非常规大小分块算法初探

ztzshiwo

2017 年 6 月 5 日

瞎 BB

这篇论文比较奇怪, 其中的内容比较多, 而且比较杂乱。里面有些东西实在没办法用 PPT 表示, 我就只能用 PPT 结合论文一起讲了, 其实是大家一起做一下嘴巴选手, 讲一些可能有用但是大部分情况没什么卵月的东西。请准备好一些草稿纸 (数论大佬心算就好了)。

分块?

分块是一种优秀毒瘤的算法它可以用一些奇奇怪怪的复杂度比如说

$$N\sqrt{N}$$

$$Nlog(N)\sqrt{N}$$

$$N\sqrt{Nlog(N)}$$

来解决一些没办法做的或者很难写的问题

分块!

可能有些同学对于一些常见的分块题还不是特别熟练, 那么我们来看几道思博题吧

分块!

Problem1: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及区间加法, 区间求和。

分块!

Problem1: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及区间加法, 区间求和。

把数列分成长度为 S 的块, 每个块维护这个块的和, 区间加法的时候两边不足一块的部分直接暴力, 中间的块一整块一整块加, 区间求和的时候两边不足一块的部分暴力加, 中间一整块加。

分块!

Problem1: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及区间加法, 区间求和。

把数列分成长度为 S 的块, 每个块维护这个块的和, 区间加法的时候两边不足一块的部分直接暴力, 中间的块一整块一整块加, 区间求和的时候两边不足一块的部分暴力加, 中间一整块加。

时间复杂度为 $O\left(N\left(\frac{N}{S}+S\right)\right)$

S 取 $\sqrt{N}$ 时最优, 复杂度为 $O\left(N\sqrt{N}\right)$

分块!

Problem2: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及区间加法, 询问区间内小于某个值 x 的元素个数。

分块!

Problem2: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及区间加法, 询问区间内小于某个值 x 的元素个数。

把数列分成长度为 S 的块, 每个块从小到大排序好, 对于每个修改两边不足一块的部分暴力修改重新排序, 中间的块直接加到标记上, 对于每个查询两边不足一块的直接暴力, 中间的块直接二分

分块!

Problem2: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及区间加法, 询问区间内小于某个值 x 的元素个数。

把数列分成长度为 S 的块, 每个块从小到大排序好, 对于每个修改两边不足一块的部分暴力修改重新排序, 中间的块直接加到标记上, 对于每个查询两边不足一块的直接暴力, 中间的块直接二分

时间复杂度为 $O\left(N\left(\frac{N}{S}logS+SlogS\right)+Nlog(N)\right)$

如果取 $S=\sqrt{N}$ 的时候复杂度为 $O\left(Nlog\left(\sqrt{N}\right)\sqrt{N}\right)$

分块!

这道题细节还是比较多的, 我们请一个人上来讲一下

分块!

这道题细节还是比较多的, 我们请一个人上来讲一下

大佬肯定已经看出来了, 这个题其实有更优的分块大小和更优的复杂度

我们在后面会提到怎么分到最优的复杂度

分块!

Problem3:(以前考试的一道题) 给出一个长为 n 的数列, 以及 m 个操作, 操作涉及区间加法, 询问区间内第 K 大的元素。

$n, m \leq 80000$ $|A_i| \leq 5000000$

分块!

Problem3:(以前考试的一道题) 给出一个长为 n 的数列, 以及 m 个操作, 操作涉及区间加法, 询问区间内第 K 大的元素。

$n, m \leq 80000$ $|A_i| \leq 5000000$

这个题的做法跟上一题差不多啊, 就是在查询的时候多个二分的 log

分块!

Problem3:(以前考试的一道题) 给出一个长为 n 的数列, 以及 m 个操作, 操作涉及区间加法, 询问区间内第 K 大的元素。

$n, m \leq 80000$ $|A_i| \leq 5000000$

这个题的做法跟上一题差不多啊, 就是在查询的时候多个二分的 log

时间复杂度为 $O\left(N\left(logW\left(logS\frac{N}{S}+S\right)+SlogS+\frac{N}{S}\right)\right)$

当取 $S=\sqrt{N}$ 时, 复杂度为 $O(NlogWlog(\sqrt{N})\sqrt{N})$

实际效率

这个时间大概要多少呢?

不考虑常数的话, 最慢大概要 4e9... ...

如果我们取 $S = \sqrt{NlogN}$ 时, 大概就只要 2e9 了... ...

