

ACM/XCPC Template Sweat Boys

浙江工商大学

糖果超甜

October 27, 2021

Contents

0	text																-
	0.1	02,03					 		 	 	 		 	 		 	
	0.2	测试模板					 		 	 	 		 	 		 	
	0.3	读入输出模板															
	0.4	庆/福山宋/															
	-	main															
	0.5	main					 		 	 	 		 	 	•	 	 •
	,	/=+ -/- +m.1>i/															
1	_	(动态规划)															4
	1.1	二维费用															
	1.2	LIS 问题 nlog	n				 		 	 	 		 	 		 	 4
	1.3	SOS dp					 		 	 	 		 	 		 	 4
	1.4	SOSDP					 		 	 	 		 	 		 	
	1.5	SOSDP2															
	1.6	背包问题方案															
	1.7	背包最优方案															
	1.8	单调队列 dp															
	1.9	多重背包															
		分组背包															
		混合背包															
	1.12	区间 dp					 		 	 	 		 	 		 	 (
		四边形优化															
		完全背包															
			 / tal #7/														
		有依赖的背包		- /													
	1.18	01 背包					 		 	 	 		 	 		 	 1:
_																	
2	树形																13
	2.1	lca(倍增)															
	2.2	lca(树链剖分	·)				 		 	 	 		 	 		 	 1:
	2.3	次小生成树					 		 	 	 		 	 		 	 1:
	$^{2.4}$	点分治					 		 	 	 		 			 	
	2.4	点分治 树的光经															
	2.5	树的半径					 		 	 	 		 	 		 	 18
	$\frac{2.5}{2.6}$	树的半径 树的直径				· · ·	 		 	 	 		 	 		 	 18 19
	2.5 2.6 2.7	树的半径 树的直径 树哈希				 	 		 	 	 		 	 		 	 18 19 19
	2.5 2.6 2.7 2.8	树的半径 树的直径 树哈希 树链剖分				 	 	· · · · · ·	 · · · · · ·	 	 	· · · · · · · · · · · · · · · · · · ·	 	 		 	 18 19 19 20
	2.5 2.6 2.7 2.8 2.9	树的半径 树的直径 树哈希 树链剖分 树上 k-th 祖夕	 . O(q+	 -nlogi	 n) .		 	· · · · · · · · · · · · · · · · · · ·	 	 	 		 	 		 	 18 19 20 21
	2.5 2.6 2.7 2.8 2.9 2.10	树的半径 树的直径 树哈希 树链剖分 树上 k-th 祖外树上差分 (边边	· · · · · · · · · · · · · · · Ē O(q+	 . nlogi	· · · · · · · · · · · · · · · · · · ·		 		 · · · · · · · · · · · · · · · · · · ·	 · · · · · · · · · ·	 		 	 		 	 18 19 19 20 21 23
	2.5 2.6 2.7 2.8 2.9 2.10	树的半径 树的直径 树哈希 树链剖分 树上 k-th 祖夕	· · · · · · · · · · · · · · · Ē O(q+	 . nlogi	· · · · · · · · · · · · · · · · · · ·		 		 · · · · · · · · · · · · · · · · · · ·	 · · · · · · · · · ·	 		 	 		 	 18 19 19 20 21 23
	2.5 2.6 2.7 2.8 2.9 2.10 2.11	树的半径 树的直径 树的高者 树链剖分 树上 k-th 祖外树上差分 (边线树上距离 (lca	 	 -nlog1 			 		 	 	 		 	 		 · · · · · · · · · · · · · · · · · · ·	 18 19 20 21 21 22 22
	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12	树的半径 树的直径 树哈希 树链剖分 树上 k-th 祖分 树上差分(边刻 树上距离(lca 树上距离(lca	 	· · · · · · · · · · · · · · · · · · ·			 		 	 	 		 	 		 · · · · · · · · · · · · · · · · · · ·	 18 19 20 21 21 22 22 23
	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13	树的半径 树的直径 树哈希 树连部分 树上差分(边刻 树上距离(lea 树上距宽发式合	·····································				 		 	 	 		 			 	 18 19 20 21 23 23 24 24 26
	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13	树的半径 树的直径 树哈希 树连部分 树上差分(边刻 树上距离(lea 树上距宽发式合	 				 		 	 	 		 			 	 18 19 20 21 23 23 24 24 26
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14	树的半径···树的直播。 树的直希···树性的一种树的一种, 树上是一种树上, 树上是一种树上, 树上, 树上, 树上, 树上, 树上, 树上, 树上, 树上, 树上,	·····································				 		 	 	 		 			 	 18 19 20 21 22 22 24 26 27
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14	树的半径··· 树的的直希引生。 树性上。 树上上, 树上, 树上, 树上, 大叶, 树上, 大叶, 树上, 大叶, 树木, 大叶, 树木, 大叶, 大叶, 大叶, 大叶, 大叶, 大叶, 大叶, 大叶, 大叶, 大叶	 . tarjan . 倍增) 并				 				 		 				 18 19 20 23 23 24 26 27 28
