

然后你就会发现卡空间啦（~~伞兵出题人，这都要卡，你!>!<?~~），线段树的空间主要消耗在叶子，我们可以设置一个常数 K，让线段树 $\leq K$ 的区间的处理直接变成下传标记在序列上暴力，之后再上传。我们可以这样很可观地以大常数的时间代价换取空间(~~这就是传说中底层分块~~)。

例题

rgxsxrs

分块就暂时告一段落了，接下来就是线性代数。

定义

令 A 是一个矩阵，如果存在一个矩阵 B 满足：$BA = I$

I 表示只有对角线是 1 其余都是 0 的矩阵。

则称 A 矩阵可逆，且 B 称为 A 的逆矩阵。

前置知识

一堆证明
矩阵的逆要么没有要么唯一
1. 首先要知道若 A 可逆，则逆唯一。
证明：若 B 和 C 都是 A 的逆，由定义：$AB = BA = I, AC = CA = I$
则：$B = IB = (AC)B = (AB)C = C$，即 $B = C$，证毕。
2. 矩阵具有结合律。

代数余子式和余子式：
余子式：设 $|A|$ 是一个 n 阶的行列式，划去第 i 行以及第 j 列，剩下的 $(n-1)^2$ 个元素按照原来的顺序组成了一个发 $n-1$ 阶行列式，这个行列式称为 $|A|$ 的第 (i,j) 元素的余子式，称为 $M_{i,j}$

代数余子式：设 $|A|$ 是一个 n 阶行列式，$M_{i,j}$ 是第 (i,j) 元素的余子式，定义 $|A|$ 的第 (i,j) 元素的代数余子式为：$A_y = (-1)^{i+j} det(M_{i,j})$

行列式判断法

设 n 阶矩阵：

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,n} \end{bmatrix}$$

构造如下方阵（其中 $A_{i,j}$ 是 A 关于 i,j 的代数余子式）：

$$A^* = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} & \cdots & A_{1,n} \\ A_{2,1} & A_{2,2} & A_{2,3} & \cdots & A_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & A_{n,3} & \cdots & A_{n,n} \end{bmatrix}$$

行列式判断法

称 A^* 为 A 的伴随矩阵，而 A 可逆的充要条件为 $|A| \neq 0$，且有性质 $A^{-1} = \frac{A^*}{|A|}$。

证明略过~~(看了许久，啥也看不懂)~~

下文有更简单的判定和计算方法。

逆变换有一重要性质：若 A,B 均可逆，则 AB 也可逆，且 $(AB)^{-1} = B^{-1}A^{-1}$。

该性质可以反复利用，因此可以拓展到 k 个可逆行列式相乘。

初等矩阵

将单位矩阵进行一次初等行变换，就得到了初等矩阵。其中，初等矩阵行变换有以下三种：

1. 交换矩阵的任意两行。
2. 用一个非零整数 k 乘矩阵的任何一行。
3. 矩阵中某一行加上另外一行的 k 倍。

初等矩阵

由于初等行变换可逆，所以初等矩阵可逆。

证明：充分性：设 E 为一初等矩阵，由于 $EI = E$，因此任意一个初等矩阵可以视为对 I 矩阵的一种变换，使其变为 E。

必要性：由于初等行变换可逆，则存在 E 变换的逆变换 F，将 E 矩阵变回 I，因此 $EF = I$，即 E 可逆，且其逆为 E 的逆变换。

初等矩阵

结论：方阵 A 可逆，当且仅当 A 行等价于 I_n，即 A 经过若干次行变换可以变成 I_n。
我们把 A 和 I_n 置于同一矩阵中：

$$\begin{bmatrix} A & I_n \end{bmatrix}$$

对其进行高斯-若尔当消元操作将 A 变换为 I_n。由于是在同一矩阵中，因此 A 和 I_n 得到的操作都是一样的。我们只需要让前面的 A 变换成 I_n，那么对于同一过程，I_n 就会变成矩阵的逆。
这个转化关系如下图所示：

$$A \underset{P}{\overset{P'}{\Longleftrightarrow}} I_n \underset{P}{\overset{P'}{\Longleftrightarrow}} A^{-1}$$

