|  |  |
| --- | --- |
| **常用 ACM 模板** |  |
| **目录** |  |
| [一、算法 ...................................................................................................................................................................................................................................................................................................................................](#page3) | [3](#page3) |
| [1、手写二分搜索 .........................................................................................................................................................................................................................................................................................................](#page3) | [3](#page3) |
| [（1）lower\_bound .............................................................................................................................................................................................................................................................................................](#page3) | [3](#page3) |
| [（2）upper\_bound ............................................................................................................................................................................................................................................................................................](#page4) | [4](#page4) |
| [2、LIS 系列 .....................................................................................................................................................................................................................................................................................................................](#page4) | [4](#page4) |
| [（1）NlogN 时间最长递增子序列：lower\_bound .................................................................................................................................................................................................................................](#page4) | [4](#page4) |
| [（2）NlogN 时间最长非递减子序列：把 lower\_bound 改为 upper\_bound ................................................................................................................................................................................](#page4) | [4](#page4) |
| [（3）NlogN 时间输出所有 LIS 的方案 .......................................................................................................................................................................................................................................................](#page4) | [4](#page4) |
| [（4）NlogN 时间求下标以 i 结尾的最大长度：线段树维护区间最大值优化朴素 dp ..............................................................................................................................................................](#page6) | [6](#page6) |
| [3、康托展开 ..................................................................................................................................................................................................................................................................................................................](#page6) | [6](#page6) |
| [4、0-1 背包记录路径(以 Installing Apps 题为例) .............................................................................................................................................................................................................................................](#page8) | [8](#page8) |
| [5、多重背包二进制优化转 0-1 背包(以焦作网络赛 Transport Ship 题为例) ......................................................................................................................................................................................](#page10) | [10](#page10) |
| [6、斜率优化 dp .........................................................................................................................................................................................................................................................................................................](#page11) | [11](#page11) |
| [（1）一维斜率优化 dp(以打印单词 hdu3507 那个经典入门题为例) ...........................................................................................................................................................................................](#page11) | [11](#page11) |
| [（2）二维斜率优化 dp(以炸铁路 hdu2829 那个经典入门题为例) ................................................................................................................................................................................................](#page12) | [12](#page12) |
| [7、三分 .........................................................................................................................................................................................................................................................................................................................](#page16) | [16](#page16) |
| [（1）整数 ...........................................................................................................................................................................................................................................................................................................](#page16) | [16](#page16) |
| [（2）分数 ...........................................................................................................................................................................................................................................................................................................](#page17) | [17](#page17) |
| [二、数据结构 ......................................................................................................................................................................................................................................................................................................................](#page17) | [17](#page17) |
| [1、LCA ..........................................................................................................................................................................................................................................................................................................................](#page18) | [18](#page18) |
| [（1）二分搜索版 .............................................................................................................................................................................................................................................................................................](#page18) | [18](#page18) |
| [（2）tarjan 版 ...................................................................................................................................................................................................................................................................................................](#page19) | [19](#page19) |
| [2、并查集 ....................................................................................................................................................................................................................................................................................................................](#page23) | [23](#page23) |
| [3、树状数组 ...............................................................................................................................................................................................................................................................................................................](#page24) | [24](#page24) |
| [（1）一维区间更新单点查询 ......................................................................................................................................................................................................................................................................](#page24) | [24](#page24) |
| [（2）一维区间更新区间查询 ......................................................................................................................................................................................................................................................................](#page25) | [25](#page25) |
| [（3）二维树状数组区间更新区间查询....................................................................................................................................................................................................................................................](#page27) | [27](#page27) |
| [4、可持久化线段树(主席树) .................................................................................................................................................................................................................................................................................](#page30) | [30](#page30) |
| 1 |  |

|  |  |
| --- | --- |
| [（1）只可查询区间第 k 小、不可修改（时间复杂度：](#page30)+，空间复杂度：+[） ..............................................................................................................](#page30) | [30](#page30) |
| [（2）例题(POJ2104) .......................................................................................................................................................................................................................................................................................](#page32) | [32](#page32) |
| [（3）区间查询第 k 小，可单点修改（时间复杂度：+ 2 ，空间复杂度：+ 2 ） .............................................................................................................](#page35) | [35](#page35) |
| [（4）例题(ZOJ2112) .......................................................................................................................................................................................................................................................................................](#page38) | [38](#page38) |
| [5、树链剖分(以 BZOJ4034 为例) .........................................................................................................................................................................................................................................................................](#page43) | [43](#page43) |
| [6、线段树 ....................................................................................................................................................................................................................................................................................................................](#page48) | [48](#page48) |
| [（1）同时支持区间加、区间乘、区间求和 ..........................................................................................................................................................................................................................................](#page48) | [48](#page48) |
| [（2）区间加等差数列，单点查询 .............................................................................................................................................................................................................................................................](#page50) | [50](#page50) |
| [（3）区间加等差数列，区间查询 .............................................................................................................................................................................................................................................................](#page52) | [52](#page52) |
| [7、树分治(以 poj1741 统计树上距离不超过 k 的点对为例) .....................................................................................................................................................................................................................](#page55) | [55](#page55) |
| [三、图论 ................................................................................................................................................................................................................................................................................................................................](#page58) | [58](#page58) |
| [1、最短路 ....................................................................................................................................................................................................................................................................................................................](#page58) | [58](#page58) |
| [（1）Dijkstra(不能处理负边图) ..................................................................................................................................................................................................................................................................](#page59) | [59](#page59) |
| [（2）SPFA-LLF 优化(能处理负边图).........................................................................................................................................................................................................................................................](#page60) | [60](#page60) |
| [（3）SPFA-LLL 优化(能处理负边图) .........................................................................................................................................................................................................................................................](#page62) | [62](#page62) |
| [2、最小生成树 ...........................................................................................................................................................................................................................................................................................................](#page63) | [63](#page63) |
| [四、数学 ................................................................................................................................................................................................................................................................................................................................](#page64) | [64](#page64) |
| [1、组合数(lucas 定理+费马小定理，要求模数 p 是素数) .........................................................................................................................................................................................................................](#page64) | [64](#page64) |
| [2、快速拉格朗日插值法求和 ...............................................................................................................................................................................................................................................................................](#page66) | [66](#page66) |
| [(1)例题(计算通式 3 + 2 + + 1的前3 × 108的和) ...........................................................................................................................................................................................................................](#page68) | [68](#page68) |
| [3、预处理 ~ 内数字 和阶乘的逆元，计算组合数.................................................................................................................................................................................................................................](#page72) | [72](#page72) |
| [4、线性基 ....................................................................................................................................................................................................................................................................................................................](#page73) | [73](#page73) |
| [（1）查第 k 小（不支持动态插入） ........................................................................................................................................................................................................................................................](#page73) | [73](#page73) |
| [（2）查最大、最小、合并（支持动态插入） ......................................................................................................................................................................................................................................](#page75) | [75](#page75) |
| [5、FWT .........................................................................................................................................................................................................................................................................................................................](#page77) | [77](#page77) |
| [五、字符串 ...........................................................................................................................................................................................................................................................................................................................](#page80) | [80](#page80) |
| [六、其他 ................................................................................................................................................................................................................................................................................................................................](#page80) | [80](#page80) |
| [1、输入挂 ....................................................................................................................................................................................................................................................................................................................](#page80) | [80](#page80) |
| [(1)slowIO .............................................................................................................................................................................................................................................................................................................](#page80) | [80](#page80) |
| [(2)fastIO ...............................................................................................................................................................................................................................................................................................................](#page80) | [80](#page80) |
| [2、常用函数 ...............................................................................................................................................................................................................................................................................................................](#page82) | [82](#page82) |
| [（1）字符数组分割 strtok ............................................................................................................................................................................................................................................................................](#page82) | [82](#page82) |
| 2 |  |

[（2）数字转字符数组 sprintf](#page83) [83](#page83)

[（3）string 转字符数组](#page83) [83](#page83)

[（4）nth\_element(把第 n 小的元素放在第 n 位置)](#page83) [83](#page83)

[（5）编译器内置的二进制处理函数](#page84) [84](#page84)

[3、BM 算法(用于求线性递推的第 项)](#page86) [86](#page86)

[4、pb\_ds 库](#page89) [89](#page89)

[(1)红黑树(以 BZOJ3224 为例)](#page89) [89](#page89)

[(2)gp\_hash\_table(像数组一样使用)](#page91) [91](#page91)

[5、快速随机数](#page92) [92](#page92)

[6、单样例输入对拍程序/批处理脚本](#page92) [92](#page92)

[(1)脚本：windows](#page92) [92](#page92)

[(2)程序：windows](#page93) [93](#page93)

[(3)程序：Ubuntu](#page93) [93](#page93)

[7、集合的整数操作汇总](#page94) [94](#page94)

**一、算法**

**1、手写二分搜索**

（1）lower\_bound

//[low, high]

int lower\_bound0**(**int **\***a**,** int low**,** int high**,** int key**){**

int mid**;**

**while(**low **<** high**){**

mid **= (**high **-** low**) /** 2 **+** low**;**

**if(**a**[**mid**] >=** key**)**high **=** mid**;**

3

**else** low **=** mid **+** 1**;**

**}**

**return** low**;**

**}**

（2）upper\_bound

/\*[low, high]

ȧa[high] <= key, return high;

\*/

int upper\_bound0**(**int **\***a**,** int low**,** int high**,** int key**){** **while(**low **<** high**){**

int mid **= (**high **-** low**) /** 2 **+** low**;**

**if(**a**[**mid**] <=** key**)**low **=** mid **+** 1**;**

**else** high **=** mid**;**

**}**

**return** low**;**

**}**

**2、LIS 系列**

（1）NlogN 时间最长递增子序列：lower\_bound

（2）NlogN 时间最长非递减子序列：把 lower\_bound 改为 upper\_bound

（3）NlogN 时间输出所有 LIS 的方案

#include<bits/stdc++.h>

**using namespace** std**;**

4

const int MAXN **=** 1e5 **+** 10**;**

int n**,** a**[**MAXN**],** dp**[**MAXN**],** par**[**MAXN**];**

map**<**int**,** int**>** mp**;**

vector**<**int**>** v**[**MAXN**];**

void print**(**int x**,** int y**) {**

**if(**x **>** 0**) {**

**for(**inti **=** 0**;** i **<** v**[**x **-** 1**].**size**(); ++**i**) {**

**if(**a**[**v**[**x **-** 1**][**i**]] <** a**[**v**[**x**][**y**]]){**

print**(**x **-** 1**,** i**);**

**break;**

**}**

**}**

**}**

printf**(**"%d "**,** a**[**v**[**x**][**y**]]);**

**}**

int main**() {**

freopen**(**"input.txt"**,** "r"**,** stdin**);**

freopen**(**"output2.txt"**,** "w"**,** stdout**);**

**while(~**scanf**(**"%d"**, &**n**)) {**

**for(**inti **=** 0**;** i **<** n**; ++**i**)**scanf**(**"%d"**,** a **+** i**); for(**inti **=** 0**;** i **<** n**; ++**i**)**v**[**i**].**clear**();** memset**(**dp**,** 0x3f**,** **sizeof(**dp**));**

memset**(**par**, -**1**,** **sizeof(**par**));**

mp**.**clear**();**

**for(**inti **=** 0**;** i **<** n**; ++**i**) {**

int p **=** lower\_bound**(**dp**,** dp **+** n**,** a**[**i**]) -** dp**;**

5

dp**[**p**] =** a**[**i**];**

mp**[**a**[**i**]] =** i**;**

**if(**p **>** 0**)**par**[**i**] =** mp**[**dp**[**p **-** 1**]];**

v**[**p**].**push\_back**(**i**);**

**}**

int len **=** lower\_bound**(**dp**,** dp **+** n**,** 0x3f3f3f3f**) -** dp**;**

printf**(**"%d\n"**,** len**);**

print**(**len **-** 1**,** 0**);**

printf**(**"\n\n"**);**

**}**

**return** 0**;**

**}**

（4）NlogN 时间求下标以 i 结尾的最大长度：线段树维护区间最大值优化朴素 dp

**3、康托展开**

#include<bits/stdc++.h>

**using namespace** std**;**

const int facts**[] = {**1**,** 1**,** 2**,** 6**,** 24**,** 120**,** 720**,** 5040**,** 40320**,** 362880**};**

/\*

返回一个排列的字典序排序。[0, n-1]

\*/

int encode**(**vector**<**int**>** seq**) {**

int res **=** 0**,** i**,** j**,** cnt**,** len **=** seq**.**size**();**

**for(**i **=** 0**;** i **<** len**; ++**i**) {**

cnt **=** seq**[**i**] -** 1**;**

**for(**j **=** 0**;** j **<** i**; ++**j**)if(**seq**[**j**] <** seq**[**i**])--**cnt**;** res **+=** cnt **\*** facts**[**len **-** i **-** 1**];**

6

**}**

**return** res**;**

**}**

/\*

在由 1 到 n 组成的前 n 个数的全排列中，找字典序排第 m 的序列。

字典序最小的序列 123...n，排名为 0；

字典序最大的序列 n...321，排名为(n!-1)。

如，在由 1、2、3 组成的序列的全排列中，

序列 123 的排名是 0, 序列 321 的排名是 5。

另外：代码中，facts[i]为 i 的阶乘；

\*/

vector**<**int**>** decode**(**int n**,** int m**) {**

vector**<**int**>** res**;**

long long board **=** 0**;**

int i**,** t**,** r**;**

/\*

注意，如果需要的排名是[1, n]，

即在由 1、2、3 组成的序列的全排列中，要求序列 123 的排名是 1，那么，这里加一句：m--。

\*/

**for(**t **=** n**;** t **>** 0**; --**t**) {**

1. **=** m **/** facts**[**t **-** 1**];** m **%=** facts**[**t **-** 1**];** **for(**i **=** 1**;** i **<=** n**; ++**i**) {**

**if(!((**board **>>** i**) &** 1**)) { if(**r **==** 0**)break; else --**r**;**

**}**

**}**

7

res**.**push\_back**(**i**);**

board **|=** 1 **<<** i**;**

**}**

**return** res**;**

**}**

int main**() {**

**for(**inti **=** 0**;** i **<** 6**; ++**i**) {**

vector**<**int**>** res **=** decode**(**3**,** i**);**

**for(**intj **=** 0**;** j **<** res**.**size**(); ++**j**)**printf**(**"%d "**,** res**[**j**]);** printf**(**"\n"**);**

**}**

vector**<**int**>** v**{**3**,** 2**,** 1**};**

cout **<<** encode**(**v**) <<** endl**;**

**return** 0**;**

**}**

**4、0-1 背包记录路径(以 Installing Apps 题为例)**

#include<bits/stdc++.h>

**using namespace** std**;**

struct app **{**

int d**,** s**,** id**;**

**}** p**[**510**];**

bool cmp**(**app a1**,** app a2**) {**

**return** max**(**a1**.**d**,** a1**.**s**) -** a1**.**s **>** max**(**a2**.**d**,** a2**.**s**) -** a2**.**s**;**

**}**

int N**,** C**,** dp**[**510**][**10010**],** ans**[**510**],** acnt**;**

bool path**[**510**][**10010**];**

8

int main**() {**

scanf**(**"%d%d"**, &**N**, &**C**);**

**for(**inti **=** 1**;** i **<=** N**; ++**i**)**scanf**(**"%d%d"**, &**p**[**i**].**d**, &**p**[**i**].**s**),** p**[**i**].**id **=** i**; for(**inti **=** 0**;** i **<=** N**; ++**i**) {**