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 数据 3.1	树的半径 树的直希 树性。hth 祖妇 树上是一种树上差距离(lea 树上距启发(lea 树上组祖 林级 级 格数 数 数 数 数 数 数 数 数 数 数 数 数 数 数 数 数 数 数	 . O(q+ . tarjan . 倍增) 并														18 19 20 21 22 22 23 24 26 27 28 28 28
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 数据 3.1 3.2	树的半径 树的直径 树的直希 树上 树上 树上是 树上距 树上距 的 树上 的 树上 的 树上 的 树上 的 树上 的 树上 的	 E O(q+ 差分). tarjan 倍增) 并														18 19 20 21 22 22 24 26 27 28 28 28
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3	树的半径 树的直看 树的奇希 树性 k-th (边域 树上差距离 (lca 树上启超发 树上启动 k 级 Max XOR(tir splay(待完善) ST 表	 e)														18 19 19 19 19 19 19 19 19 19 19 19 19 19
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3 3.4	树的半径 树的直看 树的奇希 树性 k-th (边域 树上差距离 (lca 树上启超发 树上启祖先 k 级 Max XOR(tir splay(待完善) ST 表	 e)														18 19 19 20 22 22 22 26 27 28 28 33 33 33
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3	树的半径 树的首都 树性。hth (边域 树上 k-th (边域 树上差距离(lca 树上距启发 树上的高离式合k k 级 Max XOR(tir splay(待完善) ST表 Treap(文艺	 e)														18 19 19 20 22 22 22 26 27 28 28 33 33 33
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3 3.4	树的半径 树的首都 树性。 树性。 树上。 树上。 树上。 树上。 树上。 树上。 树上。 树上。 以(lca 树上。 的高。 以之。 以之。 以之。 以之。 以之。 以之。 以之。 以之。 以之。 以之	 e) e)														18 19 19 20 22 22 22 22 22 22 22 22 23 33 33 34
3	2.5 2.6 2.7 2.8 2.9 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3 3.4 3.5	树的半径··· 树的直看··· 树的后希··· 树树上。k-th (i) (lea 树上差距距启说 树上距启说 k 构 Max XOR(tir splay(待完善) ST 表 ··文 ··· 并查集															18 19 19 20 22 23 24 26 22 25 26 27 28 28 33 33 33 33 36 36 36
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 数据 3.2 3.3 3.4 3.5 3.6 3.7	树的半径··· 树的自希··· 树的哈希··· 树树的一个 树上,是一个 树上,是一个 树上,是一个 树上,是一个 树上,是一个 大人。 大人。 大人。 大人。 大人。 大人。 大人。 大人。 大人。 大人。															18 19 19 20 22 22 25 26 27 28 28 33 34 36 33 36 37
3	2.5 2.6 2.7 2.8 2.9 2.10 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8	树的半径··· 树的哈希别·k-th (ilca 树性上差距距启祖 林上差距距启祖 从Max XOR(tir splay(大)。 ST · Treap(集) 以 证 可 等 以 证 等 明 形 是 是 的 是 是 的 是 是 的 是 是 的 是 是 的 是 是 的 是 是 的 是 是 的 是 是 的 是 是 的 是 是 的 是 是 的 是 是 。 一 、 一 、 一 、 一 、 一 、 一 、 一 、 一 、 一 、 一															18 19 20 22 22 22 22 22 22 22 23 33 34 36 33 38 38 38
3	2.5 2.6 2.7 2.8 2.9 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9	树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树	·····································														18 19 20 22 25 26 22 26 22 28 33 34 36 37 38 38 38 38
3	2.5 2.6 2.7 2.8 2.9 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10	树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树	·····································														18 19 19 20 22 22 22 22 22 22 22 22 22 22 22 22
3	2.5 2.6 2.7 2.8 2.9 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.10 3.11	树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树	·····································														18 19 19 20 22 22 25 26 27 28 33 33 34 36 36 36 36 36 46
3	2.5 2.6 2.7 2.8 2.9 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.11 3.11	树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树															18 19 20 22 22 22 22 24 26 27 28 33 34 36 37 38 38 38 38 46 44 42
3	2.5 2.6 2.7 2.8 2.9 2.11 2.12 2.13 2.14 数据 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 3.11 3.11	树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树树															18 19 20 22 22 22 22 24 26 27 28 33 34 36 37 38 38 38 38 46 44 42

