当希望的最后一片积雨云消逝在炽热的晚霞中之时 题解

闲话

经过上一次多校的经验，我深刻的认识到根本没有什么人写T4

于是经过深思熟虑， 决定把T4和T3的位置交换一下，这样就会有很多人写了?

不过反正也是一道大水题，就没什么好说的了。

不过说一下这个题的出题历程

一开始只有K=2的版本，后来有一天晚上睡觉前我想到了K=3的做法，但是过了几天我发现K=3有更简单的做法，然后后来又有一天晚上睡觉前我想到了K=4的解法，然后就有了这个题捏。

本题综合考察了根号分治，平衡规划，莫队二次离线，分块，二维偏序等算法。

略卡常，勿喷。

首先有一点，通过观察数据范围，我们可以得知需要对$K=2,3,4$各自实现一种算法，并且因为数据范围越来越小，所以可以推测复杂度越来越高。

~~三合一，勿喷~~

虽然三分代码加起来有将近340行，但是单独看每一个最多也只有100多行。

而且从开始到调完我只写了一个多小时，~~虽然卡常又卡了一个小时~~。

为了弥补看上去不太良心的问题，提供了很多的大样例，也并没有对选手的空间进行过多要求，时间在std完成的时限上多开了1s。

题解

以下为了方便定义 $f(x)$ 为将x周围的点权都加一的操作,$g(x)$ 为 x 周围的点权之和。

以下为了方便定义 $f(x)$ 为将x周围的点权都加一的操作,$g(x)$ 为 x 周围的点权之和。

以下为了方便定义 $f(x)$ 为将x周围的点权都加一的操作,$g(x)$ 为 x 周围的点权之和。

~~重要的事情说三遍~~

subtask1

枚举一条边，判断。

复杂度分别是$O(mq)$

期望得分3分。

subtask2

K=2

考虑一条边(x,y)，我们求出这两个点在排列里的位置设为(t_x,t_y)，假设$t_x<t_y$，那么它对一个区间有贡献当且仅当$l\le t_x,ty\le r$，离线二维数点即可，可用CDQ分治或树状数组扫描线。

复杂度$O((m+q)logn)$

结合前边的算法期望得分17分。

subtask3

K=3,$n,m,q\le 5000$

首先对于区间内的所有点，执行$f(p_i)$。

然后再查询这个区间内所有点的$g(p_i)$之和。

复杂度$O(qm)$

期望得分22pts

subtask4

$\sum deg^2\le 10^7$

枚举可产生贡献的二元组，只有$\sum deg^2$ 个，剩下的按照算法二做就行了。

使用分块代替树状数组可以平衡到$O(\sum deg^2+q\sqrt{n})$

期望得分36pts

subtask5

两种做法，一种是类似算法五枚举所有可能的二元组，这是$O(n)$的，然后再二维偏序。

另一种是跑莫队，动态维护链上合法的点对数量。

都可通过，期望得分46pts

subtask6

考虑莫队，每次转移时需要支持对某个点执行$f(x)$,以及查询某个点的$g(x)$。

直接莫队复杂度不太对，我们考虑魔改一下。

考虑根号分治。

我们抛弃传统的每$\sqrt{n}$ 个点为一个块，改为:

如果一个点度数大于$\sqrt{m}$，单独成一个块。

其余的块要保证每个块的总度数和$\le\sqrt{m}$

在此基础上分最少的块，显然块的个数是$O(\sqrt{m})$。

然后对这种分块跑莫队，分析一下复杂度。

对于一个块，右指针的移动时单调的，总复杂度为$O(\sum deg = m)$,因此这部分复杂度$O(m\sqrt{m})$

对于左指针，如果这个块内有度数$> \sqrt{n}$的点，那么块大小为一，这部分复杂度是$O(m)$的。

否则，因为一个块内度数和是$O(\sqrt{m})$,单次移动的复杂度是$O(\sqrt{m})$

总复杂度$O((m+q)\sqrt{m})$

期望得分62pts

subtask7

考虑dp求解。

记$dp[i][k]$为经过k条边，到达点i的路径条数，初始时，$dp[p_i][0] = 1$

转移很简单，$dp[x][i] = \sum_{(x,y)\in E} dp[y][i-1]$

复杂度$O(Kmq)$

期望得分72pts.

subtask8

留给神秘的做法。

subtask9

留给一些神秘做法，或者写了正解被卡常的做法。

subtask10

我们依然考虑莫队。

需要支持对某个点做$f(x)$，以及查询某个点周围点的$g(x)$之和。

这是等价于对某个点周围的点做$f(x)$，查询某个点的$g(x)$

依旧按照$subtask6$的思路进行。

这样我们的莫队，就只需要支持对某个点做$f(x)$,以及查询某个点的值就行了。

这两种操作各有约$m\sqrt{m}$次。

定义$F(i,x)$为对$1-i$的p_i执行对某个点周围的点执行$f(x)$之后，$g(x)$的值。

显然$F(i,x)$是具有可减性的，因此我们考虑莫队二次离线。

以右端点右移为例，答案的增量为 $\sum_{i=r+1}^{R} F(i-1,i) - F(l-1,i)$

前半部分我们可以预处理，后半部分我们可以扫描线。

离线之后，我们就有$O(m)$次$f(x)$次操作，$m\sqrt{m}$次查询某个点的值操作。

这是一个经典问题，考虑平衡规划。

考虑设阈值为B。

把度数大于B的点称为重点，小于的称为轻点。

修改时，如果一个点是轻点，我们直接做$f(x)$。

否则，我们就在这个点记录一下对这个点执行$f(x)$的次数。

这个复杂度是$O(B)$的。

查询时，这个点的权值等于他本来的权值加上周围的重点做$f(x)$的次数之和。

因为一个点周围最多有$\frac{m}{B}$个重点，因此这部分复杂度$O(\frac{m}{B})$。

因为查询和修改的总代价相同时最优。

即 $mB = m\sqrt{m}\frac{m}{B}$

取$B = m^{\frac{3}{4}}$

总复杂度为$O(m^{\frac{7}{4}})$

看上去复杂度很炸裂，但是因为跑不满，所以实际上跑$1e5$只需要不到2s。

具体实现的时候，因为后边严重跑不满，所以可以把B调小一点，实测取$B = m^{0.4-0.6}$都是可以过的。

适当的卡常和优化是十分有必要的。

总结

可能还有复杂度比较优秀的做法，但是出题人是不太会了。

仅看我的做法，可以发现这道题最大的特殊的地方就在有n, m是同阶的，因此我们才得以使用一些特殊分块以及平衡规划把传统莫队中复杂度为n的部分替换成m并完成降次。

可以看成是时间的充分利用。