虽然还是慢的飞起啊... ... 我也不知道我是怎么过的

一个小优化

对于区间加法后的区间，我们观察以后可以发现一个性质:

被修改的数和被修改的数彼此之间的大小顺序没有变，未修改的数亦然

一个小优化

对于区间加法后的区间，我们观察以后可以发现一个性质:

被修改的数和被修改的数彼此之间的大小顺序没有变，未修改的数亦然

那么我们直接归并排序一遍就好了吧？

时间复杂度变为 $O\left(N\left(logW\left(logS\frac{N}{S}+S\right)+S+\frac{N}{S}\right)\right)$

其实好像还是没有什么卵月啊... ...

分块!

Problem4: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及单点插入, 单点询问。

分块!

Problem4: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及单点插入, 单点询问。

块状链表的裸题啦... ... 把序列分成长度为 S 的块, 每一个块记录块中第一个数是第几个, 每个块维护一个链表, 然后插入就暴力插入, 查询就暴力查询... ...

还有一个问题, 要是它把一个链表弄得很长, 导致查询复杂度退化怎么办?

分块!

Problem4: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及单点插入, 单点询问。

块状链表的裸题啦... ... 把序列分成长度为 S 的块, 每一个块记录块中第一个数是第几个, 每个块维护一个链表, 然后插入就暴力插入, 查询就暴力查询... ...

还有一个问题, 要是它把一个链表弄得很长, 导致查询复杂度退化怎么办?

我们每隔 $\frac{N}{S}$ 次操作就重构一次, 每次重构的时间复杂度为 $O(N)$, 然后就解决啦

分块!

Problem5: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及询问区间的最小众数。

分块!

Problem5: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及询问区间的最小众数。

这个题如果不强制在线就是莫队 +map 应用啊... ...

那如果强制在线怎么弄呢?

分块!

Problem5: 给出一个长为 n 的数列, 以及 n 个操作, 操作涉及询问区间的最小众数。

这个题如果不强制在线就是莫队 +map 应用啊... ...

那如果强制在线怎么弄呢?

我们可以很显然有一个结论, 对于查询区间 [l,r], 假如把它分成中间的一个序列加上两边不足一块的一些数, 那么答案只有可能是块的众数和两边出现的数

分块!

题解: 把序列分成长度为 S 的块, 每两个块之间预处理众数和众数的出现次数 (这里直接枚举每一块然后向后扫一遍), 同时对于每一数弄个 vector 记录它出现的位置, 对于每个查询我们直接首先找到中间块的众数, 然后对于两边的每个数在 vector 里面二分找出个数然后更新答案就好了

分块!

题解: 把序列分成长度为 S 的块, 每两个块之间预处理众数和众数的出现次数 (这里直接枚举每一块然后向后扫一遍), 同时对于每一数弄个 vector 记录它出现的位置, 对于每个查询我们直接首先找到中间块的众数, 然后对于两边的每个数在 vector 里面二分找出个数然后更新答案就好了

时间复杂度为 $O\left(N\frac{N}{S}+NSlogN\right)$

在 S 取 $\sqrt{N}$ 的时候时间复杂度为 $O(NlogN\sqrt{N})$

分块总结

我们发现分块还是有几个套路的，首先就是考虑暴力怎么搞以及一坨数一起怎么搞，然后分一分块

一些比较常见的套路就是每个块维护一个数据结构或者块重构什么的

分块总结

我们发现分块还是有几个套路的，首先就是考虑暴力怎么搞以及一坨数一起怎么搞，然后分一分块

一些比较常见的套路就是每个块维护一个数据结构或者块重构什么的

~~好的，我的讲课就到这里了~~

~~谢谢大家~~

关于分块大小的讨论

现在开始进入正题。

前 ♂戏

算术 -几何均值不等式

关于分块大小的讨论

现在开始进入正题。

前 ♂戏

算术 -几何均值不等式

算术 -几何平均值不等式, 是一个常见而基本的不等式, 表现了算术平均数和几何平均数之间恒定的不等关系。

算术 -几何平均值不等式

设 $x_1, x_2, \cdots, x_n$ 为 n 个正实数,

它们的算术平均数是

$$A_n = \frac{x_1, x_2, \cdots, x_n}{n}$$

它们的几何平均数是

$$G_n = \sqrt[n]{x_1 * x_2 * \cdots * x_n}$$

算术 -几何平均值不等式

设 $x_1, x_2, \cdots, x_n$ 为 n 个正实数,

它们的算术平均数是

$$A_n = \frac{x_1, x_2, \cdots, x_n}{n}$$

它们的几何平均数是

$$G_n = \sqrt[n]{x_1 * x_2 * \cdots * x_n}$$

不等式表明, 对任意的正实数 $x_1, \ldots, x_n$, 总有:

$$A_n \geq G_n$$

等式成立**当且仅当**$x_1 = x_2 = \cdots = x_n$

证明

我这里介绍一个比较简单好理解的证明

数学归纳法证明

由对称性不妨设 x_{n+1} 是 $x_1, x_2 \cdots x_{n+1}$ 中最大的,

由于 $A_{n+1} = \frac{nA_n + x_{n+1}}{n+1}$

设 $x_{n+1} = A_n + b$, 则 $b \geq 0$, 并且有 $A_{n+1} = A_n + \frac{b}{n+1}$

推推推

根据二项式定理

$$
\begin{aligned}
A_{n+1}^{n+1} &= \left(A_n + \frac{b}{n+1}\right)^{n+1} \\
&\geq A_n^{n+1} + (n+1)A_n^n \frac{b}{n+1} \\
&= A_n^n(A_n + b) \\
&= A_n^n x_{n+1} \\
&\geq G_n^n x_{n+1} \\
&= G_{n+1}^{n+1}
\end{aligned}
$$

回归正题

那么这个东西有什么用呢?

举个栗子

比如对于最简单的 $O(N(S+\frac{N}{S}))$, 可知 $S+\frac{N}{S} \geq 2\sqrt{N}$

当 $S=\frac{N}{S}$ 时能取到最小值, 也就是说 $S=\sqrt{N}$

练习

$$O\left(N\left(\frac{N}{S}+SlogN\right)\right)$$

练习

$$O\left(N\left(\frac{N}{S}+SlogN\right)\right)$$

$$O\left(N\left(\frac{NlogN}{S}+S\right)\right)$$

练习

$$O\left(N\left(\frac{N}{S}+SlogN\right)\right)$$

$$O\left(N\left(\frac{NlogN}{S}+S\right)\right)$$

$$O\left(N\left(\frac{NlogN}{S}+SlogN\right)\right)$$

练习

$$O\left(N\left(\frac{N}{S}+SlogN\right)\right)$$

$$O\left(N\left(\frac{NlogN}{S}+S\right)\right)$$

$$O\left(N\left(\frac{NlogN}{S}+SlogN\right)\right)$$

$$O\left(N\left(\frac{N^2}{S^2}+S\right)\right)$$

简洁方法

如果要通过式子各种算复杂度不是很不爽吗，于是我们得到一种比较简单的方法

假设时间复杂度是 $O(A+B)$，我们直接算 $A=B$ 时的时间复杂度就好了

简洁方法

如果要通过式子各种算复杂度不是很不爽吗, 于是我们得到一种比较简单的方法

假设时间复杂度是 $O(A+B)$, 我们直接算 $A=B$ 时的时间复杂度就好了

粗略证明就是说一个是随着 S 的增大而增大, 一个是随着 S 的增大而减小, 那么自然两个相等的时候最优嘛... ...

简洁方法

如果时间复杂度是 $O(A+B+C)$, 那么我们分别算 $A=B, B=C, A=C$ 时的时间复杂度, 然后比较一下找个最优的就行了

如果时间复杂度是 $O(x_1+x_2+\cdots+x_n)$, 那么我们... ...

简洁方法

如果时间复杂度是 $O(A+B+C)$, 那么我们分别算 $A=B, B=C, A=C$ 时的时间复杂度, 然后比较一下找个最优的就行了

如果时间复杂度是 $O(x_1+x_2+\cdots+x_n)$, 那么我们... ...

我们也想不出这种复杂度对吧?

所以问题就解决了

小技 ♂巧

我们在计算复杂度的时候, 是忽略了常数的, 但是实际上常数对于实际效率的影响还是比较大的

有时候需要我们卡常数时候, 我们可以二分一下调整块的大小, 然后根据程序跑出来的实际效率来选择最终的分块大小

常数优化

下面介绍一个在许多分块问题中都适用的常数优化

我们在区间操作时，暴力处理不在一个整块中的部分时，如果这个部分的数较多，超过了 $\frac{S}{2}$，就可以转换为处理一整块，然后对这个块中本来不需要操作的块进行操作

常数优化

比如在之前的一道区间加法和区间求和的题

假设要对这个块的 [l,r] 进行 +x 的操作，那么我们可以对整个块 +x，然后对于 (r,l+S) 的数 -x，这就等效于之前的操作。

假设要对这个块的 [l,r] 进行求和的操作那么我们可以求出整个块的和，然后把 (r,l+S) 的数减掉，也等于之前的操作

常数优化

比如在之前的一道区间加法和区间求和的题

假设要对这个块的 [l,r] 进行 +x 的操作，那么我们可以对整个块 +x，然后对于 (r,l+S) 的数 -x，这就等效于之前的操作。

假设要对这个块的 [l,r] 进行求和的操作那么我们可以求出整个块的和，然后把 (r,l+S) 的数减掉，也等于之前的操作

当然有些操作还是不能这样做的，比如说区间赋值... ...