**for(**intj **=** 0**;** j **<=** C**; ++**j**)**dp**[**i**][**j**] =** 0**;**

**}**

sort**(**p **+** 1**,** p **+** N **+** 1**,** cmp**);**

**for(**inti **=** 1**;** i **<=** N**; ++**i**) {**

**for(**intj **=** 0**;** j **<=** C**; ++**j**) {**

dp**[**i**][**j**] =** dp**[**i **-** 1**][**j**];**

**if(**j **>=** p**[**i**].**s**) {**

int last **=** j **-** p**[**i**].**s**;**

**if(**C **-** last **>=** p**[**i**].**d**) {**

**if(**dp**[**i**][**j**] <** dp**[**i **-** 1**][**last**] +** 1**) {**

dp**[**i**][**j**] =** dp**[**i **-** 1**][**last**] +** 1**;**

path**[**i**][**j**] =** 1**;**

**}**

**}**

**}**

**}**

**}**

int V**,** aa **=** 0**;**

**for(**intj **=** C**;** j **>=** 0**; --**j**) {**

**if(**aa **<** dp**[**N**][**j**]) {**

aa **=** dp**[**N**][**j**],** V **=** j**;**

**}**

**}**

**for(**inti **=** N**,** j **=** V**;** i **>** 0**; --**i**) {**

**if(**path**[**i**][**j**]) {**

9

ans**[++**acnt**] =** p**[**i**].**id**;**

1. **-=** p**[**i**].**s**;**

**}**

**}**

reverse**(**ans **+** 1**,** ans **+** acnt **+** 1**);**

printf**(**"%d\n"**,** acnt**);**

**for(**inti **=** 1**;** i **<=** acnt**; ++**i**)**printf**(**"%d "**,** ans**[**i**]); return** 0**;**

**}**

**5、多重背包二进制优化转 0-1 背包(以焦作网络赛 Transport Ship 题为例)**

#include<bits/stdc++.h>

**using namespace** std**;**

**typedef** long longLL**;**

const LL mod **=** 1000000007**;**

int v**[**510**],** n**,** q**,** ncnt**;**

1. dp**[**510**][**10010**];** int main**() {**

int T**,** x**,** y**;**

**for(**scanf**(**"%d"**, &**T**);** T**--;) {** ncnt **=** 0**;** scanf**(**"%d%d"**, &**n**, &**q**);**

**for(**inti **=** 1**;** i **<=** n**; ++**i**) {** scanf**(**"%d%d"**, &**x**, &**y**);**

y **= (**1 **<<** y**) -** 1**;**

**for(**inti **=** 1**;** i **<=** y**;** i **<<=** 1**) {**

v**[++**ncnt**] =** i **\*** x**;**

**}**

**}**

10

**for(**inti **=** 0**;** i **<=** ncnt**; ++**i**) {**

**for(**intj **=** 0**;** j **<=** 10000**; ++**j**)**dp**[**i**][**j**] =** 0**;**

**}**

dp**[**0**][**0**] =** 1**;**

**for(**inti **=** 1**;** i **<=** ncnt**; ++**i**) {**

**for(**intj **=** 0**;** j **<=** 10000**; ++**j**) {**

dp**[**i**][**j**] =** dp**[**i **-** 1**][**j**];**

**if(**j **>=** v**[**i**])**dp**[**i**][**j**] = (**dp**[**i**][**j**] +** dp**[**i **-** 1**][**j **-** v**[**i**]]) %** mod**;**

**}**

**}**

**while(**q**--) {**

scanf**(**"%d"**, &**x**);**

printf**(**"%lld\n"**,** dp**[**ncnt**][**x**]);**

**}**

**}**

**return** 0**;**

**}**

**6、斜率优化 dp**

（1）一维斜率优化 dp(以打印单词 hdu3507 那个经典入门题为例)

#include<bits/stdc++.h>

**using namespace** std**;**

#define MAXN 500010

**typedef** long longLL**;**

1. a**[**MAXN**],** s**[**MAXN**],** dp**[**MAXN**];** int n**,** m**;**

11

LL f**(**int x**) {**

**return** dp**[**x**] +** s**[**x**] \*** s**[**x**];**

**}**

1. sq**(**LL x**) {** **return** x **\*** x**;**

**}**

int dq**[**MAXN **\*** 2**];**

int main**() {**

int sz**;**

**while(~**scanf**(**"%d%d"**, &**n**, &**m**)) {**

**for(**inti **=** 1**;** i **<=** n**; ++**i**)**scanf**(**"%lld"**,** a **+** i**),** s**[**i**] =** s**[**i **-** 1**] +** a**[**i**];** memset**(**dp**,** 0x3f**,** **sizeof(**dp**[**0**]) \* (**n **+** 2**));**

dp**[**0**] =** 0**;**

int head **=** 1**,** tail **=** 0**;**

dq**[++**tail**] =** 0**;**

**for(**inti **=** 1**;** i **<=** n**; ++**i**) {**

**while(**tail **-** head **>=** 1 **&& (**f**(**dq**[**head **+** 1**]) -** f**(**dq**[**head**])) < (**s**[**dq**[**head **+** 1**]] -** s**[**dq**[**head**]])\***s**[**i**] \*** 2**)**head**++;**

dp**[**i**] =** dp**[**dq**[**head**]] +** sq**(**s**[**i**] -** s**[**dq**[**head**]]) +** m**;**

**while(**tail **-** head **>=** 1 **&& (**f**(**i**) -** f**(**dq**[**tail**])) \* (**s**[**dq**[**tail**]] -** s**[**dq**[**tail **-** 1**]]) <= (**f**(**dq**[**tail**]) -** f**(**dq**[**tail **-** 1**])) \* (**s**[**i**] -** s**[**dq**[**tail**]]))**tail**--;**

dq**[++**tail**] =** i**;**

**}**

printf**(**"%lld\n"**,** dp**[**n**]);**

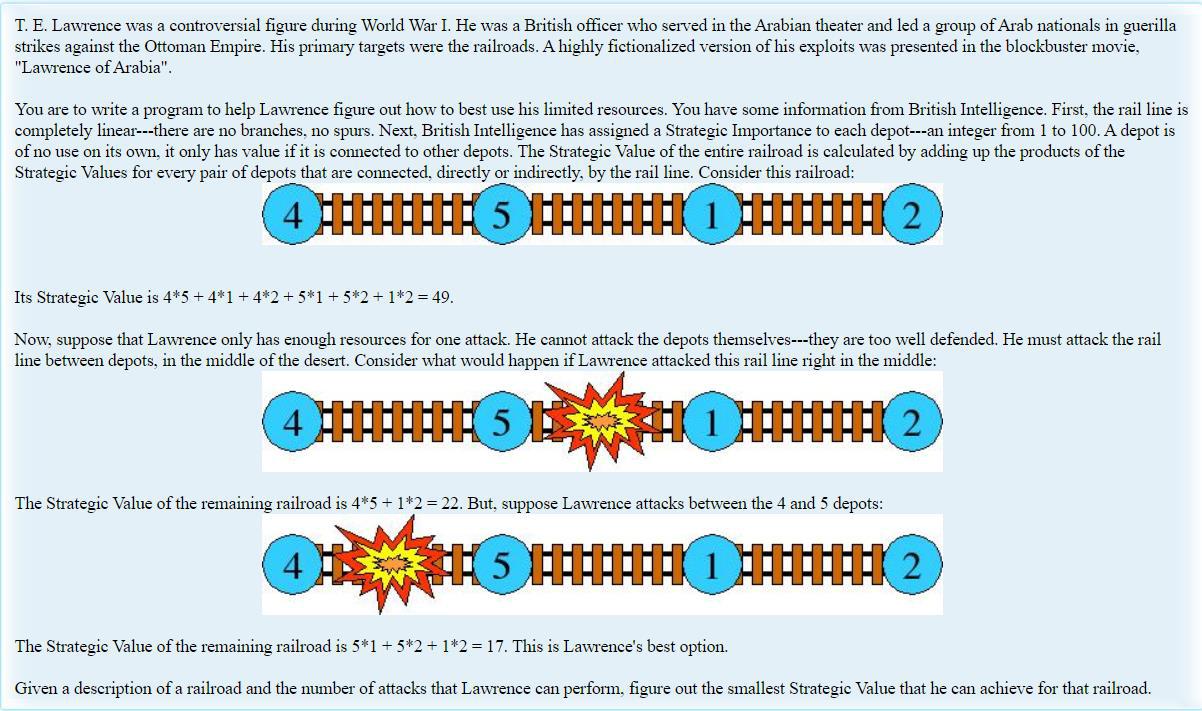
**}**

**return** 0**;**

**}**

（2）二维斜率优化 dp(以炸铁路 hdu2829 那个经典入门题为例)

12



13

/\*

dp[i][j]表示在前 i 个火车站中炸毁 j 处铁路所需的最小代价。

朴素的 dp 式子是：

for(int i=1;i<=n;++i){

for(int j=1;j<=m;++j){

for(int k=1;k<i;++k){

dp[i][j]=min(dp[i][j],dp[k][j-1]+sum[k+1][i]);

}

}

}

斜率优化 dp 注意点：

1、在朴素的 dp 中，i 的循环应该写在 j 的循环外面。但是在斜率优化 dp 里面，如果有多层 for 循环(俗称多维斜率优化 dp))，

单调队列里面的元素的循环应该置于内层，而原来处于内层的应放置在外层。

2、单调队列里面的初始元素，一定是当前“阶段”第一次进入内层循环修改 dp 值的前一个“状态”。以下面的程序为例，单调队列里面的初始元素是 j，也就是说，一定要保证要可以炸毁 j-1 处铁路，至少需要 j 个火车站。3、多维斜率优化 dp，两个 while 循环传递的“阶段”都是上一个。也就是说，以下面的程序为例，如果当前阶段是 j，那么上一个阶段是 j-1,在 f 函数传参的时候传递的是 j-1，不是 j。

4、求的是 dp 的最小值<=，所以是维护下凸包。相反，如果求的是 dp 的最大值>=，维护的是上凸包。

5、单调队列维护完成以后，一定要记得把当前“阶段”下的当前“状态”放到单调队列中。

6、由于 dp 数组的元素不需要取 min，所以，不需要初始化为 inf。

\*\*/

#include<bits/stdc++.h>

**using namespace** std**;**

**typedef** long longLL**;**

#define MAXN 1010

int n**,** m**,** a**[**MAXN**];**

1. pre**[**MAXN**],** s**[**MAXN**],** dp**[**MAXN**][**MAXN**];** int dq**[**2020**];**

14

template**<**class T**>** inline T sq**(**T x**) {**

**return** x **\*** x**;**

**}**

inline LL f**(**int x**,** int j**) {**

**return** 2 **\*** dp**[**x**][**j**] +** sq**(**pre**[**x**]) +** s**[**x**];**

**}**

int main**() {**

* freopen("input.txt", "r", stdin);

**while(**scanf**(**"%d%d"**, &**n**, &**m**),** n **||** m**) {**

**for(**inti **=** 1**;** i **<=** n**; ++**i**)**scanf**(**"%d"**,** a **+** i**),** pre**[**i**] =** pre**[**i **-** 1**] +** a**[**i**],** s**[**i**] =**

s**[**i **-** 1**] +** a**[**i**] \*** a**[**i**];**

**for(**inti **=** 1**;** i **<=** n**; ++**i**)**dp**[**i**][**0**] = (**sq**(**pre**[**i**]) -** s**[**i**]) /** 2**; for(**intj **=** 1**;** j **<=** m**; ++**j**) {**

int head **=** 1**,** tail **=** 0**;**

dq**[++**tail**] =** j**;**

**for(**inti **=** j **+** 1**;** i **<=** n**; ++**i**) {**

**while(**tail **-** head **>=** 1 **&&** f**(**dq**[**head **+** 1**],** j **-** 1**) -** f**(**dq**[**head**],** j **-** 1**) <= (**pre**[**dq**[**head **+** 1**]] -** pre**[**dq**[**head**]]) \*** 2 **\*** pre**[**i**])**head**++;**

dp**[**i**][**j**] =** dp**[**dq**[**head**]][**j **-** 1**] + (**sq**(**pre**[**i**] -** pre**[**dq**[**head**]]) - (**s**[**i**] -** s**[**dq**[**head**]])) /** 2**;**

**while(**tail **-** head **>=** 1 **&&**

**(**f**(**i**,** j **-** 1**) -** f**(**dq**[**tail**],** j **-** 1**)) \* (**pre**[**dq**[**tail**]] -** pre**[**dq**[**tail **-**

1**]]) <=**

**(**f**(**dq**[**tail**],** j **-** 1**) -** f**(**dq**[**tail **-** 1**],** j **-** 1**)) \* (**pre**[**i**] -** pre**[**dq**[**tail**]]))**tail**--;**

dq**[++**tail**] =** i**;**

**}**

**}**

printf**(**"%lld\n"**,** dp**[**n**][**m**]);**

15

**}**

**return** 0**;**

**}**

**7、三分**

（1）整数

#include<bits/stdc++.h>

**using namespace** std**;**

/\*

上突。

\*\*/

int fd3**(**int a**[],** int l**,** int r**) {**

int lmid**,** rmid**,** low **=** l**,** high **=** r**,** res **= -**1**;** **while(**low **<** high**) {**

lmid **= (**low **+** high**) /** 2**;**

rmid **= (**lmid **+** high**) /** 2**;**

**if(**a**[**lmid**] >** a**[**rmid**])**high **=** rmid**;**

**else if(**a**[**lmid**] ==** a**[**rmid**])**low **=** lmid**,** high **=** rmid**; else** low **=** lmid**;**

**}**

res **=** low**;**

**if(**a**[**res**] <** a**[**high**])**res **=** high**;**

**if(**a**[**res**] <** a**[**l**])**res **=** l**;**

**if(**a**[**res**] <** a**[**r**])**res **=** r**;**

**return** res**;**

**}**

16

|  |  |  |
| --- | --- | --- |
| int main**() {** |  |  |
| int a**[] =** | **{**0**,** 8**,** | 7**,** 6**,** 6**,** 5**,** 5**,** 4**,** 2**,** 2**,** 1**},** n **=** **sizeof(**a**) /** **sizeof(**a**[**0**]) -** 1**;** |
| int idx **=** | fd3**(**a**,** | 1**,** n**);** |
| **for(**inti | **=** 1**;** i | **<=** n**; ++**i**)**printf**(**"%d "**,** a**[**i**]);** printf**(**"\n"**);** |
| printf**(**"##%d\n"**,** | | a**[**idx**]);** |
| **return** 0**;** |  |  |
| **}** |  |  |

（2）分数

#include<bits/stdc++.h>

**using namespace** std**;**

/\*

上突。

\*\*/

int main**(){**

double low**=**下界**,**high**=**上界**,**lmid**,**rmid**;**

**while(**low**<**high**-**1e-8**){**

lmid**=(**low**+**high**)/**2**;**

rmid**=(**lmid**+**high**)/**2**;**

**if(**calc**(**lmid**)>**calc**(**rmid**))**high**=**rmid**;**

**else** low**=**lmid**;**

**}**

double ans**=**max**(**calc**(**low**),**calc**(**high**));**

**return** 0**;**

**}**

**二、数据结构**

17

**1、LCA**

（1）二分搜索版

#include<bits/stdc++.h>

**using namespace** std**;**

const int MAXN **=** 100010**;**

const int LOGN **=** 20**;**

struct edge **{**

int to**,** next**;**

edge**(**int to **=** 0**,** int next **=** 0**):** to**(**to**),** next**(**next**) {}**

**}** es**[**MAXN **\*** 2**];**

int head**[**MAXN**],** ecnt**,** n**,** m**;**

int parent**[**MAXN**][**LOGN**],** deep**[**MAXN**];**

void add**(**int from**,** int to**) {**

es**[++**ecnt**] =** edge**(**to**,** head**[**from**]),** head**[**from**] =** ecnt**;**

**}**

void init**() {**//确保在 n 输入之后调用

memset**(**head**,** 0**,** **sizeof(**head**[**0**]) \* (**n **+** 5**));**

ecnt **=** 0**;**

**}**

void dfs4lca**(**int root**,** int par**,** int d**) {**

parent**[**root**][**0**] =** par**,** deep**[**root**] =** d**;**

**for(**inti **=** head**[**root**];** i **!=** 0**;** i **=** es**[**i**].**next**) {** int to **=** es**[**i**].**to**;**