初等矩阵

还不懂的，可以看看这个网站

起源

行列式一开始是为了解方程而发明的东西（毕竟手玩 n 阶方程的高斯消元不是人能完成的）。

定义

行列式是一个函数定义，取值是一个标量。

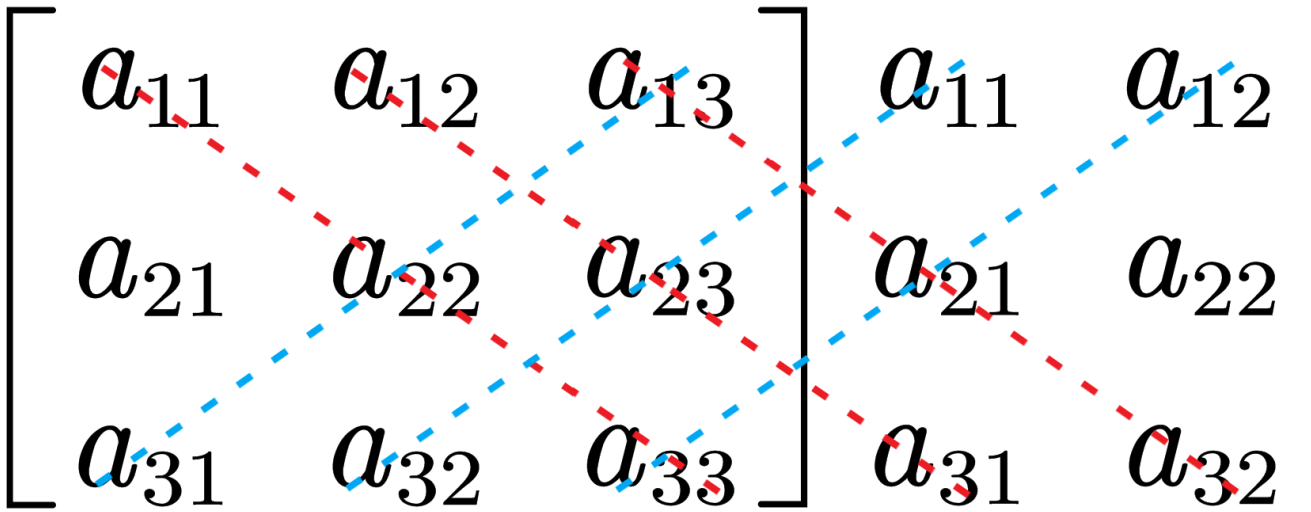
对一个 $n \times n$ 的矩阵 A，其行列式一般记作 $det(A)$（行列式：determinant）或 $|A|$，其定义为：

$$det(A) = |A| = \sum_{p} (-1)^{\Gamma(p)} \prod_{i=1}^{n} a_{i,p_i}$$

p 表示一个排列，所有可能的 p 则是 1 到 n 这 n 个数的全排列。$\Gamma(p)$ 表示一个排列 p 的逆序对个数。

计算方法：对角线计算法

首先将第一列和第二列平移到矩阵的右侧，对角线上的元素相乘，同色对角线计算后求和，用红色的减去蓝色的就是答案。如图：



注意，这只适用于 $2,3$ 阶的行列式，对于更高阶的规矩还未找到~~（所以屁用没有）~~。

计算方法：代数余子式法

说白了就是一行一行的展开。

下面我们以三阶行列式为例:

$$D_3 = \begin{vmatrix} a_{1,1} & a_{1,2} & a1,3 \\ a_{2,1} & a_{2,2} & a2,3 \\ a_{3,1} & a_{3,2} & a3,3 \end{vmatrix}$$

这时候我们按照代数余子式的定义展开第一行:

$$D_3 = (-1)^{1+1} a_{1,1} M_{1,1} + (-1)^{1+2} a_{1,2} M_{1,2} + (-1)^{1+3} a_{1,3} M_{1,3}$$

递归去做即可。

计算方法：等价转化法

性质 1：行列式的某一行（列）的各元素乘同一数然后加到另一行（列）对应的元素上去，行列式不变。
性质 2：行列式中某一行（列）的所有元素的公因子可以提到行列式记号的外面。

转化法的核心思想是将行列式转化成上三角行列式。（类似于高斯消元）。
最后的行列式也就是对角线上的所有数的乘积。

还有一种没有任何优化而且使用起来更复杂的逆序数法，其是行列式求解最基础的方法，也是前三种方法的前身，所以更 garbage（~~不然为什么叫前身~~）。

由于平时是需要取模的，在精度的要求下，可以使用辗转相除法减小

行列式的一些相关结论

1. 如果矩阵的两行（或列）互换位置，则行列式数值保持不变，但符号改变。

2. 若矩阵的某行（或列）是其他行（或列）的线性组合，则 $det(A) = 0$。

3. 单位矩阵的行列式为 1。

4. 任何一个正方矩阵 A 和它的转置矩阵 A_T (把矩阵的行列互换之后得到的矩阵) 具有相同的行列式（证明可以考虑将 A 化为上三角矩阵，A_T 化为下三角矩阵，其对角线上的值显然没有变）

行列式的一些相关结论

5. 两个矩阵乘积的行列式等于它们的行列式的乘积。

6. 若 A 为非奇异矩阵（非奇异矩阵是行列式不为 0 的矩阵，也就是可逆矩阵。），则 $det(A^{-1}) = \frac{1}{det(A)}$。

7. 三角（上三角或者下三角）矩阵 A 的行列式等于其主对角线所有元素的乘积

最基础的东西讲完了，剩下的就是一堆 shit 了。

矩阵树定理

Kirchhoff 矩阵树定理（简称矩阵树定理）解决了一张图的生成树个数计数问题。

首先设我们讨论的无向图 $G=(V,E)$ 有 n 个顶点，m 条边。

然后我们把 G 的每条边任意指定一个方向，这样我们就可以定义 G 的关联矩阵 $M(G)$，它是一个 $n\times m$ 矩阵，其中 $M_{i,j}$ 表示：

$$\begin{cases} -1 & v_i\ e_j \\ 1 & v_i\ e_j \\ 0 & otherwise \end{cases}$$

矩阵树定理

然后我们发现它的关联矩阵没有什么卵用，我们考虑重新构建 G 的拉普拉斯矩阵 $L(G)$，它是一个 $n \times n$ 矩阵，构造方法如下：

$$L_{i,j} = \begin{cases} -m_{i,j} & i \neq j, i\ j \quad m_{i,j} \\ deg(v_i) & i = j \end{cases}$$

就某巨佬所言，这个 deg 是指有多少有多少条边与之相连。