ACM/XCPC Template Sweat Boys, **糖果超甜**

	3 15	扫描线		45
		树状数组・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・		47
	3.17	树状数组套主席树		48
	3.18	替罪羊树(平衡树)		51
	3.19	线段树(区间最长连续段)		53
		线段树 (最长子序和)		53
		线段树		55
		线段树 +dfs 序		57
	3.23	线段树标记永久化		57
		线段树合并 + 动态开点		58
		Hash 表		61
	5.25	Hash 🛠 · · · · · · · · · · · · · · · · · ·	 •	01

4	数学			63
	4.1	EX_CRT		63
	4.2	EX_Lucas		63
		Lucas		65
		Miller _R abin		66
	4.5	Min25 筛		67
	4.6	杜教筛		68
	4.7	杜教筛		69
	4.8	二次剩余		70
	4.9	反演相关		71
	4.10	斐波那契		72
	4.11	光速幂		72
	4.12	康托展开		72
		快速幂		73
		莫比乌斯反演		74
		逆元		74
	4.16	欧拉函数		75
	4.17	欧拉降幂		76
		- 10-11-11-11-11-11-11-11-11-11-11-11-11-1		76
		算术基本定理		76
		质数、积性函数		77
	4.21	线性筛		78
	4.22	线性同余方程		79
		原根		79
		原根表		81
		整除分块		82
	4.26	整数拆分		82
	4.27	整数分块		83
		BSGS		83
		Simpson 积分		84
				_
		2 维计算几何		84
	4.31	3 维计算几何		98
	4.32	三分套三分		102
		三维几何		102
		1,7 = 1,2		108
		多项式 $\ln_e x p_p o w$ 1		
		二次剩余处理边界不为 1		109
	4.36	x 不连续、暴力插值		111
	4.37	x 连续、前缀优化		112
		FFT 加速带有通配符字符串匹配		
	4.90	FFT 加速帯有通配符字符串匹配 2	 •	114
		FFT 加速朴素字符串匹配		
		CDQ_FFT		116
		FFT		118
		FWT		_
		NTT		
		高斯消元		
	4.46	高斯消元 2		125

ACM/XCPC Template Sweat Boys, **糖果超甜**

		高斯消元异或	
		矩阵快速幂	
		矩阵求逆	
		线性基	
		线性基	
		康托展开	
		模数非质数的组合	
		普通型母函数	
		生成函数	
		斯特林数	
		五边形数	
		指数型母函数	
		组合数学基础	
		组合数	
	4.61	卡特兰数	. 140
5	除右★□	化算法	141
J		で弄ね SA2	
	0.1		. 171
6	图论		143
	6.1	DAG 图上建支配树	. 143
	6.2	spfa 判负环 (01 分数规划)	. 144
	6.3	边双连通分量 +e-dcc 缩点	. 146
	6.4	差分约束系统	. 147
	6.5	点双连通分量 +v-dcc 缩点	. 148
	6.6	二分图 HK 算法	. 149
		二分图 KM 算法	
	6.8	二分图匈牙利算法	. 152
	6.9	割边	. 152
	6.10	割点	. 153
	6.11	克鲁斯卡尔重构树	. 153
		克鲁斯卡尔重构树上建主席树	
		欧拉图	
		强联通分量	
		限制流量费用流 (Dijkstra)	
		一般图最大匹配 (带花树)	
		最大独立集 (一般图)	
		最大流 (Dinic)	
		最大团 (一般图)	
		最短路	
		最小费用最大流 (SPFA)	
		最小路径生成树。	
		最小生成树 boruvka	
	6.24	最小生成树 prim	. 168
7	杂项		170
•		DEBUG	
	7.2	MODINT	
	7.3	大数模板	
	7.4	待修莫队	
	7.5	对拍	
	7.6	二进制压缩枚举子集	
	7.7	二维差分	
	7.8	<u> </u>	
	7.9	高精度	
		环形均分	
		回滚莫队	
		离散化	