**if(**to **!=** par**) {**

dfs4lca**(**to**,** root**,** d **+** 1**);**

18

**}**

**}**

**}**

void init4lca**() {**

dfs4lca**(**1**,** 0**,** 1**);**

**for(**intj **=** 0**;** j **+** 1 **<** LOGN**; ++**j**) {**

**for(**inti **=** 1**;** i **<=** n**; ++**i**) {**

**if(**parent**[**i**][**j**] ==** 0**)**parent**[**i**][**j **+** 1**] =** 0**;**

**else** parent**[**i**][**j **+** 1**] =** parent**[**parent**[**i**][**j**]][**j**];**

**}**

**}**

**}**

int lca**(**int a**,** int b**) {**

**if(**deep**[**a**] <** deep**[**b**])**swap**(**a**,** b**);**

**for(**inti **=** 0**;** i **<** LOGN**; ++**i**) {**

**if(((**deep**[**a**] -** deep**[**b**]) >>** i**) &** 1**)**a **=** parent**[**a**][**i**];**

**}**

**if(**a **==** b**)return** a**;**

**for(**inti **=** LOGN **-** 1**;** i **>=** 0**; --**i**) {**

**if(**parent**[**a**][**i**] !=** parent**[**b**][**i**]) {**

a **=** parent**[**a**][**i**],** b **=** parent**[**b**][**i**];**

**}**

**}**

**return** parent**[**a**][**0**];**

**}**

（2）tarjan 版

19

#include<iostream>

#include<cstdio>

#include<cstring>

#include<vector>

**using namespace** std**;**

const int MAXN **=** 1010**;**

**typedef** struct **{**

int to**,** next**;**

**}** Edge**;**

Edge tree**[**MAXN **\*** 2**];**

int head**[**MAXN**],** cnt**;**

**typedef** structS1 **{**

int bro**;**

int anc**;**//和 to 的祖先

S1**():** bro**(**0**),** anc**(**0**) {}**

S1**(**int arg1**,** int arg2**) {**

bro **=** arg1**;**

anc **=** arg2**;**

**}**

**}** Node**;**

vector**<**Node**>** query**[**MAXN**];**

bool vis**[**MAXN**];**

int n**,** m**;**

/\*并查集部分\*/

int f**[**MAXN**];**

int lookup**(**int x**) {**

**if(**f**[**x**] ==** x**)return** x**;**

**else return** f**[**x**] =** lookup**(**f**[**x**]);**

**}**

20

void unite**(**int root**,** int to**) {**

**if(**lookup**(**root**) ==** lookup**(**to**))return;**

**else** f**[**to**] =** root**;**

**}**

/\*并查集部分\*/

void add**(**int from**,** int to**) {**

tree**[**cnt**].**to **=** to**;**

tree**[**cnt**].**next **=** head**[**from**];**

head**[**from**] =** cnt**++;**

**}**

void init**() {**

memset**(**head**, -**1**,** **sizeof(**head**));**

memset**(**vis**,** 0**,** **sizeof(**vis**));**

**for(**inti **=** 0**;** i **<** MAXN**; ++**i**)**query**[**i**].**clear**();** cnt **=** 0**;**

/\*并查集部分\*/

**for(**inti **=** 1**;** i **<=** MAXN**; ++**i**)**f**[**i**] =** i**;**

/\*并查集部分\*/

**}**

void tarjan\_lca**(**int root**) {**

**for(**inti **=** head**[**root**],** to **= -**1**;** i **!= -**1**;** i **=** tree**[**i**].**next**) {** to **=** tree**[**i**].**to**;**

tarjan\_lca**(**to**);**

unite**(**root**,** to**);**

vis**[**to**] =** **true;**

21

**}**

**for(**inti **=** 0**,** sz **=** query**[**root**].**size**(),** bro **= -**1**;** i **<** sz**; ++**i**) {**

bro **=** query**[**root**][**i**].**bro**;**

**if(**vis**[**bro**])**query**[**root**][**i**].**anc **=** lookup**(**bro**);**

**}**

**}**

int main**() {**

freopen**(**"input.txt"**,** "r"**,** stdin**);**

int a**,** b**;**

scanf**(**"%d%d"**, &**n**, &**m**);**

init**();**

**for(**inti **=** 1**;** i **<** n**; ++**i**) {** //construct tree.

scanf**(**"%d%d"**, &**a**, &**b**);**

add**(**a**,** b**);**

**}**

**for(**inti **=** 1**;** i **<=** m**; ++**i**) {**//m times queries.

scanf**(**"%d%d"**, &**a**, &**b**);**

query**[**a**].**push\_back**(**Node**(**b**, -**1**));**

query**[**b**].**push\_back**(**Node**(**a**, -**1**));**

**}**

tarjan\_lca**(**1**);**

**for(**inti **=** 1**;**i **<=** n**;++**i**){**

**for(**intj **=** 0**,** sz **=** query**[**i**].**size**();**j **<** sz**;++**j**){ if(**query**[**i**][**j**].**anc **== -**1**)continue;**

printf**(**"LCA(%d, %d) = %d\n"**,** i**,** query**[**i**][**j**].**bro**,** query**[**i**][**j**].**anc**);**

**}**

22

**}**

**return** 0**;**

**}**

**2、并查集**

#define MAXN 100010

**typedef** struct **{**

int par**[**MAXN**],** rk**[**MAXN**];**

void init**() {**

**for(**inti **=** 0**;** i **<** MAXN**; ++**i**)**par**[**i**] =** i**,** rk**[**i**] =** 0**;**

**}**

int root**(**int v**) {**

**return** v **==** par**[**v**] ?** v **:** par**[**v**] =** root**(**par**[**v**]);**

**}**

bool same**(**int u**,** int v**) {**

**return** root**(**u**) ==** root**(**v**);**

**}**

void unite**(**int u**,** int v**) {**

int x **=** root**(**u**),** y **=** root**(**v**);**

**if(**x **==** y**)return;**

**if(**rk**[**x**] <** rk**[**y**])**par**[**x**] =** y**;**

**else {**

par**[**y**] =** x**;**

**if(**rk**[**x**] ==** rk**[**y**])**rk**[**x**]++;**

**}**

23

**}**

**}** UniFinder**;**

**3、树状数组**

（1）一维区间更新单点查询

#include<cstring>

#include<cstdio>

#define MAXN 50010

**using namespace** std**;**

**typedef** long longLL**;**

int n**,** m**;**

LL bit**[**MAXN**];**

LL query**(**int x**) {**

LL res **=** 0**;**

**for(**inti **=** x**;** i **>** 0**;** i **-=** i **& -**i**)**res **+=** bit**[**i**]; return** res**;**

**}**

void update**(**int x**,** int val**) {**

**for(**inti **=** x**;** i **<=** n**;** i **+=** i **& -**i**)**bit**[**i**] +=** val**;**

**}**

int main**() {**

24

freopen**(**"input.txt"**,** "r"**,** stdin**);**

freopen**(**"output2.txt"**,** "w"**,** stdout**);**

int a**,** b**,** v**;**

memset**(**bit**,** 0**,** **sizeof(**bit**));**

scanf**(**"%d%d"**, &**n**, &**m**);**

**for(**inti **=** 1**;**i **<=** n**;++**i**){**

scanf**(**"%d"**, &**v**);**

update**(**i**,** v**);**

update**(**i **+** 1**, -**v**);**

**}**

**for(**inti **=** 1**;**i **<=** m**;++**i**){**

scanf**(**"%d%d%d"**, &**a**, &**b**, &**v**);**

update**(**a**,** v**);**

update**(**b **+** 1**, -**v**);**

**}**

**for(**inti **=** 1**;**i **<=** n**;++**i**)**printf**(**"%I64d\n"**,** query**(**i**));**

**return** 0**;**

**}**

（2）一维区间更新区间查询

#include<iostream>

#include<cstdio>

#include<cstring>

#define MAXN 50010

**using namespace** std**;**

25

**typedef** long longLL**;**

int n**,** m**,** q**;**

LL d**[**MAXN**],** f**[**MAXN**];**

LL query**(**LL**\*** arr**,** int x**) {**

LL res **=** 0**;**

**for(**inti **=** x**;** i **>** 0**;** i **-= (**i **& -**i**))**res **+=** arr**[**i**]; return** res**;**

**}**

void update**(**LL**\*** arr**,** int x**,** LL val**) {**

**for(**inti **=** x**;** i **<=** n**;** i **+= (**i **& -**i**))**arr**[**i**] +=** val**;**

**}**

inline LL getsum**(**int x**) {**

**return (**x **+** 1**) \*** query**(**d**,** x**) -** query**(**f**,** x**);**

**}**

int main**() {**

freopen**(**"input.txt"**,** "r"**,** stdin**);**

freopen**(**"output2.txt"**,** "w"**,** stdout**);**

memset**(**d**,** 0**,** **sizeof(**d**));**

memset**(**f**,** 0**,** **sizeof(**f**));**

scanf**(**"%d%d%d"**, &**n**, &**m**, &**q**);**

int x**,** y**;**

LL v**,** v0 **=** 0**;**

26

**for(**inti **=** 1**;** i **<=** n**; ++**i**) {**

scanf**(**"%I64d"**, &**v**);**

int t **=** v **-** v0**;**

update**(**d**,** i**,** t**);**

update**(**f**,** i**,** t **\*** i**);**

//这里要注意：不能多写。

v0 **=** v**;**

**}**

**for(**inti **=** 1**;** i **<=** m**; ++**i**) {**

scanf**(**"%d%d%I64d"**, &**x**, &**y**, &**v**);**

update**(**d**,** x**,** v**);**

update**(**d**,** y **+** 1**, -**v**);**

update**(**f**,** x**,** v **\*** x**);**

update**(**f**,** y **+** 1**, -**v **\* (**y **+** 1**));**

**}**

**for(**inti **=** 1**;** i **<=** q**; ++**i**) {**

scanf**(**"%d%d"**, &**x**, &**y**);**

printf**(**"%I64d\n"**,** getsum**(**y**) -** getsum**(**x **-** 1**));**

**}**

**return** 0**;**

**}**

（3）二维树状数组区间更新区间查询

#include<bits/stdc++.h>

**using namespace** std**;**

struct bit2d **{**

27

**typedef** long longLL**;**

#define MAXN 2050

1. b**[**MAXN**][**MAXN**],** bi**[**MAXN**][**MAXN**],** bj**[**MAXN**][**MAXN**],** bij**[**MAXN**][**MAXN**];** int N**,** M**;**

void init**(**int n**,** int m**){** N **=** n**,** M **=** m**;**

**}**

void clear**(){**

memset**(**b**,** 0**,** **sizeof(**b**));** memset**(**bi**,** 0**,** **sizeof(**bi**));** memset**(**bj**,** 0**,** **sizeof(**bj**));** memset**(**bij**,** 0**,** **sizeof(**bij**));**

**}**

private**:**

void add0**(**LL a**[][**MAXN**],** int x**,** int y**,** LL val**) {** **for(**inti **=** x**;** i **<=** N**;** i **+=** i **& -**i**) {**

**for(**intj **=** y**;** j **<=** M**;** j **+=** j **& -**j**)**a**[**i**][**j**] +=** val**;**

**}**

**}**

private**:**

void add**(**int x**,** int y**,** LL val**) {**

add0**(**b**,** x**,** y**,** val**);**

add0**(**bi**,** x**,** y**,** 1LL **\*** x **\*** val**);**

add0**(**bj**,** x**,** y**,** 1LL **\*** y **\*** val**);**

add0**(**bij**,** x**,** y**,** 1LL **\*** x **\*** y **\*** val**);**

**}**

public**:**

void update**(**int x1**,** int y1**,** int x2**,** int y2**,** LL val**) {** add**(**x1**,** y1**,** val**);**

28

add**(**x1**,** y2 **+** 1**, -**val**);**

add**(**x2 **+** 1**,** y1**, -**val**);**

add**(**x2 **+** 1**,** y2 **+** 1**,** val**);**

**}**

private**:**

1. sum0**(**LL a**[][**MAXN**],** LL x**,** LL y**) {** LL res **=** 0**;**

**for(**inti **=** x**;** i **>** 0**;** i **-=** i **& -**i**) {**

**for(**intj **=** y**;** j **>** 0**;** j **-=** j **& -**j**)**res **+=** a**[**i**][**j**];**

**}**

**return** res**;**

**}**

private**:**

LL sum**(**int x**,** int y**) {**

**return** sum0**(**b**,** x**,** y**) \* (**x **\*** y **+** x **+** y **+** 1**)**

* sum0**(**bi**,** x**,** y**) \* (**y **+** 1**)**
* sum0**(**bj**,** x**,** y**) \* (**x **+** 1**)** **+** sum0**(**bij**,** x**,** y**);**

**}**

public**:**

LL query**(**int x1**,** int y1**,** int x2**,** int y2**) {**

**return** sum**(**x2**,** y2**) -** sum**(**x2**,** y1 **-** 1**) -** sum**(**x1 **-** 1**,** y2**) +** sum**(**x1 **-** 1**,** y1 **-** 1**);**

**}**

**};**

bit2d bd**;**

int n**,** m**;**

int main**() {**

// freopen("input.txt", "r", stdin);

29

int x1**,** y1**,** x2**,** y2**,** val**;**

char op**[**5**];**

scanf**(**"%s%d%d"**,** op**, &**n**, &**m**);**

bd**.**init**(**n**,** m**);**

**while(~**scanf**(**"%s"**,** op**)) {**

scanf**(**"%d%d%d%d"**, &**x1**, &**y1**, &**x2**, &**y2**);**

**if(**op**[**0**] ==** 'L'**) {**

scanf**(**"%d"**, &**val**);**

bd**.**update**(**x1**,** y1**,** x2**,** y2**,** val**);**

**}**

**else** printf**(**"%lld\n"**,** bd**.**query**(**x1**,** y1**,** x2**,** y2**));**

**}**

**return** 0**;**

**}**

**4、可持久化线段树(主席树)**

（1）只可查询区间第 k 小、不可修改（时间复杂度： + ，空间复杂度： + ）

/\*\*

只支持一棵线段树一个值。

如果要实现一棵线段树插入多个值，需要修改 update0 函数的 if。

\*\*/

struct PerSegTree **{**

#define MAXN 100010

int N**,** M**,** root**[**MAXN**],** ncnt**;**

struct seg **{**

int lch**,** rch**,** cnt**;**

**}** segs**[**MAXN **\*** 20**];**

30

public**:** //an:线段树的数量；am:线段树叶子节点的数量； void init**(**int an**,** int am**) {**

N **=** an**,** M **=** am**;**

ncnt **=** 0**;**

memset**(**root**,** 0**,** **sizeof(**root**[**0**]) \* (**an **+** 3**));**

**}**

private**:**

void update0**(**int**&** croot**,** int proot**,** int val**,** int l**,** int r**) {** **if(**croot **==** 0**) {**

croot **= ++**ncnt**;**

segs**[**croot**].**cnt **=** segs**[**proot**].**cnt **+** 1**;**

segs**[**croot**].**lch **=** segs**[**croot**].**rch **=** 0**;**

**}**

**if(**l **==** r**)return;**

int mid **= (**l **+** r**) >>** 1**;**

**if(**val **<=** mid**) {**

segs**[**croot**].**rch **=** segs**[**proot**].**rch**;**

update0**(**segs**[**croot**].**lch**,** segs**[**proot**].**lch**,** val**,** l**,** mid**);**

**}**

**else {**

segs**[**croot**].**lch **=** segs**[**proot**].**lch**;**

update0**(**segs**[**croot**].**rch**,** segs**[**proot**].**rch**,** val**,** mid **+** 1**,** r**);**

**}**

**}**

public**:**

void insert**(**int x**,** int val**) {**

update0**(**root**[**x**],** root**[**x **-** 1**],** val**,** 1**,** M**);**

**}**

private**:**

31

int query0**(**int qlr**,** int qrr**,** int k**,** int l**,** int r**) {** **if(**l **==** r**)return** l**;**

int mid **= (**l **+** r**) >>** 1**,** tmp **=** segs**[**segs**[**qrr**].**lch**].**cnt **-** segs**[**segs**[**qlr**].**lch**].**cnt**;** **if(**k **<=** tmp**) {**

**return** query0**(**segs**[**qlr**].**lch**,** segs**[**qrr**].**lch**,** k**,** l**,** mid**);**

**}**

**else {**

**return** query0**(**segs**[**qlr**].**rch**,** segs**[**qrr**].**rch**,** k **-** tmp**,** mid **+** 1**,** r**);**

**}**

**}**

public**:**

int query**(**int ql**,** int qr**,** int k**) {**

**return** query0**(**root**[**ql **-** 1**],** root**[**qr**],** k**,** 1**,** M**);**

**}**

**}** pst**;**

（2）例题(POJ2104)

#include<bits/stdc++.h>

**using namespace** std**;**

/\*\*

只支持一棵线段树一个值。

如果要实现一棵线段树插入多个值，需要修改 update0 函数的 if。

\*\*/

struct PerSegTree **{**

#define MAXN 100010

int N**,** M**,** root**[**MAXN**],** ncnt**;**

struct seg **{**

int lch**,** rch**,** cnt**;**

32

**}** segs**[**MAXN **\*** 20**];**

public**:** //an:线段树的数量；am:线段树叶子节点的数量； void init**(**int an**,** int am**) {**

N **=** an**,** M **=** am**;**

ncnt **=** 0**;**

memset**(**root**,** 0**,** **sizeof(**root**[**0**]) \* (**an **+** 3**));**

**}**

private**:**

void update0**(**int**&** croot**,** int proot**,** int val**,** int l**,** int r**) {** **if(**croot **==** 0**) {**

croot **= ++**ncnt**;**

segs**[**croot**].**cnt **=** segs**[**proot**].**cnt **+** 1**;**

segs**[**croot**].**lch **=** segs**[**croot**].**rch **=** 0**;**

**}**

**if(**l **==** r**)return;**

int mid **= (**l **+** r**) >>** 1**;**

**if(**val **<=** mid**) {**

segs**[**croot**].**rch **=** segs**[**proot**].**rch**;**

update0**(**segs**[**croot**].**lch**,** segs**[**proot**].**lch**,** val**,** l**,** mid**);**

**}**

**else {**

segs**[**croot**].**lch **=** segs**[**proot**].**lch**;**

update0**(**segs**[**croot**].**rch**,** segs**[**proot**].**rch**,** val**,** mid **+** 1**,** r**);**

**}**

**}**

public**:**

void insert**(**int x**,** int val**) {**

update0**(**root**[**x**],** root**[**x **-** 1**],** val**,** 1**,** M**);**

**}**

33

private**:**

int query0**(**int qlr**,** int qrr**,** int k**,** int l**,** int r**) {** **if(**l **==** r**)return** l**;**

int mid **= (**l **+** r**) >>** 1**,** tmp **=** segs**[**segs**[**qrr**].**lch**].**cnt **-** segs**[**segs**[**qlr**].**lch**].**cnt**;** **if(**k **<=** tmp**) {**

