一些简(sha)单(bi)技(dong)巧(xi) By SYC

分块

CONTENTS

序列分块

- 动态的序列
  - 设块的大小为S,有N/S个块
  - 区间操作 -> O(S)全局操作+O(N/S)单点操作
  - 区间询问 -> O(S)全局询问+O(N/S)单点询问
  - 带$\sqrt{n}$的复杂度在分治中不会追加log
- 静态的序列
  - 预处理块到块的答案，然后添加多余的部分
- 莫队算法(offline)

CodeChef COUNTARI

- 给一个长为N的序列A(N <= 10^5,A[i] <= 30000)
- 计算有多少对(i,j,k)满足Aj-Ai=Ak-Aj

CodeChef COUNTARI

- 给一个长为N的序列A(N <= 10^5,A[i] <= 30000)
- 计算有多少对(i,j,k)满足Aj-Ai=Ak-Aj
- 将序列分块，从左往右扫每个块，讨论i,j,k属于的块位置。
- (1) i,j,k 在当前块 : O(NSlogw) 暴力+map
- (2) i,j在当前块，k在后面 : O(NSlogw)暴力+map
- (3) j,k在当前块，i在前面 : O(NSlogw)暴力+map
- (4) i在前面，j在当前，k在后面 : O(N/Swlogw)FFT
- 取S =$\sqrt{w}$ 最优，复杂度O(N$\sqrt{w}$ logw)

静态RMQ

- 线段树 O(N) - O(logN)
- Sparse Table O(NlogN) - O(logN)
- 分块(Block O($\sqrt{n}$)) O(NloglogN) - O(logloglogN)/O(1)
- 分块2(Block O(log N)) O(Nlog*N) - O(loglog*N)/O(1)
- 分块3(Method of Four Russians) O(N) - O(1)

按数量分块

- 考虑方程A1+A2+A3+..+Am=n
- 那么A中不同元素是O($\sqrt{n}$)的.

Codechef SERINVS

- 有一个长为N的排列A
- 有Q次询问,每次询问a,b,c,d(a<=b<c<=d,d-c=b-a)
- 问有多少个排列V[1..b-a+1]满足A[a+i-1]<A[c+V[i]-1]
- 答案对10^9+7取模
- N,M<=10^5,逆序对数目S<=10^5

Codechef SERINVS

- 对于每个a<=i<=b,定义Ci为c<=j<=d且A[j]>A[i]的j个数
- 定义 w[x] 为 Ci 为 x 的 i 个数, 再设 L = b - a + 1
- 那么答案就是 $\prod_{i=1}^{L}(i-\sum_{x=1}^{i-1}w[x])^{\underline{w[i]}}$
- 注意到w[i]只有 O($\sqrt{S}$) 种不同的取值,把这些值暴力找出来就好了。
- 时间复杂度O(N$\sqrt{S}$ log N).

IOI2009 Region

- 有一个树 T , 每个点上有一个颜色 col[i]
- 有 Q 次询问, 每次询问给出 a, b
- 计算有多少个点对 (u, v) 满足 col[u] = a, col[v] = b
- 且 u 是 v 的祖先
- 强制在线, 暂定内存 2G.
- |T|, Q <= 300000

IOI2009 Region

- 把颜色按照阈值S分成大颜色和小颜色.
- 对于大颜色C:
  - 可以O(N)递推每种颜色子树和祖先中颜色C的点数。
  - 时间复杂度O(N^2/S+Q)
- 对于小颜色gt:
  - 再设一个阈值D,有块状数组维护到根路径的权值信息.
  - 时间复杂度O(QS+N(D+N/D))

树上分块

- 有 3 种方法

Classic Approach

- dfs 这个树
- 实时维护一个栈
- 假设当前 dfs 到 x , 当前栈中元素个数为 tot.
- dfs 所有子树，如果某个子树做完以后栈中元素 tot’>= tot + B, 那么就把栈中最后 tot’ - tot 个拿出来合成一个块.
- 做完所有子树后,把 x 加入栈中.
- 最后把所有栈中的剩余元素合成一个块
- 这样做的话每个块的大小是[B, 3B]
- 但是块根可能不在块中