ACM/XCPC Template Sweat Boys, **糖果超甜**

	7.14 普通莫队	. 189
8	字符串	193
	3.1 (SAM)k 长子串最大出现次数	. 195
	8.2 (SAM) 第 k 小子串	. 196
	8.3 (SAM) 多串 lcs	. 198
	8.4 ac 自动机	
	8.5 kmp	. 201
	8.6 二维字符串哈希	
	8.7 后缀自动机	. 202
	8.8 回文自动机	
	8.9 拓展 kmp	
	8.10 字典树	. 206
	8.11 字符串哈希	. 207
	8.12 最小表示法	. 207

```
0
        text
   0.1 \quad 02.03
   #pragma GCC optimize(3,"Ofast","inline")
   #pragma GCC optimize(2)
   0.2
         测试模板
1
   signed main() {
        ios_base::sync_with_stdio(false);
2
3
        cin.tie(0);
        cout.tie(0);
 4
   #ifdef ACM_LOCAL
5
        freopen("in.txt", "r", stdin);
6
        //freopen("out.txt", "w", stdout);
7
        signed test_index_for_debug = 1;
8
        char acm_local_for_debug = 0;
9
        do {
10
            if (acm_local_for_debug == '$') exit(0);
11
            if (test_index_for_debug > 20)
12
                throw runtime_error("Check the stdin!!!");
13
            auto start_clock_for_debug = clock();
14
15
            solve();
            auto end_clock_for_debug = clock();
16
            cout << "Test " << test_index_for_debug << " successful" << endl;</pre>
17
            cerr << "Test " << test_index_for_debug++ << " Run Time: "</pre>
18
                << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC <<</pre>
19
       "s" << endl;
            cout << "-
20
        } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
21
   #else
22
        solve();
23
   #endif
24
25
        return 0;
26
   }
   0.3 读入输出模板
   inline int read(){
       int s=0, w=1;
2
       char ch=getchar();
3
      while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
 4
      while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();</pre>
5
       return s*w;
6
   }
7
8
9
   namespace StandardIO {
10
        inline char nc() {
11
12
            static char buf[1000000],*p1=buf,*p2=buf;
            return p1==p2&&(p2=(p1=buf)+fread(buf,1,1000000,stdin),p1==p2)?E0F:*p1++;
13
14
        template <typename _Tp> inline void read(_Tp&sum) {
15
            char ch=nc();sum=0;
16
            while(!(ch>='0'&&ch<='9')) ch=nc();</pre>
17
```

```
while(ch>='0'&&ch<='9') sum=(sum<<3)+(sum<<1)+(ch-48), ch=nc();</pre>
18
        }
19
20
        template<typename T>
21
        inline void write(T x) {
22
            if (x < 0) putchar('-'), x *= -1;
23
            if (x >= 10) write(x / 10);
24
25
            putchar(x % 10 + '0');
26
        }
   }
27
30 inline __int128 read()
31
       int X=0, w=0; char ch=0;
32
      while(!isdigit(ch)) {wl=ch=='-';ch=getchar();}
33
      while(isdigit(ch)) X=(X<<3)+(X<<1)+(ch^48), ch=getchar();
34
35
       return w?-X:X;
36 }
37 inline void print(__int128 x)
38 {
       if(x<0){putchar('-');x=-x;}</pre>
39
       if(x>9) print(x/10);
40
       putchar(x%10+'0');
41
42 }
   0.4
 1 /***0 0 10 ***/
 2 /*SiberianSquirrel*//*CuteKiloFish*/
 3 #include <bits/stdc++.h>
 4 //#include<bits/extc++.h>
 5 #include<ext/rope>
6 #include<ext/pb_ds/assoc_container.hpp>
 7 #include<ext/pb_ds/tree_policy.hpp>
 8 using namespace std;
9 using namespace __gnu_cxx;
10 using namespace __gnu_pbds;
11 #define Inv(x) quick_pow(x, mod - 2)
12 #define Polynomial vector<int>
#define DEBUG(x, y) cout << x << ": " << y << '\n'; #define mem(a, x) memset(a, x, sizeof a)
15 #define right_1_pos(x) __builtin_ffs(x)
16 #define left_0_num(x) __builtin_clz(x)
17 #define right_0_num(x) __builtin_ctz(x)
18 #define num_of_1(x) __builtin_popcount(x)
19 #define Pii pair<int, int>
20 #define mp_(x, y) make_pair(x, y)
21 #define all(v) (v).begin(), (v).end()
22 using ld = long double;
23 using ll = long long;
24 //using ill = __int128;
25 using ull = unsigned long long;
26 using i16 = short;
27 const ld pi = acos(-1.0);
28 const ld eps = 1e-8;
29 \quad const \ ll \ mod = 998244353, \ mod_q = 3, \ imq = 86583718;
30 int inv2, inv3;
```