**return** query0**(**segs**[**qlr**].**lch**,** segs**[**qrr**].**lch**,** k**,** l**,** mid**);**

**}**

**else {**

**return** query0**(**segs**[**qlr**].**rch**,** segs**[**qrr**].**rch**,** k **-** tmp**,** mid **+** 1**,** r**);**

**}**

**}**

public**:**

int query**(**int ql**,** int qr**,** int k**) {**

**return** query0**(**root**[**ql **-** 1**],** root**[**qr**],** k**,** 1**,** M**);**

**}**

**}** pst**;**

int a**[**MAXN**],** b**[**MAXN**],** n**,** q**;**

int main**() {**

freopen**(**"a.in"**,** "r"**,** stdin**);**

freopen**(**"a2.out"**,** "w"**,** stdout**);**

int T**;**

**for(**scanf**(**"%d"**, &**T**);** T**--;) {**

scanf**(**"%d%d"**, &**n**, &**q**);**

**for(**inti **=** 1**;** i **<=** n**; ++**i**)**scanf**(**"%d"**,** a **+** i**),** b**[**i**] =** a**[**i**];** sort**(**b **+** 1**,** b **+** n **+** 1**);**

int m **=** unique**(**b **+** 1**,** b **+** n **+** 1**) -** b **-** 1**;**

pst**.**init**(**n**,** m**);**

**for(**inti **=** 1**;** i **<=** n**; ++**i**)**pst**.**insert**(**i**,** lower\_bound**(**b **+** 1**,** b **+** m **+** 1**,** a**[**i**]) -**

b**);**

34

**while(**q**--) {**

int l**,** r**,** k**;**

scanf**(**"%d%d%d"**, &**l**, &**r**, &**k**);**

printf**(**"%d\n"**,** b**[**pst**.**query**(**l**,** r**,** k**)]);**

**}**

**}**

**return** 0**;**

**}**

（3）区间查询第 k 小，可单点修改（时间复杂度： + 2 ，空间复杂度： + 2 ）

#define MAXN 50010

#define MAXQ 10010

/\*

n:数字序列长度；

q:操作的个数；

m:对 b 去重后的数字个数；

a:输入序列；

b:输入序列+更新数字的集合。要排序，用于离散化。

\*/

int n**,** q**,** m**,** a**[**MAXN**],** b**[**MAXN **+** MAXQ**];**

/\*操作需要先全部记录下来\*/

struct oper **{**

int type**;**

int l**,** r**,** k**;**

int x**,** val**;**

35

**}** p**[**MAXQ**];**

/\*先 insert，再 add\*/

**typedef** struct **{**

int root0**[**MAXN**],** root1**[**MAXN**],** qrt**[**MAXN**],** ncnt**;** struct seg **{**

int lch**,** rch**,** cnt**;**

**}** segs**[**MAXN **\*** 16 **+** MAXQ **\*** 256**];**

/\*MAXN \* log2(MAXN) + MAXQ \* log2(MAXN) \* log2(MAXN)\*/ public**:**

/\*通常情况下，传递参数为 n。\*/

void init**(**int size**) {**

ncnt **=** 0**;**

memset**(**root0**,** 0**,** **sizeof(**root0**[**0**]) \* (**size **+** 1**));** memset**(**root1**,** 0**,** **sizeof(**root0**[**1**]) \* (**size **+** 1**));** segs**[**0**].**lch **=** segs**[**0**].**rch **=** segs**[**0**].**cnt **=** 0**;**

**}**

private**:**

int update0**(**int proot**,** int val**,** int d**,** int l **=** 1**,** int r **=** m**) {** int nroot **= ++**ncnt**;**

segs**[**nroot**].**cnt **=** segs**[**proot**].**cnt **+** d**;**

**if(**l **==** r**)return** nroot**;**

int mid **= (**l **+** r**) >>** 1**;**

**if(**val **<=** mid**) {**

segs**[**nroot**].**rch **=** segs**[**proot**].**rch**;**

segs**[**nroot**].**lch **=** update0**(**segs**[**proot**].**lch**,** val**,** d**,** l**,** mid**);**

36

**}**

**else {**

segs**[**nroot**].**lch **=** segs**[**proot**].**lch**;**

segs**[**nroot**].**rch **=** update0**(**segs**[**proot**].**rch**,** val**,** d**,** mid **+** 1**,** r**);**

**}**

**return** nroot**;**

**}**

public**:**

void add**(**int x**,** int val**,** int d**) {**

**for(**inti **=** x**;** i **<=** n**;** i **+= (**i **& -**i**))**root1**[**i**] =** update0**(**root1**[**i**],** val**,** d**);**

**}**

public**:**

/\*用法：insert(i, a[i])\*/

void insert**(**int x**,** int val**){**

root0**[**x**] =** update0**(**root0**[**x **-** 1**],** val**,** 1**);**

**}**

private**:**

int getsum**(**int x**){**

int res **=** 0**;**

**for(**inti **=** x**;**i **>** 0**;**i **-= (**i **& -**i**))**res **+=** segs**[**segs**[**qrt**[**i**]].**lch**].**cnt**;**

**return** res**;**

**}**

private**:**

int query0**(**int ql**,** int qr**,** int rl**,** int rr**,** int k**,** int l **=** 1**,** int r **=** m**) {**

**if(**l **==** r**)return** l**;**

37

int sum **=** getsum**(**qr**) -** getsum**(**ql **-** 1**) +** segs**[**segs**[**rr**].**lch**].**cnt **-** segs**[**segs**[**rl**].**lch**].**cnt**,** mid **= (**l **+** r**) >>** 1**;**

**if(**k **<=** sum**) {**

**for(**inti **=** ql **-** 1**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** segs**[**qrt**[**i**]].**lch**; for(**inti **=** qr**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** segs**[**qrt**[**i**]].**lch**; return** query0**(**ql**,** qr**,** segs**[**rl**].**lch**,** segs**[**rr**].**lch**,** k**,** l**,** mid**);**

**}**

**else {**

**for(**inti **=** ql **-** 1**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** segs**[**qrt**[**i**]].**rch**;**

**for(**inti **=** qr**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** segs**[**qrt**[**i**]].**rch**;**

**return** query0**(**ql**,** qr**,** segs**[**rl**].**rch**,** segs**[**rr**].**rch**,** k **-** sum**,** mid **+** 1**,** r**);**

**}**

**}**

public**:**

int query**(**int ql**,** int qr**,** int k**) {**

**for(**inti **=** ql **-** 1**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** root1**[**i**]; for(**inti **=** qr**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** root1**[**i**]; return** query0**(**ql**,** qr**,** root0**[**ql **-** 1**],** root0**[**qr**],** k**);**

**}**

**}** ModPerSegTree**;**

（4）例题(ZOJ2112)

#pragma comment(linker, "/STACK:102400000,102400000")

#include<bits/stdc++.h>

**using namespace** std**;**

#define MAXN 50010

38

#define MAXQ 10010

int n**,** q**,** m**,** a**[**MAXN**],** b**[**MAXN **+** MAXQ**];**

struct oper **{**

int type**;**

int l**,** r**,** k**;**

int x**,** val**;**

**}** p**[**MAXQ**];**

**typedef** struct **{**

int root0**[**MAXN**],** root1**[**MAXN**],** qrt**[**MAXN**],** ncnt**;**

struct seg **{**

int lch**,** rch**,** cnt**;**

**}** segs**[**MAXN **\*** 16 **+** MAXQ **\*** 160**];** public**:**

void init**(**int size**) {** ncnt **=** 0**;**

memset**(**root0**,** 0**,** **sizeof(**root0**[**0**]) \* (**size **+** 1**));** memset**(**root1**,** 0**,** **sizeof(**root0**[**1**]) \* (**size **+** 1**));** segs**[**0**].**lch **=** segs**[**0**].**rch **=** segs**[**0**].**cnt **=** 0**;**

**}**

private**:**

int update0**(**int proot**,** int val**,** int d**,** int l **=** 1**,** int r **=** m**) {** int nroot **= ++**ncnt**;**

segs**[**nroot**].**cnt **=** segs**[**proot**].**cnt **+** d**;** **if(**l **==** r**)return** nroot**;**

39

int mid **= (**l **+** r**) >>** 1**;**

**if(**val **<=** mid**) {**

segs**[**nroot**].**rch **=** segs**[**proot**].**rch**;**

segs**[**nroot**].**lch **=** update0**(**segs**[**proot**].**lch**,** val**,** d**,** l**,** mid**);**

**}**

**else {**

segs**[**nroot**].**lch **=** segs**[**proot**].**lch**;**

segs**[**nroot**].**rch **=** update0**(**segs**[**proot**].**rch**,** val**,** d**,** mid **+** 1**,** r**);**

**}**

**return** nroot**;**

**}**

public**:**

void add**(**int x**,** int val**,** int d**) {**

**for(**inti **=** x**;** i **<=** n**;** i **+= (**i **& -**i**))**root1**[**i**] =** update0**(**root1**[**i**],** val**,** d**);**

**}**

public**:**

void insert**(**int x**,** int val**){**

root0**[**x**] =** update0**(**root0**[**x **-** 1**],** val**,** 1**);**

**}**

private**:**

int getsum**(**int x**){**

int res **=** 0**;**

**for(**inti **=** x**;**i **>** 0**;**i **-= (**i **& -**i**))**res **+=** segs**[**segs**[**qrt**[**i**]].**lch**].**cnt**;**

**return** res**;**

**}**

private**:**

40

int query0**(**int ql**,** int qr**,** int rl**,** int rr**,** int k**,** int l **=** 1**,** int r **=** m**) {**

**if(**l **==** r**)return** l**;**

int sum **=** getsum**(**qr**) -** getsum**(**ql **-** 1**) +** segs**[**segs**[**rr**].**lch**].**cnt **-** segs**[**segs**[**rl**].**lch**].**cnt**,** mid **= (**l **+** r**) >>** 1**;**

**if(**k **<=** sum**) {**

**for(**inti **=** ql **-** 1**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** segs**[**qrt**[**i**]].**lch**; for(**inti **=** qr**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** segs**[**qrt**[**i**]].**lch**; return** query0**(**ql**,** qr**,** segs**[**rl**].**lch**,** segs**[**rr**].**lch**,** k**,** l**,** mid**);**

**}**

**else {**

**for(**inti **=** ql **-** 1**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** segs**[**qrt**[**i**]].**rch**;**

**for(**inti **=** qr**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** segs**[**qrt**[**i**]].**rch**;**

**return** query0**(**ql**,** qr**,** segs**[**rl**].**rch**,** segs**[**rr**].**rch**,** k **-** sum**,** mid **+** 1**,** r**);**

**}**

**}**

public**:**

int query**(**int ql**,** int qr**,** int k**) {**

**for(**inti **=** ql **-** 1**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** root1**[**i**]; for(**inti **=** qr**;** i **>** 0**;** i **-= (**i **& -**i**))**qrt**[**i**] =** root1**[**i**]; return** query0**(**ql**,** qr**,** root0**[**ql **-** 1**],** root0**[**qr**],** k**);**

**}**

**}** ModPerSegTree**;**

ModPerSegTree mpst**;**

int main**() {**

freopen**(**"input.txt"**,** "r"**,** stdin**);**

41

freopen**(**"output5.txt"**,** "w"**,** stdout**);**

int T**;**

char cmd**[**5**];**

**for(**scanf**(**"%d"**, &**T**);** T**--;) {**

m **=** 0**;**

scanf**(**"%d%d"**, &**n**, &**q**);**

mpst**.**init**(**n**);**

**for(**inti **=** 1**;** i **<=** n**; ++**i**)**scanf**(**"%d"**,** a **+** i**),** b**[++**m**] =** a**[**i**]; for(**inti **=** 1**;** i **<=** q**; ++**i**) {**

scanf**(**"%s"**,** cmd**);**

**if(**cmd**[**0**] ==** 'Q'**) {**

p**[**i**].**type **=** 0**;**

scanf**(**"%d%d%d"**, &**p**[**i**].**l**, &**p**[**i**].**r**, &**p**[**i**].**k**);**

**}**

**else {**

p**[**i**].**type **=** 1**;**

scanf**(**"%d%d"**, &**p**[**i**].**x**, &**p**[**i**].**val**);**

b**[++**m**] =** p**[**i**].**val**;**

**}**

**}**

sort**(**b **+** 1**,** b **+** m **+** 1**);**

m **=** unique**(**b **+** 1**,** b **+** m **+** 1**) - (**b **+** 1**);**

**for(**inti **=** 1**;** i **<=** n**; ++**i**)**mpst**.**insert**(**i**,** lower\_bound**(**b **+** 1**,** b **+** m **+** 1**,** a**[**i**]) -**

b**);**

**for(**inti **=** 1**;** i **<=** q**; ++**i**) {**

**if(**p**[**i**].**type **==** 0**)**printf**(**"%d\n"**,** b**[**mpst**.**query**(**p**[**i**].**l**,** p**[**i**].**r**,** p**[**i**].**k**)]);**

42

**else {**

mpst**.**add**(**p**[**i**].**x**,** lower\_bound**(**b **+** 1**,** b **+** m **+** 1**,** a**[**p**[**i**].**x**]) -** b**, -**1**);** mpst**.**add**(**p**[**i**].**x**,** lower\_bound**(**b **+** 1**,** b **+** m **+** 1**,** p**[**i**].**val**) -** b**,** 1**);** a**[**p**[**i**].**x**] =** p**[**i**].**val**;**

**}**

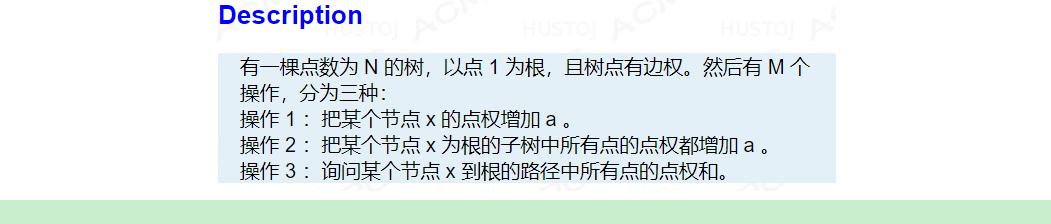
**}**

**}**

**return** 0**;**

**}**

**5、树链剖分(以 BZOJ4034 为例)**



#include<bits/stdc++.h>

**using namespace** std**;**

#define MAXN 100010

**typedef** long longLL**;**

struct edge **{**

int to**,** next**;**

edge**(**int t **=** 0**,** int nx **=** 0**):** to**(**t**),** next**(**nx**) {}** **}** es**[**MAXN **\*** 2**];**

43

int head**[**MAXN**],** ecnt**;**

int f**[**MAXN**],** deep**[**MAXN**],** size**[**MAXN**],** zson**[**MAXN**],** top**[**MAXN**],** o2n**[**MAXN**],** dfscnt**;** int n**,** q**;**

1. a**[**MAXN**];** struct node **{**

LL sum**,** lazy**;** **}** ns**[**MAXN **\*** 4**];**

void add**(**int from**,** int to**) {**

es**[++**ecnt**] =** edge**(**to**,** head**[**from**]),** head**[**from**] =** ecnt**;**

**}**

void init**() {**

memset**(**head**,** 0**,** **sizeof(**head**[**0**]) \* (**n **+** 3**));** ecnt **=** 0**;**

memset**(**zson**,** 0**,** **sizeof(**zson**[**0**]) \* (**n **+** 3**));** dfscnt **=** 0**;**

memset**(**ns**,** 0**,** **sizeof(**ns**));**

**}**

void pushdown**(**int rt**,** int l**,** int r**) {**

**if(**ns**[**rt**].**lazy**) {**

int lch **=** rt **<<** 1**,** rch **=** rt **<<** 1 **|** 1**,** mid **= (**l **+** r**) >>** 1**;** ns**[**lch**].**sum **+= (**mid **-** l **+** 1**) \*** ns**[**rt**].**lazy**;** ns**[**lch**].**lazy **+=** ns**[**rt**].**lazy**;**

ns**[**rch**].**sum **+= (**r **-** mid**) \*** ns**[**rt**].**lazy**;**

ns**[**rch**].**lazy **+=** ns**[**rt**].**lazy**;**

ns**[**rt**].**lazy **=** 0**;**

**}**

44

**}**

void update**(**int ul**,** int ur**,** LL v**,** int rt **=** 1**,** int l **=** 1**,** int r **=** n**) {** **if(**l **>** ur **||** r **<** ul**)return;**

**if(**l **>=** ul **&&** r **<=** ur**) {**

ns**[**rt**].**sum **+=** LL**(**r **-** l **+** 1**) \*** v**,** ns**[**rt**].**lazy **+=** v**;** **return;**