BZOJ COT2强制在线

- 有个 N 个点的树 T.
- 每个点有个点权w[i]
- 有 Q 次询问
- 每次询问给出 x, y
- 询问 x -> y 路径上不同的权值个数
- N <= 40000, Q <= 10^5
- TL = 2s
- ML = 400MB

BZOJ COT2强制在线版

- 把树按照刚才的方法分块
- 可用主席树预处理出从第X个块的根到Y这个点路径的权值信息.
- 询问的时候再把多出来的点修改一下就好了
- 时空复杂度O(N*sqrt*log)
- 但是这样做会爆空间
- 可以把块个数开小来优化空间
- 可以参照orzsyf的代码

BZOJ COT2强制在线版

- 还有另外一种姿势
- 只预处理第X的块根到Y这个点的答案,而不是存权值.
- 考虑计算路径(x, y), 设 x 所在的块根是 u, y 所在的是 v.
- 若 u == v 就暴力.
- 不妨假设 dep[u] > dep[v], 那么(u, y) 的答案就知道了.
- 现在只需把(u, x) 的权值加入路径.
- 对每个点预处理出到1号点权值为w最深的点的深度,记为 pre(u, w)
- 检查(u, x)上的某个权值w是否已经出现过只需检查:
  - dep(lca(u, y)) <= dep(pre(u, w))
  - 或者 dep(lca(u, y)) <= dep(pre(v, w))
- 预处理pre(u, w)可以可持久化块状数组.
- 时间复杂度O((n+Q)sqrt(n))

Xllend3's Approach

- 设一个阈值S
- DFS这个树,找出每个点的dep和size.
- 把所有dep%S==0的点标成关键点.
- 但是这样关键点有很多
- 把所有size<=S的关键点删除.

Analysis

- 关键点个数是O(N/S)的.
- 任何一个点,离他最近的关键祖先是O(S)的.
- 任意两个相邻关键点之间的距离是O(S)的.

用Xllend分块解决链询问

- BZOJ COT2强制在线版
- 将树Xllend分块
- 然后套用之前的做法即可.

用Xllend分块解决子树问题

- IOI2009 Region

用Xllend分块解决子树问题

- IOI2009 Region
- 假设选了S个关键点
- 把树Xllend分块,并将关键点两两lca也标为关键点.
- 把路径分成经过了关键点和没有经过关键点的.
- 对于没有关键点的路径,一个点暴力向上爬O(N/S)步.
- 这里时空复杂度O(N^2/S)
- 对于经过了关键点的,必然是一条虚树边上的点走到关键点子树.
- 预处理每条虚边和关键点子树的权值信息.
- 时空复杂度O(NS)
- 总复杂度O(N^2/S+NS)

Shooter's Approach

- Xllend3的做法是在树上有意识地选取了若干个点.
- Shooter的做法在树上随机S个点标成关键点.
- 然后把S个点按照dfs序排序,把相邻两个点的lca也加入关键点
  - 那么一个点期望离他最近的关键点的距离:
    - $$\sum_{i=1}^{n-m} \frac{C(n-i,m)}{C(n,m)} = \frac{n+1}{m+1}$$

用Shooter的方法解决链询问

- BZOJ COT2强制在线版
- 按照shooter的方法标识关键点.
- 然后暴力爬一下就好了.

Summary

- 三种方法的本质都是标识了若干个关键点
- 然后预处理关键点到关键点的信息
- 询问时把多余的部分修改一下
- 使得个数是O(S)时,一个点到最近关键点的距离是O(n/S)的。
- 都保证可以找到一个距离是O(n/S)的祖先
- 但是叶队那个是期望O(n/S)的.

点分分块

- 点分中的size和是O(nlogn)的
- 点分树上size >= T 的节点数是O(n/T)的.