31 tree<ll, null_type, less<ll>, rb_tree_tag, tree_order_statistics_node_update> tr;

0.5 main

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
 4 typedef long long ll;
5 typedef pair<ll, ll> PII;
6 typedef unsigned long long ull;
7 const ll inf = 1e18;
8 \text{ const int } N = 2e6 + 10;
9 const int M = 1e6 + 10;
10 const double eps = 1e-8;
11 const int mod = 1e9 + 7;
12
13 #define fi first
14 #define se second
15 #define re register
16 #define lowbit (-x&x)
17
18 signed main() {
19
        ios_base::sync_with_stdio(false);
20
        cin.tie(0);
        cout.tie(0);
21
   #ifdef ACM_LOCAL
22
        freopen("input", "r", stdin);
freopen("output", "w", stdout);
23
24
   #endif
25
26
   #ifdef ACM_LOCAL
27
28
        auto start = clock();
29
   #endif
30
        int t = 1;
31
          cin >> t;
32
        while (t--)
33
            solve();
   #ifdef ACM_LOCAL
34
        auto end = clock();
35
        cerr << "Run Time: " << double(end - start) / CLOCKS_PER_SEC << "s" << endl;</pre>
36
37 #endif
38
        return 0;
39 }
```

1 dp (动态规划)

1.1 二维费用

15

16

17

signed main() {

ios::sync_with_stdio(false);

```
时间复杂度 0(n^3)
  f[i][j] = max(f[i][j], f[i-a]+f[j-b] + w);
   1.2 LIS 问题 nlogn
1
2
3
   void solve() {
4
       while (cin \gg a[++n]);
5
       n--;
6
       for (int i = 1; i \le n; i++) {
            if (a[i] > b[len1]) b[++len1] = a[i];
7
            else {
8
                int pos = lower_bound(b + 1, b + len1 + 1, a[i]) - b;
9
10
                b[pos] = a[i];
11
            }
       }
12
13
14
       reverse(a+1, a+n+1);
15
       for (int i = 1; i <= n; i++) {
16
            if (a[i] >= b[len2]) b[++len2] = a[i];
17
                int pos = upper_bound(b + 1, b + len2 + 1, a[i]) - b;
18
                b[pos] = a[i];
19
           }
20
       }
21
       cout << len2 << endl;</pre>
22
       cout << len1 << endl;</pre>
23
24 }
25
26
  */
27
   1.3 SOS dp
1
   //
  // Created by SANZONG on 2021/7/21.
4 //
5 #include "bits/stdc++.h"
6
7 #define int long long
8 using namespace std;
9 const int N = 1 << 20;
10 const int mod = 998244353;
11 int a[N], b[N];
12 int c[N];
13 int d1[N], d2[N];
14 int d3[N], d4[N];
```