**}**

pushdown**(**rt**,** l**,** r**);**

int mid **= (**l **+** r**) >>** 1**,** lch **=** rt **<<** 1**,** rch **=** rt **<<** 1 **|** 1**;** **if(**ul **<=** mid**)**update**(**ul**,** ur**,** v**,** lch**,** l**,** mid**);**

**if(**ur **>** mid**)**update**(**ul**,** ur**,** v**,** rch**,** mid **+** 1**,** r**);** ns**[**rt**].**sum **=** ns**[**lch**].**sum **+** ns**[**rch**].**sum**;**

**}**

1. query0**(**int ql**,** int qr**,** int rt **=** 1**,** int l **=** 1**,** int r **=** n**) {** **if(**l **>** qr **||** r **<** ql**)return** 0LL**;**

**if(**l **>=** ql **&&** r **<=** qr**)return** ns**[**rt**].**sum**;** pushdown**(**rt**,** l**,** r**);**

int mid **= (**l **+** r**) >>** 1**,** lch **=** rt **<<** 1**,** rch **=** rt **<<** 1 **|** 1**;** LL res **=** 0**;**

**if(**ql **<=** mid**)**res **+=** query0**(**ql**,** qr**,** lch**,** l**,** mid**);**

**if(**qr **>** mid**)**res **+=** query0**(**ql**,** qr**,** rch**,** mid **+** 1**,** r**); return** res**;**

**}**

LL query**(**int u**,** int v**) {**

int tu **=** top**[**u**],** tv **=** top**[**v**];**

LL res **=** 0**;**

45

**while(**tu **!=** tv**) {**

**if(**deep**[**tu**] >=** deep**[**tv**]) {**

res **+=** query0**(**o2n**[**tu**],** o2n**[**u**]);**

u **=** f**[**tu**],** tu **=** top**[**u**];**

**}**

**else {**

res **+=** query0**(**o2n**[**tv**],** o2n**[**v**]);**

v **=** f**[**tv**],** tv **=** top**[**v**];**

**}**

**}**

**if(**o2n**[**u**] <=** o2n**[**v**])**res **+=** query0**(**o2n**[**u**],** o2n**[**v**]); else** res **+=** query0**(**o2n**[**v**],** o2n**[**u**]);**

**return** res**;**

**}**

void dfs1**(**int root**,** int par**) {**

deep**[**root**] =** deep**[**par**] +** 1**,** f**[**root**] =** par**,** size**[**root**] =** 1**;** int ms **=** 0**;**

**for(**inti **=** head**[**root**];** i**;** i **=** es**[**i**].**next**) {** int to **=** es**[**i**].**to**;**

**if(**to **!=** par**) {**

dfs1**(**to**,** root**);**

size**[**root**] +=** size**[**to**];**

**if(**ms **<** size**[**to**]) {**

ms **=** size**[**to**],** zson**[**root**] =** to**;**

**}**

**}**

**}**

**}**

46

void dfs2**(**int root**,** int par**,** int tp**) {**

top**[**root**] =** tp**;**

o2n**[**root**] = ++**dfscnt**;**

update**(**dfscnt**,** dfscnt**,** a**[**root**]);**

**if(!**zson**[**root**])return;**

dfs2**(**zson**[**root**],** root**,** tp**);**

**for(**inti **=** head**[**root**];** i**;** i **=** es**[**i**].**next**) {** int to **=** es**[**i**].**to**;**

**if(**to **!=** par **&&** to **!=** zson**[**root**]) {**

dfs2**(**to**,** root**,** to**);**

**}**

**}**

**}**

int main**() {**

* freopen("input.txt", "r", stdin); int op**,** x**,** y**,** z**;** **while(~**scanf**(**"%d%d"**, &**n**, &**q**)) {**

init**();**

**for(**inti **=** 1**;** i **<=** n**; ++**i**)**scanf**(**"%lld"**,** a **+** i**); for(**inti **=** 1**;** i **<** n**; ++**i**) {**

scanf**(**"%d%d"**, &**x**, &**y**);** add**(**x**,** y**),** add**(**y**,** x**);**

**}**

dfs1**(**1**,** 0**);** dfs2**(**1**,** 0**,** 1**);** **while(**q**--) {**

scanf**(**"%d"**, &**op**);**

47

**if(**op **==** 1**) {**

scanf**(**"%d%d"**, &**x**, &**z**);**

update**(**o2n**[**x**],** o2n**[**x**],** z**);**

**}**

**else if(**op **==** 2**) {**

scanf**(**"%d%d"**, &**x**, &**z**);**

update**(**o2n**[**x**],** o2n**[**x**] +** size**[**x**] -** 1**,** z**);**

**}**

**else {**

scanf**(**"%d"**, &**x**);**

printf**(**"%lld\n"**,** query**(**1**,** x**));**

**}**

**}**

**}**

**return** 0**;**

**}**

**6、线段树**

（1）同时支持区间加、区间乘、区间求和

**namespace** st **{**

struct node **{**

ULL sum**,** slazy**,** plazy**;**

node**(**ULL a1 **=** 0**,** ULL a2 **=** 0**,** ULL a3 **=** 0**):** sum**(**a1**),** slazy**(**a2**),** plazy**(**a3**) {}**

**}** ns**[**MAXN **\*** 4**];** void init**() {**

**for(**inti **=** 0**,** t **=** 4 **\*** n **+** 5**;** i **<** t**; ++**i**)**ns**[**i**] =** node**(**0**,** 0**,** 1**);**

**}**

48

void pushdown**(**int rt**,** int l**,** int r**) {**

int lch **=** rt **<<** 1**,** rch **=** rt **<<** 1 **|** 1**,** mid **= (**l **+** r**) >>** 1**;** **if(**ns**[**rt**].**plazy **!=** 1 **||** ns**[**rt**].**plazy **!=** 0**) {**

ns**[**lch**].**sum **=** ns**[**lch**].**sum **\*** ns**[**rt**].**plazy **+ (**mid **-** l **+** 1**) \*** ns**[**rt**].**slazy**;**

ns**[**rch**].**sum **=** ns**[**rch**].**sum **\*** ns**[**rt**].**plazy **+ (**r **-** mid**) \*** ns**[**rt**].**slazy**;**

ns**[**lch**].**plazy **\*=** ns**[**rt**].**plazy**;**

ns**[**rch**].**plazy **\*=** ns**[**rt**].**plazy**;**

ns**[**lch**].**slazy **=** ns**[**lch**].**slazy **\*** ns**[**rt**].**plazy **+** ns**[**rt**].**slazy**;** ns**[**rch**].**slazy **=** ns**[**rch**].**slazy **\*** ns**[**rt**].**plazy **+** ns**[**rt**].**slazy**;** ns**[**rt**].**plazy **=** 1**,** ns**[**rt**].**slazy **=** 0**;**

**}**

**}**

void update**(**int ul**,** int ur**,** ULL x**,** int type**,** int rt **=** 1**,** int l **=** 1**,** int r **=** n**) {** **if(**l **>** ur **||** r **<** ul**)return;**

**if(**l **>=** ul **&&** r **<=** ur**) {**

**if(**type **==** 2**)**ns**[**rt**].**sum **+=** x **\*** ULL**(**r **-** l **+** 1**),** ns**[**rt**].**slazy **+=** x**; else** ns**[**rt**].**sum **\*=** x**,** ns**[**rt**].**plazy **\*=** x**,** ns**[**rt**].**slazy **\*=** x**; return;**

**}**

pushdown**(**rt**,** l**,** r**);**

int mid **= (**l **+** r**) >>** 1**;**

**if(**ul **<=** mid**)**update**(**ul**,** ur**,** x**,** type**,** rt **<<** 1**,** l**,** mid**);**

**if(**ur **>** mid**)**update**(**ul**,** ur**,** x**,** type**,** rt **<<** 1 **|** 1**,** mid **+** 1**,** r**);** ns**[**rt**].**sum **=** ns**[**rt **<<** 1**].**sum **+** ns**[**rt **<<** 1 **|** 1**].**sum**;**

**}**

ULL query**(**int ql**,** int qr**,** int rt **=** 1**,** int l **=** 1**,** int r **=** n**) {** **if(**l **>** qr **||** r **<** ql**)return** 0LL**;**

**if(**l **>=** ql **&&** r **<=** qr**)return** ns**[**rt**].**sum**;**

pushdown**(**rt**,** l**,** r**);**

49

int mid **= (**l **+** r**) >>** 1**;**

ULL res **=** 0**;**

**if(**ql **<=** mid**)**res **+=** query**(**ql**,** qr**,** rt **<<** 1**,** l**,** mid**);**

**if(**qr **>** mid**)**res **+=** query**(**ql**,** qr**,** rt **<<** 1 **|** 1**,** mid **+** 1**,** r**); return** res**;**

**}**

**};**

（2）区间加等差数列，单点查询

#include<bits/stdc++.h>

**using namespace** std**;**

#define MAXN 100010

**typedef** long longLL**;**

struct node **{**

* 1. f**,** d**;**

**}** ns**[**MAXN **\*** 4**];** int n**,** m**;**

void pushdown**(**int rt**,** int l**,** int r**) {**

**if(**ns**[**rt**].**f **||** ns**[**rt**].**d**) {**

int mid **= (**l **+** r**) >>** 1**,** lch **=** rt **<<** 1**,** rch **=** rt **<<** 1 **|** 1**;** LL f **=** ns**[**rt**].**f**,** d **=** ns**[**rt**].**d**;**

ns**[**lch**].**f **+=** ns**[**rt**].**f**,** ns**[**lch**].**d **+=** ns**[**rt**].**d**;**

ns**[**rch**].**f **+=** ns**[**rt**].**f **+ (**mid **-** l **+** 1**) \*** d**,** ns**[**rch**].**d **+=** ns**[**rt**].**d**;** ns**[**rt**].**f **=** ns**[**rt**].**d **=** 0**;**

**}**

**}**

50

void update**(**int ul**,** int ur**,** LL f**,** LL d**,** int rt **=** 1**,** int l **=** 1**,** int r **=** n**) {** **if(**r **<** ul **||** l **>** ur**)return;**

**if(**l **>=** ul **&&** r **<=** ur**) {**

ns**[**rt**].**f **+=** f **+** 1LL **\* (**l **-** ul**) \*** d**,** ns**[**rt**].**d **+=** d**;** **return;**

**}**

pushdown**(**rt**,** l**,** r**);**

int mid **= (**l **+** r**) >>** 1**;**

**if(**ul **<=** mid**)**update**(**ul**,** ur**,** f**,** d**,** rt **<<** 1**,** l**,** mid**);**

**if(**ur **>** mid**)**update**(**ul**,** ur**,** f**,** d**,** rt **<<** 1 **|** 1**,** mid **+** 1**,** r**);**

**}**

1. query**(**int x**,** int rt **=** 1**,** int l **=** 1**,** int r **=** n**) {** **if(**l **==** r **&&** l **==** x**)return** ns**[**rt**].**f**;** pushdown**(**rt**,** l**,** r**);**

int mid **= (**l **+** r**) >>** 1**;**

**if(**x **<=** mid**)return** query**(**x**,** rt **<<** 1**,** l**,** mid**); else return** query**(**x**,** rt **<<** 1 **|** 1**,** mid **+** 1**,** r**);**

**}**

int main**() {**

int x**,** op**,** l**,** r**,** f**,** d**;**

scanf**(**"%d%d"**, &**n**, &**m**);**

**for(**inti **=** 1**;** i **<=** n**; ++**i**) {**

scanf**(**"%d"**, &**x**);**

update**(**i**,** i**,** x**,** 0**);**

**}**

**while(**m**--) {**

scanf**(**"%d"**, &**op**);**

51

**if(**op **==** 1**) {**

scanf**(**"%d%d%d%d"**, &**l**, &**r**, &**f**, &**d**);**

**if(**n **==** 0**)**printf**(**"0"**);**

**else** update**(**l**,** r**,** f**,** d**);**

**}**

**else {**

scanf**(**"%d"**, &**x**);**

**if(**n **==** 0**)**printf**(**"0"**);**

**else** printf**(**"%lld\n"**,** query**(**x**));**

**}**

**}**

**return** 0**;**

**}**

（3）区间加等差数列，区间查询

#include<bits/stdc++.h>

**using namespace** std**;**

#define MAXN 100010

**typedef** long longLL**;**

struct node **{**

* 1. f**,** d**,** sum**;**

**}** ns**[**MAXN **\*** 4**];** int n**,** m**;**

inline LL calc**(**LL a1**,** LL d**,** int len**) {**

**return** len **\*** a1 **+** d **\*** len **\* (**len **-** 1**) /** 2**;**

**}**

52

void pushdown**(**int rt**,** int l**,** int r**) {**

**if(**ns**[**rt**].**f **||** ns**[**rt**].**d**) {**

int mid **= (**l **+** r**) >>** 1**,** lch **=** rt **<<** 1**,** rch **=** rt **<<** 1 **|** 1**;**

1. f **=** ns**[**rt**].**f**,** d **=** ns**[**rt**].**d**;** ns**[**lch**].**f **+=** f**,** ns**[**lch**].**d **+=** d**;** ns**[**lch**].**sum **+=** calc**(**f**,** d**,** mid **-** l **+** 1**);**

ns**[**rch**].**f **+=** f **+ (**mid **-** l **+** 1**) \*** d**,** ns**[**rch**].**d **+=** d**;**

ns**[**rch**].**sum **+=** calc**(**f **+ (**mid **-** l **+** 1**) \*** d**,** d**,** r **-** mid**);** ns**[**rt**].**f **=** ns**[**rt**].**d **=** 0**;**

**}**

**}**

void update**(**int ul**,** int ur**,** LL f**,** LL d**,** int rt **=** 1**,** int l **=** 1**,** int r **=** n**) {** **if(**r **<** ul **||** l **>** ur**)return;**

**if(**l **>=** ul **&&** r **<=** ur**) {**

1. tf **=** f **+** 1LL **\* (**l **-** ul**) \*** d**,** td **=** d**;** ns**[**rt**].**f **+=** tf**,** ns**[**rt**].**d **+=** td**;** ns**[**rt**].**sum **+=** calc**(**tf**,** td**,** r **-** l **+** 1**);** **return;**

**}**

pushdown**(**rt**,** l**,** r**);**

int mid **= (**l **+** r**) >>** 1**;**

**if(**ul **<=** mid**)**update**(**ul**,** ur**,** f**,** d**,** rt **<<** 1**,** l**,** mid**);**

**if(**ur **>** mid**)**update**(**ul**,** ur**,** f**,** d**,** rt **<<** 1 **|** 1**,** mid **+** 1**,** r**);** ns**[**rt**].**sum **=** ns**[**rt **<<** 1**].**sum **+** ns**[**rt **<<** 1 **|** 1**].**sum**;**

**}**

1. query**(**int ql**,** int qr**,** int rt **=** 1**,** int l **=** 1**,** int r **=** n**) {** **if(**l **>** qr **||** r **<** ql**)return** 0**;**

53

**if(**l **>=** ql **&&** r **<=** qr**) {**

**return** ns**[**rt**].**sum**;**

**}**

pushdown**(**rt**,** l**,** r**);**

int mid **= (**l **+** r**) >>** 1**;**

LL sum **=** 0**;**

**if(**ql **<=** mid**)**sum **+=** query**(**ql**,** qr**,** rt **<<** 1**,** l**,** mid**);**

**if(**qr **>** mid**)**sum **+=** query**(**ql**,** qr**,** rt **<<** 1 **|** 1**,** mid **+** 1**,** r**); return** sum**;**

**}**

int main**() {**

* freopen("2.in", "r", stdin); int x**,** op**,** l**,** r**,** f**,** d**;** scanf**(**"%d%d"**, &**n**, &**m**);** **for(**inti **=** 1**;** i **<=** n**; ++**i**) {**

scanf**(**"%d"**, &**x**);** update**(**i**,** i**,** x**,** 0**);**

**}**

**while(**m**--) {** scanf**(**"%d"**, &**op**);** **if(**op **==** 1**) {**

scanf**(**"%d%d%d%d"**, &**l**,&**r**,&**f**, &**d**);** update**(**l**,** r**,** f**,** d**);**

**}**

**else {**

scanf**(**"%d%d"**, &**l**,&**r**);**

printf**(**"%lld\n"**,** query**(**l**,** r**));**

**}**

54

**}**

**return** 0**;**

**}**

**7、树分治(以 poj1741 统计树上距离不超过 k 的点对为例)**

/\*统计树上距离不超过 k 的点对，(x,y)和(y,x)只记 1 次。\*\*/

#include<cstdio>

#include<cstring>

#include<cstdlib>

#include<cassert>

#include<iostream>

#include<vector>

#include<algorithm>

**using namespace** std**;**

#define MAXN 100010

**typedef** long longLL**;**

struct edge **{**

int to**,** next**,** wt**;**

edge**(**int t **=** 0**,** int n **=** 0**,** int w **=** 0**):** to**(**t**),** next**(**n**),** wt**(**w**) {}**