CTSC 珠宝商

- 有一个串 S
- 有一个树 T
- 定义 F(i, j) 表示树上 i 到 j 的路径在 S 中出现了多少次
- 计算 F(i, j) 的和
- |T|, |S| <= 50000

CTSC 珠宝商

- 点分这个树.
- 当size<=X的时候O(X^2)暴力.
- 当size>X的时候O(size+|S|)后缀树.
- 时间复杂度O(|S|*n/X + NlogN + NX)

IOI2009 Region

- 考虑点分,考虑所有经过重心的路径.
- 对于一个询问(a, b), 设当前的重心是w, 当前联通快最浅的点是c
- 那么这个子树对答案的贡献是(c,w)上a的个数乘上别的子树b的个数.
- 对于size<=X的块,O(X^2)暴力并预处理答案.
- 对于size>X的块,预处理(c,w)的权值信息和子树权值信息.
- 这步时间复杂度O(NlogN),空间O(N^2/X)
- 询问时枚举大块就行了.
- 时间复杂度O(QN/X+N^2/X+NlogN+NX).

Mo's algorithm

- 假设有m个询问(l1,r1),(l2,r2),..(lm,rm)
- 如果维护的信息只支持单点加(or 删).
- 那么从询问(l1,r1)到(l2,r2)所需要的操作数是|l1-l2|+|r1-r2|
- 把询问区间投影到平面上.
- 2维曼哈顿距离最小哈密尔顿路径!
- 这个问题的一个上界是O(W^{3/2}),W是权值范围.
- 构造方法是把横坐标分块,每块内的点按纵坐标升序排序.
- 这个可以扩展到k维,一个上界是O(W^{2-1/k}).

Codeforces 765F

- 有一个长为 N 的序列 A .
- 有 Q 次询问, 每次询问给出 l, r
- 要求找一对 l <= i < j <= r 使得 |A[i] - A[j]| 最小
- N <= 10^5, Q <= 3 * 10^5

Codeforces 765F

- 考虑莫队.
- 你可以在加入的时候快速维护答案
- 但是你只能做到快速删除(双向链表)
- 因为插入的时候你要知道插入的位置.
- 考虑现在统计到了某个块[l,l+S-1]
- 把左端点在这个块的询问按照右端点排序.
- 然后O(n)构建一个记录[l,n]权值的双向链表.
- 然后从n到l+S从后往前删一遍
- 并记录修改了哪些位置.
- 然后从前往后做,你就知道应该插入哪个位置了.
- 对[l,l+S-1]这一块也做同样的事情即可.
- 时间复杂度O(N^{3/2})

Mo’s Tree version

- 方法一:取dfs序然后序列莫队.
- 方法二:把树分块然后每次从上一个询问暴力爬到下一个询问.

定时重构

- 把操作分块,并设一个阈值D.
- 每Q/D次修改就重构一次.
- 每次询问的时侯暴力加入未重构的部分.

BZOJ 带插入区间K小值

- 有一个初始长为N的序列A.
- 要求支持:
  - 1 x y 在第x个位置后插入y.
  - 2 x y 把A[x]=y
  - 3 l r k 询问区间[l, r]的第k小值.
- N,A[i],Q<=30000

BZOJ 带插入区间K小值

- 把序列分块.
- 对每个块维护一个权值线段树.
- 询问把O(N^{1/2})个单点和整个块的权值线段树拿出来一起二分.
- 修改的时候修改块的权值线段树.
- 插入的时候也插到对应的块中.
- 但是这样一个块的大小会爆.
- 所以每O(N^{1/2})次插入后重构一下.

带Link/Cut的路径K小值

- 有一个树T.
- 要求支持:
  - Link/Cut
  - 询问路径权值K小值.
- N,Q<=30000

带Link/Cut的路径K小值

- 如果没有Link/Cut,那么可以主席树解决.
- 如果Link/Cut了一次,那么一条新树上的路径对应原树上的最多2条路径，在8个主席树上一起二分即可.
- 如果Link/Cut了K次,那么一条新树上的路径对应原树上的最多(K+1)条路径，在O(K+1)个主席树上一起二分即可.
- 但是次数一多的话,新树一条路径上可能对应原树上很多条路径.
- 所以每O(M^{1/2})次重构一下,这样K<=O(M^{1/2})了.