Arithmetic Progressions[1]

给出 $A_1, A_2, \cdots, A_N$ 统计满足

$$i < j < k$$

$$A_i + A_k = 2A_j$$

的 (i, j, k) 数量

$(N \leq 30000, A_i \leq 10^5)$

[1] Codechef November Challenge 2012

Tips

如果不考虑各种顺序, 并且 i, j, k 可以相同的话, 就是一个 FFT 裸题

直接搞一个多项式

$$F(x) = \sum_{i=1}^{N} x^{A_i}$$

对于 A_i 把 $F(x)^2$ 的第 $2A_i$ 项系数加上, 总和就是答案

那如果讨论顺序的话怎么弄呢?

题解

我们考虑分块, 把序列分成长度为 S 的块，对于每一个块构建两个多项式

设这个块的范围为 (l,r)

$$F(x) = \sum_{i=1}^{l} x^{A_i}$$

$$G(x) = \sum_{i=r}^{N} x^{A_i}$$

对于块内的每个数 A_i 把 $F(x)G(x)$ 的第 $2 * A_i$ 项系数累加进答案

题解

在上个部分里我们累计了 i,j,k 都不在同一个块内的情况

接下来我们考虑 i, j, k 中有至少两个数在同一个块内的情况

对于每个块我们暴力枚举其中两个数的位置, 假设这两个位置为 x,y, 弄两个桶维护 $1 \sim x-1$ 和 $y+1 \sim N$ 的数

然后分情况讨论加一加就好了

题解

在上个部分里我们累计了 i,j,k 都不在同一个块内的情况

接下来我们考虑 i, j, k 中有至少两个数在同一个块内的情况

对于每个块我们暴力枚举其中两个数的位置, 假设这两个位置为 x,y, 弄两个桶维护 $1 \sim x-1$ 和 $y+1 \sim N$ 的数

然后分情况讨论加一加就好了

时间复杂度为 $O\left(\frac{N}{S}S^2 + \frac{N}{S}WlogW\right)$

当 $S = \sqrt{WlogW}$ 时最优，复杂度为 $O\left(N\sqrt{WlogW}\right)$

End

其实我前面讲的东西只占了整篇论文的前面一部分，但是我觉得这些内容比较重要，而且在竞赛的时候有可能会用到，就重点讲了一下。

接下来的论文就是一大堆卡常以及理论复杂度很优但实际效率毫无意义，又难写又难调的算法。有些地方我会一笔带过，有兴趣的同学可以听一听，反正我们现在是嘴巴选手，看一看还是很简单的嘛

End

其实我前面讲的东西只占了整篇论文的前面一部分，但是我觉得这些内容比较重要，而且在竞赛的时候有可能会用到，就重点讲了一下。

接下来的论文就是一大堆卡常以及理论复杂度很优但实际效率毫无意义，又难写又难调的算法。有些地方我会一笔带过，有兴趣的同学可以听一听，反正我们现在是嘴巴选手，看一看还是很简单的嘛

至于写的话... ...

End

其实我前面讲的东西只占了整篇论文的前面一部分，但是我觉得这些内容比较重要，而且在竞赛的时候有可能会用到，就重点讲了一下。

接下来的论文就是一大堆卡常以及理论复杂度很优但实际效率毫无意义，又难写又难调的算法。有些地方我会一笔带过，有兴趣的同学可以听一听，反正我们现在是嘴巴选手，看一看还是很简单的嘛

至于写的话... ...

活着不好吗？

由乃救爷爷[2]

给出一个长度为 n 的序列，m 个询问，查询 [l,r] 的最大值

$n, m <= 20000000$

数据完全随机

这个题应该有人看了吧，请大佬上来秒掉

[2]洛谷 5 月月赛 R2

由乃救爷爷[2]

给出一个长度为 n 的序列，m 个询问，查询 [l,r] 的最大值

$n, m <= 20000000$

数据完全随机

这个题应该有人看了吧，请大佬上来秒掉

虽然我们在之前学了一个 $O(N) - O(1)$ 的算法，但是我可以很负责的告诉你，当你辛辛苦苦写完的时候，你会发现这个算法 T 得飞起

~~其实我也没写过~~

[2]洛谷 5 月月赛 R2

还是考虑分块，把序列分成长度为 S 的块，求出每个块的最大值以及前缀最大值和后缀最大值

然后我们预处理出块与块之间的 RMQ, 查询的时候直接 RMQ 与两边不足一块的前缀后缀一起取 Max

如果 l,r 在同一个块内的时候，我们就只能暴力算了

时间复杂度为 $O(\frac{n}{S}log\left(\frac{n}{S}\right)+mS)$

因为数据是完全随机的，所以实际上可以认为每次查询都是 $O(1)$ 的

在 $S=log(n)$ 时最优，大概是 $O(n)$ 的

最后祝您身体健康，再见!