**}** es**[**MAXN **\*** 2**];**

int head**[**MAXN**],** ecnt**;**

int n**,** k**;**

int siz**[**MAXN**],** maxsonsiz**[**MAXN**],** G**,** subn**;**

vector**<**int**>** dis**;**

bool vis**[**MAXN**];**

LL ans**;**

void add**(**int from**,** int to**,** int wt**) {**

55

es**[++**ecnt**] =** edge**(**to**,** head**[**from**],** wt**),** head**[**from**] =** ecnt**;**

**}**

void init**() {**

memset**(**head**,** 0**,** **sizeof(**head**[**0**]) \* (**n **+** 3**));** memset**(**vis**,** 0**,** **sizeof(**vis**[**0**]) \* (**n **+** 3**));** ecnt **=** 0**;**

G **=** 0**,** subn **=** n**,** maxsonsiz**[**G**] =** subn**;**

**}**

void dfs4siz**(**int root**,** int par**) {**

siz**[**root**] =** 1**,** maxsonsiz**[**root**] =** 0**;**

**for(**inti **=** head**[**root**];** i**;** i **=** es**[**i**].**next**) {** int to **=** es**[**i**].**to**;**

**if(**to **!=** par **&& !**vis**[**to**]) {**

dfs4siz**(**to**,** root**);**

siz**[**root**] +=** siz**[**to**];**

maxsonsiz**[**root**] =** max**(**maxsonsiz**[**root**],** siz**[**to**]);**

**}**

**}**

maxsonsiz**[**root**] =** max**(**maxsonsiz**[**root**],** subn **-** siz**[**root**]);** **if(**maxsonsiz**[**root**] <** maxsonsiz**[**G**])**G **=** root**;**

**}**

void dfs4dis**(**int root**,** int par**,** int dd**) {**

dis**.**push\_back**(**dd**);**

**for(**inti **=** head**[**root**];** i**;** i **=** es**[**i**].**next**) {** int to **=** es**[**i**].**to**;**

**if(**to **!=** par **&& !**vis**[**to**]) {**

56

dfs4dis**(**to**,** root**,** dd **+** es**[**i**].**wt**);**

**}**

**}**

**}**

1. calc**(**int root**,** int dd**) {** dis**.**clear**();** dfs4dis**(**root**,** 0**,** dd**);**

sort**(**dis**.**begin**(),** dis**.**end**());**

LL res **=** 0**;**

**for(**inti **=** 0**,** j **= (**int**)**dis**.**size**() -** 1**;** i **<** j**;) { if(**dis**[**i**] +** dis**[**j**] >** k**) {**

j**--;**

**}**

**else {**

res **+=** j **-** i**;**

i**++;**

**}**

**}**

**return** res**;**

**}**

void dfs**(**int root**,** int par**) {**

dfs4siz**(**root**,** par**);**

assert**(**G **!=** 0**);**

vis**[**G**] =** 1**;**

ans **+=** calc**(**G**,** 0**);**

**for(**inti **=** head**[**G**];** i**;** i **=** es**[**i**].**next**) {**

int to **=** es**[**i**].**to**;**

57

**if(!**vis**[**to**]) {**

ans **-=** calc**(**to**,** es**[**i**].**wt**);**

1. **=** 0**,** subn **=** siz**[**to**],** maxsonsiz**[**G**] =** subn**;** dfs**(**to**,** G**);**

**}**

**}**

**}**

int main**() {**

int x**,** y**,** z**;**

**while(**scanf**(**"%d%d"**, &**n**, &**k**),** n **||** k**) {**

init**();**

**for(**inti **=** 1**;** i **<** n**; ++**i**) {**

scanf**(**"%d%d%d"**, &**x**, &**y**, &**z**);**

add**(**x**,** y**,** z**),** add**(**y**,** x**,** z**);**

**}**

ans **=** 0**;**

dfs**(**1**,** 0**);**

printf**(**"%lld\n"**,** ans**);**

**}**

**return** 0**;**

**}**

**三、图论**

**1、最短路**

58

（1）Dijkstra(不能处理负边图)

#define MAXN 10010

**typedef** structEdge0 **{**

int to**,** wt**;**

Edge0**(**int t**,** int w**) {**

to **=** t**;**

wt **=** w**;**

**}**

**}** Edge**;**

vector**<**Edge**>** g**[**MAXN**];**

int V**,** E**,** S**;**

**typedef** structNode0**{**

int p**,** d**;**

Node0**(**int \_p**,** int \_d**){**

1. **=** \_p**;** d **=** \_d**;**

**}**

bool **operator** **> (**const Node0**&** n**)** const**{**

**return** d **>** n**.**d**;**

**}**

**}**Node**;**

int d**[**MAXN**];**

priority\_queue**<**Node**,** vector**<**Node**>,** greater**<**Node**> >** que**;**

void dijkstra**(**int s**) {**

59

memset**(**d**,** 0x3f**,** **sizeof(**d**));**

**while(!**que**.**empty**())**que**.**pop**();**

d**[**s**] =** 0**,** que**.**push**(**Node**(**s**,** 0**));**

**while(!**que**.**empty**()) {**

Node nd **=** que**.**top**();**

que**.**pop**();**

**for(**inti **=** 0**;** i **<** g**[**nd**.**p**].**size**(); ++**i**) {**

Edge**&** e **=** g**[**nd**.**p**][**i**];**

**if(**d**[**e**.**to**] >** d**[**nd**.**p**] +** e**.**wt**){**

d**[**e**.**to**] =** d**[**nd**.**p**] +** e**.**wt**;**

que**.**push**(**Node**(**e**.**to**,** d**[**e**.**to**]));**

**}**

**}**

**}**

**}**

（2）SPFA-LLF 优化(能处理负边图)

#define MAXN 200010

**typedef** structEdge0 **{**

int to**,** wt**;**

Edge0**(**int t**,** int w**) {**

to **=** t**,** wt **=** w**;**

**}**

**}** Edge**;**

vector**<**Edge**>** g**[**MAXN**];**

60

int V**,** E**,** S**;**

long long d**[**MAXN**];**

bool inque**[**MAXN**];**

deque**<**int**>** que**;**

void spfa**(**int s**) {**

memset**(**d**,** 0x3f**,** **sizeof(**d**));**

memset**(**inque**,** 0**,** **sizeof(**inque**));**

**while(!**que**.**empty**())**que**.**pop\_front**();**

d**[**s**] =** 0**,** inque**[**s**] =** **true,** que**.**push\_back**(**s**);** **while(!**que**.**empty**()) {**

int v **=** que**.**front**();**

que**.**pop\_front**();**

inque**[**v**] =** **false;**

**for(**inti **=** 0**;** i **<** g**[**v**].**size**(); ++**i**) {**

Edge**&** e **=** g**[**v**][**i**];**

**if(**d**[**e**.**to**] >** d**[**v**] +** e**.**wt**) {**

d**[**e**.**to**] =** d**[**v**] +** e**.**wt**;**

**if(!**inque**[**e**.**to**]) {**

**if(**que**.**empty**() ||** d**[**e**.**to**] >** d**[**que**.**front**()])**que**.**push\_back**(**e**.**to**); else** que**.**push\_front**(**e**.**to**);**

inque**[**e**.**to**] =** **true;**

**}**

**}**

**}**

**}**

61

**}**

（3）SPFA-LLL 优化(能处理负边图)

#define MAXN 200010

**typedef** structEdge0 **{**

int to**,** wt**;**

Edge0**(**int t**,** int w**) {**

to **=** t**,** wt **=** w**;**

**}**

**}** Edge**;**

vector**<**Edge**>** g**[**MAXN**];**

int V**,** E**,** S**;**

long long d**[**MAXN**],** sum**;**

bool inque**[**MAXN**];**

deque**<**int**>** que**;**

void spfa**(**int s**) {**

memset**(**d**,** 0x3f**,** **sizeof(**d**));**

memset**(**inque**,** 0**,** **sizeof(**inque**));**

**while(!**que**.**empty**())**que**.**pop\_front**();**

d**[**s**] =** 0**,** inque**[**s**] =** **true,** que**.**push\_back**(**s**),** sum **=** 0**;** **while(!**que**.**empty**()) {**

int v **=** que**.**front**();**

**while(**d**[**v**] \*** que**.**size**() >** sum**) {**

que**.**pop\_front**();**

62

que**.**push\_back**(**v**);**

v **=** que**.**front**();**

**}**

que**.**pop\_front**();**

inque**[**v**] =** **false;**

sum **-=** d**[**v**];**

**for(**inti **=** 0**;** i **<** g**[**v**].**size**(); ++**i**) {**

Edge**&** e **=** g**[**v**][**i**];**

**if(**d**[**e**.**to**] >** d**[**v**] +** e**.**wt**) {**

d**[**e**.**to**] =** d**[**v**] +** e**.**wt**;**

**if(!**inque**[**e**.**to**]) {**

**if(**que**.**empty**() ||** d**[**e**.**to**] >** d**[**que**.**front**()])**que**.**push\_back**(**e**.**to**); else** que**.**push\_front**(**e**.**to**);**

inque**[**e**.**to**] =** **true;**

sum **+=** d**[**e**.**to**];**

**}**

**}**

**}**

**}**

**}**

**2、最小生成树**

**typedef** structEdge0**{**

int u**,** v**,** w**;**

Edge0**(**int \_u**,** int \_v**,** int \_w**){**

63

u **=** \_u**,** v **=** \_v**,** w **=** \_w**;**

**}**

bool **operator** **< (**const Edge0**&** e**)** const**{**

**return** w **<** e**.**w**;**

**}**

**}**Edge**;**

vector**<**Edge**>** edges**;**

UniFinder uf**;** //用之前记得要初始化 uf.init();

int kruskal**(){**

int res **=** 0**;**

sort**(**edges**.**begin**(),** edges**.**end**());**

**for(**inti **=** 0**;**i **<** edges**.**size**();++**i**){**

Edge**&** e **=** edges**[**i**];**

**if(!**uf**.**same**(**e**.**u**,** e**.**v**)){**

res **+=** e**.**w**;**

uf**.**unite**(**e**.**u**,** e**.**v**);**

**}**

**}**

**return** res**;**

**}**

**四、数学**

**1、组合数(lucas 定理+费马小定理，要求模数 p 是素数)**

64

#define MAXN 100100

**using namespace** std**;**

**typedef** long longLL**;**

LL facts**[**MAXN**],** p**;**

1. qpow**(**LL a**,** LL x**) {** LL res **=** 1**;**

**while(**x **>** 0**) {**

**if(**x **&** 1**)**res **= (**res **\*** a**) %** p**;**

a **= (**a **\*** a**) %** p**;**

x **>>=** 1**;**

**}**

**return** res**;**

**}**

/\*调用 lucas(n, m)前一定初始化。同时保证 p 是素数。\*/

void init**() {**

facts**[**0**] =** 1**;**

**for(**inti **=** 1**;** i **<** MAXN**; ++**i**) {**

facts**[**i**] = (**facts**[**i **-** 1**] \*** i**) %** p**;**

**}**

**}**

LL C**(**LL n**,** LL m**) {**

**if(**n **<** m**)return** 0**;**

**if(**n **==** m **||** m **==** 0**)return** 1**;**

65

**if(**m **==** 1**)return** n **%** p**;**

**return** facts**[**n**] \*** qpow**(**facts**[**m**],** p **-** 2**) %** p **\*** qpow**(**facts**[**n **-** m**],** p **-** 2**) %** p**;**

**}**

LL lucas**(**LL n**,** LL m**) {**

**if(**n **<** p **&&** m **<** p**)return** C**(**n**,** m**);**

**else return** lucas**(**n **/** p**,** m **/** p**) \*** lucas**(**n **%** p**,** m **%** p**) %** p**;**

**}**

**2、快速拉格朗日插值法求和**

#include<bits/stdc++.h>

**using namespace** std**;**

#define MAXN 1010

**typedef** long longLL**;**

const LL mod **=** 1e9 **+** 7**;**

LL powmod**(**LL aa**,** LL x**) {**

LL res **=** 1**;**

**for(;** x **>** 0**;** x **>>=** 1**) {**

**if(**x **&** 1**)**res **= (**res **\*** aa**) %** mod**;**

aa **= (**aa **\*** aa**) %** mod**;**

**}**

**return** res**;**

**}**

struct lagrange **{**

66

#define ll long long

#define rep(i,a,n) for (int i=a;i<n;i++) #define per(i,a,n) for (int i=n-1;i>=a;i--) #define D 2010 //D 比 MAXN 大 100 就行

1. a**[**D**],** f**[**D**],** g**[**D**],** p**[**D**],** p1**[**D**],** p2**[**D**],** b**[**D**],** h**[**D**][**2**],** C**[**D**];** void init**(**int M**) {**//初始化：参数填 MAXN + 20

f**[**0**] =** f**[**1**] =** g**[**0**] =** g**[**1**] =** 1**;**

rep**(**i**,** 2**,** M **+** 5**)** f**[**i**] =** f**[**i **-** 1**] \*** i **%** mod**;** g**[**M **+** 4**] =** powmod**(**f**[**M **+** 4**],** mod **-** 2**);**

per**(**i**,** 1**,** M **+** 4**)** g**[**i**] =** g**[**i **+** 1**] \* (**i **+** 1**) %** mod**;**

**}**

/\*给定一组样本数据 a[]，规模为 0-d，计算出第 n 项\*/ ll calcn**(**int d**,** ll **\***a**,** ll n**) {**

**if (**n **<=** d**) return** a**[**n**];** p1**[**0**] =** p2**[**0**] =** 1**;** rep**(**i**,** 0**,** d **+** 1**) {**

1. t **= (**n **-** i **+** mod**) %** mod**;** p1**[**i **+** 1**] =** p1**[**i**] \*** t **%** mod**;**

**}**

rep**(**i**,** 0**,** d **+** 1**) {**

1. t **= (**n **-** d **+** i **+** mod**) %** mod**;** p2**[**i **+** 1**] =** p2**[**i**] \*** t **%** mod**;**

**}**

ll ans **=** 0**;**

rep**(**i**,** 0**,** d **+** 1**) {**

ll t **=** g**[**i**] \*** g**[**d **-** i**] %** mod **\*** p1**[**i**] %** mod **\*** p2**[**d **-** i**] %** mod **\*** a**[**i**] %** mod**;**

67

**if ((**d **-** i**) &** 1**)** ans **= (**ans **-** t **+** mod**) %** mod**; else** ans **= (**ans **+** t**) %** mod**;**

**}**

**return** ans**;**

**}**

/\*

给定一组观测点(0, a[0]), (1, a[1]), ...，(m, a[m])，、样本点的个数为 f(x)的最高次+1。

求在该函数模型下，a[0]+a[1]+...+a[n]的和。

\*/

ll ta**[**D**];**

1. polysum**(**ll m**,** ll **\***a**,** ll n**) {** // a[0].. a[m] \sum\_{i=0}^{n} a[i] memcpy**(**ta**,** a**,** **sizeof(**a**[**0**]) \* (**m **+** 1**));**

ta**[**m **+** 1**] =** calcn**(**m**,** ta**,** m **+** 1**);**

rep**(**i**,** 1**,** m **+** 2**)**ta**[**i**] = (**ta**[**i **-** 1**] +** ta**[**i**]) %** mod**;** **return** calcn**(**m **+** 1**,** ta**,** n**);**

**}**

**};**

lagrange lag**;**

int main**(){**

lag**.**init**(**MAXN **+** 20**);**

**return** 0**;**

**}**

(1)例题(计算通式 3 + 2 + + 1的前3 × 108的和)

68

#include<bits/stdc++.h>

**using namespace** std**;**

#define MAXN 1010

**typedef** long longLL**;**

const LL mod **=** 1e9 **+** 7**;**

1. powmod**(**LL aa**,** LL x**) {** LL res **=** 1**;**

**for(;** x **>** 0**;** x **>>=** 1**) {**

**if(**x **&** 1**)**res **= (**res **\*** aa**) %** mod**;**

aa **= (**aa **\*** aa**) %** mod**;**

**}**

**return** res**;**

**}**

struct lagrange **{**

#define ll long long

#define rep(i,a,n) for (int i=a;i<n;i++) #define per(i,a,n) for (int i=n-1;i>=a;i--) #define D 2010 //D 比 MAXN 大 100 就行

1. a**[**D**],** f**[**D**],** g**[**D**],** p**[**D**],** p1**[**D**],** p2**[**D**],** b**[**D**],** h**[**D**][**2**],** C**[**D**];** void init**(**int M**) {**//初始化：参数填 MAXN + 20

f**[**0**] =** f**[**1**] =** g**[**0**] =** g**[**1**] =** 1**;**

rep**(**i**,** 2**,** M **+** 5**)** f**[**i**] =** f**[**i **-** 1**] \*** i **%** mod**;** g**[**M **+** 4**] =** powmod**(**f**[**M **+** 4**],** mod **-** 2**);**

per**(**i**,** 1**,** M **+** 4**)** g**[**i**] =** g**[**i **+** 1**] \* (**i **+** 1**) %** mod**;**