Blocks on bits

- 考虑一个w位的数字
- 把数字分成最高的w/2位(高位)和最低的w/2位(低位).
- 用一个2^w的数组F[S][T]表示修改高位为S,询问低位为T时的信息.
- 这样修改和询问只会修改2^{w/2}个位置的值.
- 本质上就是一个2层2^{w/2}叉的线段树.

HDU 5735

- 有一个有根树 T.
- 每个点有一个权值w[i]
- 要求你找一个点列 X[1], X[2], X[3]… X[k]
- 满足
  - 1. 对于任意 i > 1, X[i - 1] 是 X[i] 的祖先
  - 2. w[X[1]] + (w[X[2]] opt W[X[1]]) + … + (w[X[k]] opt w[X[k -1]] 最大
  - 其中opt是任意按位逻辑运算(会给出真值表).
- |T| <= 2^16, w[i] <= 2 ^16

HDU 5735

- 显然的DP:
  - F[i] <- max(j is an ancestor, F[j] + W[j] opt W[i])
  - F[i] <- W[i]
- 用W[S][T]表示j的高位是S,i的低位是T时F[j]+(j的低位 opt T)的最大值.
- 询问时T已知.枚举S,计算W[S][T]+(i的高位 opt S)的最大值.
- 修改时S已知.枚举T,用(F[j] + (j的低位 opt T))更新W[S][T].
- 回溯的时候大力撤销
- 时间复杂度O(2^{w/2}*n)

Codechef GOODPROB

- 有一个长为N的数列A
- 定义F(i, j) = 1 if (A[i] & A[j]) = A[i] or (A[i]&A[j])=A[j]
- 定义Max(i, j) = max(a[i], a[i+1], .., a[j])
- 求sum(i<j, F(i, j) * Max(i, j))
- N<=100000,A[i]<=2^14.

Codechef GOODPROB

- 考虑(a[i]&a[j])=a[i],另一种情况类似.
- 用W[S][T]表示A[i]高位是S,A[j]低位是T的时的数的集合.
- 在询问a[j]的时候,枚举a[j]高位的子集,然后询问:
  - sum(i ∈ W[S][T], max(a[i], a[i+1],a[i+2],..a[j]))
  - 这个对W[S][T]开一个维护后缀max的单调栈,在上面二分一下就好了.
- 插入a[i]的时候,枚举a[i]低位的超集,然后更新一下单调栈即可.
- 时间复杂度O(NlogN*2^7)

Codechef GOODPROB

- 还有一种做法是分治.
- 考虑左半边[l,m]对[m+1,r]的影响.
- 对每个点处理出到中点的max,然后sort一下.
- 只要统计个数就好了.

自己YY的垃圾题

- 有一个长为N的序列A,初始全是1
- 有M次操作,每次操作是2种之一:
  - 1 l r x y 对于$i \in [l,r]$ A_i *= (x^i+y)
  - 2 x 询问A_x的值
- N,M<=30000，答案对998244353取模

自己YY的垃圾题

- CDQ分治，变成修改完以后再询问。
- 把右边的所有询问位置拿出来分块,设块个数为S.
- 对于左边一次修改,O(N/S)次单点乘,和O(S)次全局乘
- 考虑右边每个块,找出包含他所有的全局乘,每个全局乘对应一个一阶多项式.
- 把一阶多项式用分治FFT乘出来,总复杂度O(NSlog^2N)
- 然后对每块多项式mod+暴力计算点值,总复杂度O(NSlogN+N^2/S)
- 取S=$\sqrt{n}$ / logN 最优, 于是T(n)=2T(n/2)+O(N$\sqrt{n}$logN)
- 于是总复杂度O(N$\sqrt{n}$logN)
- 如果不用分块而用线段树,复杂度是O(Nlog^4N)的。