```
int T;
18
                   cin >> T;
19
                   while (T--) {
20
21
                              int n;
22
                              cin >> n;
23
                              for (int i = 0; i < n; ++i) {
                                        cin >> a[i];d1[i] = a[i];d2[i] = a[i];
24
25
                              for (int i = 0; i < n; ++i) {
26
                                        cin >> b[i];d3[i] = b[i];d4[i] = b[i];
27
28
29
                              c[n] = -(111 \iff 62);
                              for (int i = n - 1; i >= 0; i--) {
30
                                                                                                                                                  //枚举每个数
                                        for (int j = 0; (1 << j) < n; j++) {
31
                                                   if (i & (1 << j)) {
32
                                                                                                                                                //属于子集
                                                            d1[i \land (1 << j)] = max(d1[i \land (1 << j)], d1[i ]);
                                                                                                                                                                                                              //求k<=i中k
33
                  能对应的最大的a[i],注意转移
                                                            d2[i \land (1 << j)] = min(d2[i \land (1 << j)], d2[i]);
34
                                                            d3[i \land (1 << j)] = max(d3[i \land (1 << j)], d3[i ]);
35
                                                            d4[i \land (1 << j)] = min(d4[i \land (1 << j)], d4[i ]);
36
                                                   }
37
                                        }
38
                              }
39
                              int ans = 0;
40
41
                              for (int i = n - 1; i >= 0; --i) {
                                        c[i] = max({c[i + 1], d2[i] * d4[i], d1[i] * d3[i], d1[i] * d4[i], d2[i] * d4[i], d2[i], d2[i],
42
                  d3[i]});
43
                                        ans = (ans + c[i]) \% mod;
44
45
                              cout << (ans + mod) % mod << endl;</pre>
46
                   }
47 }
         1.4 SOSDP
  1 const int bit = 20;
        const int mx = 1 \ll bit \mid 1;
        void sosdp(vector<int> &a, vector<int> &dp, int n) {
 4 //
                         for(int i = 0; i \le n; ++ i) dp[i] = 0;
                   for(int i = n; i >= 0; ++ i) {
 5
                              for(int j = 0; j < bit; ++ j) {
 6
                                        if((i \mid (1 \lessdot j)) > n) continue;
 7
                                        dp[i] = (dp[i] + dp[i | (1 << j)]) \% mod;
 8
 9
                              }
10
                   }
11 }
         1.5 SOSDP2
 1 const int bit = 19;
  2 const int mx = 1 \ll bit \mid 1;
 3 11 A[mx], F[mx], B[mx];
 4 ll maxx[2][mx], minn[2][mx];
 5
 6
        void solve() {
                   int o; cin >> o; while(o--) {
 7
                              int n; cin >> n;
```

```
for (int i = 0; i < n; ++ i) {
9
               cin >> A[i];
10
               \max x[0][i] = \min[0][i] = A[i];
11
12
           for (int i = 0; i < n; ++ i) {
13
14
               cin >> B[i];
               \max x[1][i] = \min [1][i] = B[i];
15
           }
16
17
18
           for(int j = mx; j >= 0; -- j) {
               for(int i = 0; i < bit; ++ i) {</pre>
19
20
                   if((j \mid (1 \ll i)) >= n) continue;
                   \max[0][j] = \max(\max[0][j], \max[0][j \mid (1 << i)]);
21
                   \max [1][j] = \max(\max[1][j], \max[1][j \mid (1 \ll i)]);
22
                   minn[0][j] = min(minn[0][j], minn[0][j | (1 << i)]);
23
                   minn[1][j] = min(minn[1][j], minn[1][j | (1 << i)]);
24
               }
25
26
           }
           ll maxxx = -2e18, res = 0;
27
           for(int i = n - 1; i >= 0; -- i) {
28
               maxxx = max(maxxx, maxx[0][i] * maxx[1][i]);
29
               maxxx = max(maxxx, maxx[0][i] * minn[1][i]);
30
               maxxx = max(maxxx, minn[0][i] * maxx[1][i]);
31
32
               maxxx = max(maxxx, minn[0][i] * minn[1][i]);
33
               res = (res + maxxx) \% mod;
               while(res < 0) res += mod;</pre>
34
35
           }
           cout << res << '\n';</pre>
36
       }
37
  }
38
        背包问题方案
   如果要求字典序最小
1
2
   n --> 1 做一遍普通的背包
3
4
   1 --> n 取一遍方案, 如果f[i][cur] = f[i+1][cur-v]+w, 那么i就是方案
5
6
7
   字典序最大, 反过来即可
8
   时间复杂度0(nm)
   1.7 背包最优方案计数问题
1
   可以不装满背包,保证总价值最大
2
  f[N]记录价值, cnt[N]记录个数
3
4
   f[j] = max(f[j], f[j-v]+w);
5
6
   如果当前f[j-v] + w > f[j],说明是f[j-v]+w更新的答案,因此cnt[j] = cnt[j-v];
7
8
9
   如果当前f[j-v] + w = f[j], 说明两者都可, cnt[j] += cnt[j-v];
10
   别忘了cnt初始化为 1,因为背包可以空着也算一种方案
11
12
   时间复杂度0(nm)
13
```

1.8 单调队列 dp