**}**

/\*给定一组样本数据 a[]，规模为 0-d，计算出第 n 项\*/

69

1. calcn**(**int d**,** ll **\***a**,** ll n**) {** **if (**n **<=** d**) return** a**[**n**];** p1**[**0**] =** p2**[**0**] =** 1**;** rep**(**i**,** 0**,** d **+** 1**) {**

ll t **= (**n **-** i **+** mod**) %** mod**;**

p1**[**i **+** 1**] =** p1**[**i**] \*** t **%** mod**;**

**}**

rep**(**i**,** 0**,** d **+** 1**) {**

1. t **= (**n **-** d **+** i **+** mod**) %** mod**;** p2**[**i **+** 1**] =** p2**[**i**] \*** t **%** mod**;**

**}**

ll ans **=** 0**;**

rep**(**i**,** 0**,** d **+** 1**) {**

1. t **=** g**[**i**] \*** g**[**d **-** i**] %** mod **\*** p1**[**i**] %** mod **\*** p2**[**d **-** i**] %** mod **\*** a**[**i**] %** mod**;** **if ((**d **-** i**) &** 1**)** ans **= (**ans **-** t **+** mod**) %** mod**;**

**else** ans **= (**ans **+** t**) %** mod**;**

**}**

**return** ans**;**

**}**

/\*

给定一组观测点(0, a[0]), (1, a[1]), ...，(m, a[m])，、样本点的个数为 f(x)的最高次+1。

求在该函数模型下，a[0]+a[1]+...+a[n]的和。

\*/

ll ta**[**D**];**

ll polysum**(**ll m**,** ll **\***a**,** ll n**) {** // a[0].. a[m] \sum\_{i=0}^{n} a[i]

70

memcpy**(**ta**,** a**,** **sizeof(**a**[**0**]) \* (**m **+** 1**));**

ta**[**m **+** 1**] =** calcn**(**m**,** ta**,** m **+** 1**);**

rep**(**i**,** 1**,** m **+** 2**)**ta**[**i**] = (**ta**[**i **-** 1**] +** ta**[**i**]) %** mod**;** **return** calcn**(**m **+** 1**,** ta**,** n**);**

**}**

**};**

lagrange lag**;**

int main**(){**

lag**.**init**(**MAXN **+** 20**);**

**return** 0**;**

**}**

int main**(){**

lag**.**init**(**MAXN **+** 20**);**

**typedef** long longLL**;**

1. a**[] = {**1**,** 4**,** 15**,** 40**};**//x^3+x^2+x+1 LL ans **=** lag**.**polysum**(**3**,** a**,** 300000000**);** cout **<<** ans **<<** endl**;**

LL ans2 **=** 0**;**

**for(**LL i **=** 0**;**i **<=** 300000000LL**;++**i**){**

ans2 **+= (**i **\*** i **%** mod **\*** i **+** i **\*** i **%** mod **+** i **+** 1**);** ans2 **%=** mod**;**

**}**cout **<<** ans2 **<<** endl**; return** 0**;**

**}**

71

**3、预处理** ~ **内数字 和阶乘的逆元，计算组合数**

#include<bits/stdc++.h>

**using namespace** std**;**

**typedef** long longLL**;**

const LL mod **=** 998244353**;**

#define MAXN 1000100

LL fact**[**MAXN**],** inv**[**MAXN**],** finv**[**MAXN**];**

1. qpow**(**LL a**,** LL x**) {** LL res **=** 1**;**

**for(;** x **>** 0**;** x **>>=** 1**) {**

**if(**x **&** 1**)**res **= (**res **\*** a**) %** mod**;**

a **= (**a **\*** a**) %** mod**;**

**}**

**return** res**;**

**}**

void init**() {**

fact**[**0**] =** fact**[**1**] =** 1**;**

inv**[**0**] =** inv**[**1**] =** 1**;**

**for(**inti **=** 2**;** i **<** MAXN**;** i**++){**

fact**[**i**] =** fact**[**i **-** 1**] \*** i **%** mod**;**

inv**[**i**] = (**mod **-** mod **/** i**) \*** inv**[**mod **%** i**] %** mod**;;**

**}**

finv**[**MAXN **-** 1**] =** qpow**(**fact**[**MAXN **-** 1**],** mod **-** 2**);**

**for(**inti **=** MAXN **-** 2**;** i **>=** 0**; --**i**)** finv**[**i**] =** finv**[**i **+** 1**] \* (**i **+** 1**) %** mod**;**

**}**

72

LL C**(**int n**,** int m**){**

**if(**n **==** m **||** m **==** 0**)return** 1LL**;**

**return (**fact**[**n**] \*** finv**[**n **-** m**] %** mod**) \*** finv**[**m**] %** mod**;**

**}**

int main**(){**

**return** 0**;**

**}**

**4、线性基**

（1）查第 k 小（不支持动态插入）

#include <cstdio>

#include <vector>

const int MAXN **=** 100010**;**

const int MAXL **=** 62**;**

struct LinearBasis **{**

std**::**vector**<**long long**>** v**;**

int n**;**

void build**(**long long **\***x**,** int n**) {**

**this->**n **=** n**;**

std**::**vector**<**long long**>** a**(**MAXL **+** 1**);**

**for (**inti **=** 1**;** i **<=** n**;** i**++) {**

long long t **=** x**[**i**];**

**for (**intj **=** MAXL**;** j **>=** 0**;** j**--) {**

**if ((**t **& (**1ll **<<** j**)) ==** 0**) continue;**

**if (**a**[**j**])** t **^=** a**[**j**];**

**else {**

73

**for (**intk **=** 0**;** k **<** j**;** k**++) if (**t **& (**1ll **<<** k**))** t **^=** a**[**k**];**

**for (**intk **=** j **+** 1**;** k **<=** MAXL**;** k**++) if (**a**[**k**] & (**1ll **<<** j**))** a**[**k**] ^=** t**;** a**[**j**] =** t**;**

**break;**

**}**

**}**

**}**

v**.**clear**();**

**for (**inti **=** 0**;** i **<=** MAXL**;** i**++) if (**a**[**i**])** v**.**push\_back**(**a**[**i**]);**

**}**

long long kmin**(**long long k**) {**//从 1 开始的第 k 小 **if (**int**(**v**.**size**()) !=** n**) {**

* 可能是 0

k**--;**

**}**

* 如果 k 超过的所有不同异或和的数量

**if (**k **> (**1ll **<<** v**.**size**()) -** 1**) return -**1**;**

long long ans **=** 0**;**

**for (**size\_ti **=** 0**;** i **<** v**.**size**();** i**++) {**

**if (**k **& (**1ll **<<** i**)) {**

ans **^=** v**[**i**];**

**}**

**}**

**return** ans**;**

**}**

**}** lb**;**

int main**() {**

74

int n**,** T**;**

scanf**(**"%d"**, &**T**);**

**for(**intca **=** 1**;** ca **<=** T**; ++**ca**) {**

scanf**(**"%d"**, &**n**);**

static long long a**[**MAXN **+** 1**];**

**for (**inti **=** 1**;** i **<=** n**;** i**++)** scanf**(**"%lld"**, &**a**[**i**]);** lb**.**build**(**a**,** n**);**

int m**;**

scanf**(**"%d"**, &**m**);**

printf**(**"Case #%d:\n"**,** ca**);**

**while (**m**--) {**

long long k**;**

scanf**(**"%lld"**, &**k**);**

printf**(**"%lld\n"**,** lb**.**kmin**(**k**));**

**}**

**}**

**return** 0**;**

**}**

（2）查最大、最小、合并（支持动态插入）

#include<bits/stdc++.h>

**using namespace** std**;**

struct LinearBasis **{**

#define MAXL 62

long long a**[**MAXL **+** 1**];**

LinearBasis**() {**

std**::**fill**(**a**,** a **+** MAXL **+** 1**,** 0**);**

**}**

LinearBasis**(**long long **\***x**,** int n**) {**

75

build**(**x**,** n**);**

**}**

void insert**(**long long t**) {**

**for (**intj **=** MAXL**;** j **>=** 0**;** j**--) {**

**if (!**t**) return;**

**if (!(**t **& (**1ll **<<** j**))) continue;**

**if (**a**[**j**])** t **^=** a**[**j**];**

**else {**

**for (**intk **=** 0**;** k **<** j**;** k**++) if (**t **& (**1ll **<<** k**))** t **^=** a**[**k**];**

**for (**intk **=** j **+** 1**;** k **<=** MAXL**;** k**++) if (**a**[**k**] & (**1ll **<<** j**))** a**[**k**] ^=** t**;** a**[**j**] =** t**;**

**break;**

**}**

**}**

**}**

* 数组 x 表示集合 S，下标范围 [1...n] void build**(**long long **\***x**,** int n**) {**

memset**(**a**,** 0**,** **sizeof(**a**));**

**for (**inti **=** 1**;** i **<=** n**;** i**++) {** insert**(**x**[**i**]);**

**}**

**}**

long long queryMax**() {**

long long res **=** 0**;**

**for (**inti **=** 0**;** i **<=** MAXL**;** i**++)** res **^=** a**[**i**]; return** res**;**

**}**

76

void mergeFrom**(**const LinearBasis **&**other**) {**

**for (**inti **=** 0**;** i **<=** MAXL**;** i**++)** insert**(**other**.**a**[**i**]);**

**}**

static LinearBasis merge**(**const LinearBasis **&**a**,** const LinearBasis **&**b**) {** LinearBasis res **=** a**;**

**for (**inti **=** 0**;** i **<=** MAXL**;** i**++)** res**.**insert**(**b**.**a**[**i**]); return** res**;**

**}**

**}** lb**;**

long long a**[**55**];**

int main**() {**

int n**;**

scanf**(**"%d"**, &**n**);**

**for(**inti **=** 1**;** i **<=** n**; ++**i**)**scanf**(**"%lld"**,** a **+** i**);** lb **=** LinearBasis**(**a**,** n**);**

printf**(**"%lld\n"**,** lb**.**queryMax**());**

**return** 0**;**

**}**

**5、FWT**

#include<bits/stdc++.h>

**using namespace** std**;**

**typedef** long longLL**;**

const LL mod **=** 1e9 **+** 7**;**

/\*

77

如果不需要取模，直接去掉取模，乘逆元变成除法，不影响最终结果。

调用时，只需要改变 solve()函数中的 type 即可。

\*\*/

struct FFF **{**

#define AND 0

#define OR 1

#define XOR 2

//下标从 0 开始,N 为 2 的幂次

int N**,** type**;**

public**:**

void init**(**int n**) {**

N **=** n**;**

type **=** XOR**;**

**}**

private**:**

template**<**class T**>** void FWT**(**T a**[]) {**

**for(**inti **=** 1**;** i **<** N**;** i **<<=** 1**) {**

**for(**intp **=** i **<<** 1**,** q **=** 0**;** q **<** N**;** q **+=** p**) {**

**for(**intj **=** 0**;** j **<** i**;** j**++) {**

T x **=** a**[**q **+** j**],** y **=** a**[**q **+** j **+** i**];**

//and

**if(**type **==** AND**)**a**[**q **+** j**] = (**x **+** y**) %** mod**;**

//or

**else if(**type **==** OR**)**a**[**q **+** j **+** i**] = (**x **+** y**) %** mod**;** //xor

**else if(**type **==** XOR**)**a**[**q **+** j**] = (**x **+** y**) %** mod**,** a**[**q **+** j **+** i**] = (**x **-** y **+**

mod**) %** mod**;**

**}**

**}**

78

**}**

**}**

private**:**

template**<**class T**>** void UFWT**(**T a**[]) {**

T inv2 **= (**mod **+** 1**) >>** 1**;**

**for(**inti **=** 1**;** i **<** N**;** i **<<=** 1**) {**

**for(**intp **=** i **<<** 1**,** q **=** 0**;** q **<** N**;** q **+=** p**) {**

**for(**intj **=** 0 **;** j **<** i**;** j**++) {**

T x **=** a**[**q **+** j**],** y **=** a**[**q **+** j **+** i**];**

//and

**if(**type **==** AND**)**a**[**q **+** j**] = (**x **-** y **+** mod**) %** mod**;** //or

**else if(**type **==** OR**)**a**[**q **+** j **+** i**] = (**y **-** x **+** mod**) %** mod**;** //xor

**else if(**type **==** XOR**)**a**[**q **+** j**] =** 1LL **\* (**x **+** y**) \*** inv2 **%** mod**,** a**[**q **+** j **+** i**]**

* **(**1LL **\* (**x **-** y**) \*** inv2 **%** mod **+** mod**) %** mod**;**

**}**

**}**

**}**

**}**

public**:**

template**<**class T**>** void solve**(**T a**[],** T b**[],** T c**[]) {** FWT**(**a**),** FWT**(**b**);**

**for(**inti **=** 0**;** i **<** N**;** i**++)** c**[**i**] =** 1LL **\*** a**[**i**] \*** b**[**i**] %** mod**;** UFWT**(**c**);**

**}**

**}** fwt**;**

79

**五、字符串**

**六、其他**

**1、输入挂**

(1)slowIO

template **<**class T**>** inline void read**(**T **&**x**) {** char t**;**

bool sign **=** **false;**

**while((**t **=** getchar**()) !=** '-' **&& (**t **<** '0' **||** t **>** '9'**)); if(**t **==** '-'**)** sign **= true,** t **=** getchar**();**

x **=** t **-** '0'**;**

**while((**t **=** getchar**()) >=** '0' **&&** t **<=** '9'**)** x **=** x **\*** 10 **+** t **-** '0'**; if(**sign**)** x **= -**x**;**

**}**

(2)fastIO

**namespace** fastIO **{**

#define BUF\_SIZE 100000

//fread -> read

bool IOerror **=** 0**;**

inline char nc**() {**

static char buf**[**BUF\_SIZE**], \***p1 **=** buf **+** BUF\_SIZE**, \***pend **=** buf **+** BUF\_SIZE**;** **if(**p1 **==** pend**) {**

p1 **=** buf**;**

80

pend **=** buf **+** fread**(**buf**,** 1**,** BUF\_SIZE**,** stdin**);** **if(**pend **==** p1**) {**

IOerror **=** 1**;**

**return -**1**;**

**}**

**}**

**return \***p1**++;**

**}**

inline bool blank**(**char ch**) {**

**return** ch **==** ' ' **||** ch **==** '\n' **||** ch **==** '\r' **||** ch **==** '\t'**;**

**}**

template**<**class T**>**

inline void read**(**T **&**x**) {**

bool sign **=** 0**;** char ch **=** nc**();** x **=** 0**;**

**for (;** blank**(**ch**);** ch **=** nc**());**

**if (**IOerror**)return;**

**if (**ch **==** '-'**)**sign **=** 1**,** ch **=** nc**();**

**for (;** ch **>=** '0' **&&** ch **<=** '9'**;** ch **=** nc**())**x **=** x **\*** 10 **+** ch **-** '0'**; if (**sign**)**x **= -**x**;**

**}**

struct Ostream\_fwrite **{**

char **\***buf**, \***p1**, \***pend**;**

Ostream\_fwrite**() {**buf **=** **new** char**[**BUF\_SIZE**];** p1 **=** buf**;** pend **=** buf **+** BUF\_SIZE**;}** void out**(**char ch**) {**

**if (**p1 **==** pend**) {**

fwrite**(**buf**,** 1**,** BUF\_SIZE**,** stdout**);** p1 **=** buf**;**

**}**

**\***p1**++ =** ch**;**

**}**

81

template**<**class T**>** void print**(**T x**) {**

static char s**[**25**], \***s1**;** s1 **=** s**;**

**if (!**x**)\***s1**++ =** '0'**;**

**if (**x **<** 0**)**out**(**'-'**),** x **= -**x**;**

**while(**x**)\***s1**++ =** x **%** 10 **+** '0'**,** x **/=** 10**;**

**while(**s1**-- !=** s**)**out**(\***s1**);**

**}**

void print**(**char **\***s**) {while** **(\***s**)**out**(\***s**++);}**

void flush**() {if** **(**p1 **!=** buf**) {**fwrite**(**buf**,** 1**,** p1 **-** buf**,** stdout**);** p1 **=** buf**;}}** **~**Ostream\_fwrite**() {**flush**();}**

**}** Ostream**;** template**<**class T**>**

inline void print**(**T x**) {**Ostream**.**print**(**x**);}** template**<**class T**>**

inline void println**(**T x**) {**Ostream**.**print**(**x**);**Ostream**.**out**(**'\n'**);}** inline void print**(**char **\***s**) {**Ostream**.**print**(**s**);}**

inline void println**(**char **\***s**) {**Ostream**.**print**(**s**);**Ostream**.**out**(**'\n'**);}**

**};**

**using namespace** fastIO**;**

**2、常用函数**

（1）字符数组分割 strtok

#include<cstdio>

#include<cstring>

#include<cstdlib>

char str**[] =** "12#34#56#78"**;**

int main**(){**

82

char **\***tmp **=** strtok**(**str**,** "#"**);**

**do{**

printf**(**"%d\n"**,** atoi**(**tmp**));**

tmp **=** strtok**(NULL,** "#"**);**

**}while(**tmp **!= NULL);**

**return** 0**;**

**}**

（2）数字转字符数组 sprintf

char buf**[**70**];**//确保足够

sprintf**(**buf**,** "%d"**,** 123456**);**

（3）string 转字符数组

string s**(**"123456"**);**

char buf**[**70**];**//确保足够

strcpy**(**buf**,** s**.**data**());**

（4）nth\_element(把第 n 小的元素放在第 n 位置)

#include<bits/stdc++.h>

**using namespace** std**;**

int main**() {**

int a**[] = {**1**,** 7**,** 5**,** 3**,** 2**,** 8**,** 4**,** 6**};**

**for(**intk **=** 0**;** k **<** 8**; ++**k**) {**

nth\_element**(**a**,** a **+** k**,** a **+** 8**);**

printf**(**"The %dth smallest is a[%d]:"**,** k**,** k**);**

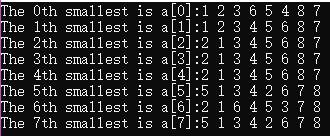
83

**for(**inti **=** 0**;** i **<** 8**; ++**i**)**printf**(**"%d "**,** a**[**i**]);** cout **<<** endl**;**

**}**

**return** 0**;**

**}**



（5）编译器内置的二进制处理函数

— Built**-**in Function**:** int \_\_builtin\_ffs **(**unsigned int x**)**

Returns one plus the index of the least significant 1**-**bit of x**,** **or if** x is zero**,** returns zero**.**

返回右起第一个‘1’的位置。

— Built**-**in Function**:** int \_\_builtin\_clz **(**unsigned int x**)**

Returns the number of leading 0**-**bits in x**,** starting at the most significant bit position**.**

If x is 0**,** the result is undefined**.**

返回左起第一个‘1’之前 0 的个数。

— Built**-**in Function**:** int \_\_builtin\_ctz **(**unsigned int x**)**

Returns the number of trailing 0**-**bits in x**,** starting at the least significant bit position**.** If x is 0**,** the result is undefined**.**

返回右起第一个‘1’之后的 0 的个数。

— Built**-**in Function**:** int \_\_builtin\_popcount **(**unsigned int x**)** Returns the number of 1**-**bits in x**.**

84

返回‘1’的个数。

— Built**-**in Function**:** int \_\_builtin\_parity **(**unsigned int x**)**

Returns the parity of x**,** i**.**e**.** the number of 1**-**bits in x modulo 2.

返回‘1’的个数的奇偶性。

— Built**-**in Function**:** int \_\_builtin\_ffsl **(**unsigned long**)**

Similar to \_\_builtin\_ffs**,** except the argument type is unsigned long**.**

— Built**-**in Function**:** int \_\_builtin\_clzl **(**unsigned long**)**

Similar to \_\_builtin\_clz**,** except the argument type is unsigned long**.**

— Built**-**in Function**:** int \_\_builtin\_ctzl **(**unsigned long**)**

Similar to \_\_builtin\_ctz**,** except the argument type is unsigned long**.**

— Built**-**in Function**:** int \_\_builtin\_popcountl **(**unsigned long**)**

Similar to \_\_builtin\_popcount**,** except the argument type is unsigned long**.**

— Built**-**in Function**:** int \_\_builtin\_parityl **(**unsigned long**)**

Similar to \_\_builtin\_parity**,** except the argument type is unsigned long**.**

— Built**-**in Function**:** int \_\_builtin\_ffsll **(**unsigned long long**)**

Similar to \_\_builtin\_ffs**,** except the argument type is unsigned long long**.**

— Built**-**in Function**:** int \_\_builtin\_clzll **(**unsigned long long**)**

Similar to \_\_builtin\_clz**,** except the argument type is unsigned long long**.**

— Built**-**in Function**:** int \_\_builtin\_ctzll **(**unsigned long long**)**

Similar to \_\_builtin\_ctz**,** except the argument type is unsigned long long**.**

85

— Built**-**in Function**:** int \_\_builtin\_popcountll **(**unsigned long long**)**

Similar to \_\_builtin\_popcount**,** except the argument type is unsigned long long**.**

— Built**-**in Function**:** int \_\_builtin\_parityll **(**unsigned long long**)**

Similar to \_\_builtin\_parity**,** except the argument type is unsigned long long**.**

**3、BM 算法(用于求线性递推的第 项)**

#include<bits/stdc++.h>

**using namespace** std**;**

#define rep(i,a,n) for (int i=a;i<n;i++)

#define per(i,a,n) for (int i=n-1;i>=a;i--)

#define pb push\_back

#define mp make\_pair

#define all(x) (x).begin(),(x).end()

#define fi first

#define se second

#define SZ(x) ((int)(x).size())

**typedef** long longll**;**

**typedef** vector**<**ll**>** VI**;**

**typedef** pair**<**int**,** int**>** PII**;**

const ll mod **=** 1000000007**;**

ll powmod**(**ll a**,** ll b**) {**

1. res **=** 1**;** a **%=** mod**;** assert**(**b **>=** 0**);**

**for(;** b**;** b **>>=** 1**) {**

**if(**b **&** 1**)**res **=** res **\*** a **%** mod**;**

86

a **=** a **\*** a **%** mod**;**

**}**

**return** res**;**

**}**

// head

**namespace** linear\_seq **{**

const int N **=** 10010**;**

ll res**[**N**],** base**[**N**],** \_c**[**N**],** \_md**[**N**];**

vector**<**int**>** Md**;**

void mul**(**ll **\***a**,** ll **\***b**,** int k**) {**

rep**(**i**,** 0**,** k **+** k**)** \_c**[**i**] =** 0**;**

rep**(**i**,** 0**,** k**)** **if** **(**a**[**i**])** rep**(**j**,** 0**,** k**)** \_c**[**i **+** j**] = (**\_c**[**i **+** j**] +** a**[**i**] \*** b**[**j**]) %** mod**;** **for (**inti **=** k **+** k **-** 1**;** i **>=** k**;** i**--) if (**\_c**[**i**])**

rep**(**j**,** 0**,** SZ**(**Md**))** \_c**[**i **-** k **+** Md**[**j**]] = (**\_c**[**i **-** k **+** Md**[**j**]] -** \_c**[**i**] \*** \_md**[**Md**[**j**]]) %** mod**;**

rep**(**i**,** 0**,** k**)** a**[**i**] =** \_c**[**i**];**

**}**

int solve**(**ll n**,** VI a**,** VI b**) {** // a 系数 b 初值 b[n+1]=a[0]\*b[n]+...

* printf("SIZE %d\n",SZ(b)); ll ans **=** 0**,** pnt **=** 0**;**

int k **=** SZ**(**a**);**

assert**(**SZ**(**a**) ==** SZ**(**b**));**

rep**(**i**,** 0**,** k**)** \_md**[**k **-** 1 **-** i**] = -**a**[**i**];** \_md**[**k**] =** 1**;** Md**.**clear**();**

rep**(**i**,** 0**,** k**)** **if** **(**\_md**[**i**] !=** 0**)** Md**.**push\_back**(**i**);** rep**(**i**,** 0**,** k**)** res**[**i**] =** base**[**i**] =** 0**;** res**[**0**] =** 1**;**

87

**while ((**1ll **<<** pnt**) <=** n**)** pnt**++;**

**for (**intp **=** pnt**;** p **>=** 0**;** p**--) {**

mul**(**res**,** res**,** k**);**

**if ((**n **>>** p**) &** 1**) {**

**for (**inti **=** k **-** 1**;** i **>=** 0**;** i**--)** res**[**i **+** 1**] =** res**[**i**];** res**[**0**] =** 0**;**

rep**(**j**,** 0**,** SZ**(**Md**))** res**[**Md**[**j**]] = (**res**[**Md**[**j**]] -** res**[**k**] \*** \_md**[**Md**[**j**]]) %** mod**;**

**}**

**}**

rep**(**i**,** 0**,** k**)** ans **= (**ans **+** res**[**i**] \*** b**[**i**]) %** mod**;** **if (**ans **<** 0**)** ans **+=** mod**;**

**return** ans**;**

**}**

VI BM**(**VI s**) {**

VI C**(**1**,** 1**),** B**(**1**,** 1**);**

int L **=** 0**,** m **=** 1**,** b **=** 1**;**

rep**(**n**,** 0**,** SZ**(**s**)) {**

ll d **=** 0**;**

rep**(**i**,** 0**,** L **+** 1**)** d **= (**d **+ (**ll**)**C**[**i**] \*** s**[**n **-** i**]) %** mod**;** **if (**d **==** 0**) ++**m**;**

**else if (**2 **\*** L **<=** n**) {**

VI T **=** C**;**

1. c **=** mod **-** d **\*** powmod**(**b**,** mod **-** 2**) %** mod**;** **while (**SZ**(**C**) <** SZ**(**B**) +** m**)** C**.**pb**(**0**);**

rep**(**i**,** 0**,** SZ**(**B**))** C**[**i **+** m**] = (**C**[**i **+** m**] +** c **\*** B**[**i**]) %** mod**;** L **=** n **+** 1 **-** L**;** B **=** T**;** b **=** d**;** m **=** 1**;**

**}**

**else {**

1. c **=** mod **-** d **\*** powmod**(**b**,** mod **-** 2**) %** mod**;** **while (**SZ**(**C**) <** SZ**(**B**) +** m**)** C**.**pb**(**0**);**

88

rep**(**i**,** 0**,** SZ**(**B**))** C**[**i **+** m**] = (**C**[**i **+** m**] +** c **\*** B**[**i**]) %** mod**;** **++**m**;**

**}**

**}**

**return** C**;**

**}**

int gao**(**VI a**,** ll n**) {**

VI c **=** BM**(**a**);**

c**.**erase**(**c**.**begin**());**

rep**(**i**,** 0**,** SZ**(**c**))** c**[**i**] = (**mod **-** c**[**i**]) %** mod**;**

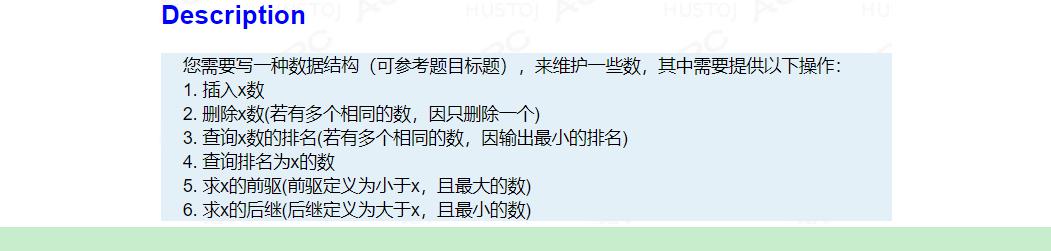
**return** solve**(**n**,** c**,** VI**(**a**.**begin**(),** a**.**begin**() +** SZ**(**c**)));**

**}**

**};**

**4、pb\_ds 库**

(1)红黑树(以 BZOJ3224 为例)



#include<bits/stdc++.h>

#include<ext/pb\_ds/assoc\_container.hpp>

89

#include<ext/pb\_ds/tree\_policy.hpp>

**using namespace** std**;**

**using namespace** \_\_gnu\_pbds**;**

**typedef** long longLL**;**

/\*g++ 4.4 以下的编译器版本第二个参数为:null\_mapped\_type\*/

**typedef** tree**<**LL**,** null\_type**,** less**<**LL**>,** rb\_tree\_tag**,** tree\_order\_statistics\_node\_update**>**

rb\_tree**;**

rb\_tree st**;**

**typedef** unordered\_map**<**LL**,**int**>** ump**;**

ump cnt**;**

int main**() {**

* freopen("input.txt", "r", stdin); int q**,** op**;**

LL x**;**

scanf**(**"%d"**, &**q**);**

**while(**q**--) {**

scanf**(**"%d%lld"**, &**op**, &**x**);**

**if(**op **==** 1**) {**

st**.**insert**(**x **<<** 32 **| (++**cnt**[**x**]));**

**}**

**else if(**op **==** 2**) {**

1. v **=** x **<<** 32 **| (**cnt**[**x**]--);**

assert**(**st**.**find**(**v**) !=** st**.**end**());** st**.**erase**(**v**);**

**}**

**else if(**op **==** 3**) {**

printf**(**"%d\n"**, (**int**)**st**.**order\_of\_key**(**x **<<** 32 **|** 1**) +** 1**);**

**}**

**else if(**op **==** 4**) {**

90

1. v **= \***st**.**find\_by\_order**(**x **-** 1**);** v **>>=** 32**;**

printf**(**"%lld\n"**,** v**);**

**}**

**else if(**op **==** 5**) {**

rb\_tree**::**iterator it **=** st**.**lower\_bound**(**x **<<** 32**);** assert**(**it **!=** st**.**begin**());** **--**it**;**

printf**(**"%lld\n"**, (\***it**) >>** 32**);**

**}**

**else if(**op **==** 6**) {**

rb\_tree**::**iterator it **=** st**.**upper\_bound**(**x **<<** 32 **| (**cnt**.**find**(**x**) !=** cnt**.**end**() ?**

cnt**[**x**] :** 0**));**

assert**(**it **!=** st**.**end**());**

printf**(**"%lld\n"**, (\***it**) >>** 32**);**

**}**

**}**

**return** 0**;**

**}**

(2)gp\_hash\_table(像数组一样使用)

#include<bits/stdc++.h>

#include<ext/pb\_ds/assoc\_container.hpp>

#include<ext/pb\_ds/hash\_policy.hpp>

**using namespace** std**;**

**using namespace** \_\_gnu\_pbds**;**

gp\_hash\_table**<**int**,**bool**>** tab**;**

int main**(){**

91

**return** 0**;**

**}**

**5、快速随机数**

#include<iostream>

#include<chrono>

#include<random>

**using namespace** std**;**

/\*mt19937:随机产生 unsigned int 范围内的整数；

mt19937\_64:随机产生 unsigned long long 范围内的整数；

\*/

unsigned test1**() {**

unsigned seed **=** chrono**::**system\_clock**::**now**().**time\_since\_epoch**().**count**();** mt19937 gen**(**seed**);** // mt19937 is a standard mersenne\_twister\_engine **return** gen**();**

**}**

**6、单样例输入对拍程序/批处理脚本**

(1)脚本：windows

@**echo** off

**:again**

dataCreator.exe**>**a.**in**

sol.exe**<**a.**in>**a1.out

bf.exe**<**a.**in>**a2.out

fc a1.out a2.out

**if %errorlevel%==**0 **goto** again

92

**pause**

(2)程序：windows

#include<bits/stdc++.h>

**using namespace** std**;**

int main**(){**

**while(**1**){**

system**(**"dataCreator.exe>a.in"**);**

system**(**"sol.exe<a.in>a1.out"**);**

system**(**"bf.exe<a.in>a2.out"**);**

**if(**system**(**"fc a1.out a2.out"**))break;**

**}**

**return** 0**;**

**}**

(3)程序：Ubuntu

#include<bits/stdc++.h>

**using namespace** std**;**

int main**(){**

**while(**1**){**

system**(**"./dataCreator>a.in"**);**

system**(**"./sol<a.in>a1.out"**);**

system**(**"./bf<a.in>a2.out"**);**

**if(**system**(**"diff a1.out a2.out"**))break;**

**else** puts**(**"AC"**);**

**}**

**return** 0**;**

**}**

93

**7、集合的整数操作汇总**

#include<bits/stdc++.h>

**using namespace** std**;**

int main**() {**

/\*1、枚举子集\*/

**for(**intn **=** 5**,** s **=** 0**;** s **<** 1 **<<** n**; ++**s**) {**

**for(**intj **=** s**;** j**;** j **= (**j **-** 1**)&**s**) {**

/\*j 为 s 的所有子集。

例如 s=11，二进制表示为 1011，那么，j 会从大到小遍历 11,10,9,8,3,2,1 \*\*/

**}**

**}**

/\*2、枚举大小为 k 的子集\*/

**for(**intn **=** 5**,** k **=** 3**,** sub **= (**1 **<<** k**) -** 1**;** sub **<** 1 **<<** n**;) {** cout **<<** bitset**<**8**>(**sub**) <<** endl**;**

int x **=** sub **& -**sub**,** y **=** sub **+** x**;**

sub **= ((**sub **& ~**y**) /** x **>>** 1**) |** y**;**

**}**

**return** 0**;**

**}**

94