```
2 // Created by SANZONG on 2021/9/17.
3 //
4
5 //
6 // Created by SANZONG on 2020/8/10.
7 //
8 //lca模板
9
10 #include "bits/stdc++.h"
11 #define int long long
12 const int maxn = 1e6;
13 using namespace std;
14 int dp[maxn];
15 int sum[maxn];
16 int q[maxn];
17 int a[maxn];
   signed main() {
19
        int n,k;
20
        cin >> n >> k;
        for (int i = 1; i <= n; ++i) {
21
            int s;cin >> s;
22
            sum[i] = sum[i-1] + s;
23
24
        int head = 1;
25
26
        int tail = 0;
27
        q[++tail] = 0;
28
        for (int i = 1; i <= n; ++i) {
            dp[i] = a[q[head]] + sum[i];
29
30
            a[i] = dp[i-1]-sum[i];
            while (head \leftarrow tail && i - q[head] + 1 > k)
31
32
                head++;
            while (head <= tail && a[q[tail]] < a[i])</pre>
33
                tail--;
34
            q[++tail] = i;
35
36
        cout << dp[n] << endl;</pre>
37
  }
38
```

1.9 多重背包

```
1. 朴素算法
3
   时间复杂度0(n^3)
4
   for (int i = 1; i <= n; i++)
5
       for (int j = m; j >= v[i]; j--)
6
           for (int k = 1; k \le s[i]; k++)
7
               f[j] = max(f[j], f[j - k*v[i]] + k*w[i]);
8
9
10
   核心 f[j] = max(f[j], f[j-v] + w, f[j-v*2] + 2*w, f[j-3*v] + 3*w ....)
11
   2. 二进制优化
12
13
   时间复杂度0(nm*log(n))
14
15
```

```
例如: 10 = 1 + 2 + 4 + 3
16
17
18
   将问题转换为01背包问题
19
   3.单调队列优化
20
21
22 时间复杂度0(nm)
23
  f[0] = max(f[0], f[v] + w, f[2*v] + 2*w, f[3*v] + 3*w...)
24
  f[1] = max(f[1], f[1+v] + w, f[1+2*v] + 2*w, f[1+3*v] + 3*w...)
  f[2] = max(f[2], f[2+v] + w, f[2+2*v] + 2*w, f[2+3*v] + 3*w...)
27
28 f[v-1] = max(f[v-1], f[v-1+v] + w, f[v-1+2*v] + 2*w, f[v-1+3*v] + 3*w...)
29
30 所有的 m 都可以转换为 c + a * x,而且是互不影响的
31
32 f[j] = f[j]
33 f[j+v] = \max f[j+v], f[j] + w
34 f[j+2*v] = max f[j+2*v], f[j+v] + w, f[j] + 2*w
35
36 修改一下变成
37 f[j] = f[j]
38 f[j+v] = max(f[j+v] - w, f[j]) + w
  f[j+2*v] = max(f[j+2*v] - 2*w, f[j+v] - w, f[j]) + 2*w
40
   将 f[j + k*w] - k*w 存入单调队列中, 取值时用加上 k*w
41
42
43
   #include <bits/stdc++.h>
   using namespace std;
44
45
46 const int N = 20010;
47
  int f[N], g[N], n, m;
48
  int q[N];
49
50
   int main() {
51
52
       cin >> n >> m;
53
       for (int i = 1; i \le n; i++) {
54
           int v, w, s;
           cin >> v >> w >> s;
55
           memcpy(g, f, sizeof f);
56
           for (int j = 0; j < v; j++) {
57
               int hh = 1, tt = 0;
58
               for (int k = j; k \le m; k += v) {
59
                   while (hh <= tt && k - q[hh] > s*v) hh++; //队列中最多只能有s+1个值
60
                   if (hh \leftarrow tt) f[k] = max(f[k], g[q[hh]] + (k - q[hh]) / v * w);
61
                   while (hh <= tt && g[q[tt]] - (q[tt] - j) / v * w <= g[k] - (k - j) / v
62
        * w) tt--;
                   q[++tt] = k;
63
               }
64
65
           }
66
67
       cout << f[m] << endl;</pre>
68 }
```

1.10 分组背包

1 v[0....n]代表组

```
f[j] = \max(f[j], f[j-v[0]]+w[0], f[j-v[1]]+w[1], f[j-v[2]]+w[2]....)
   枚举三层循环即可
   1.11 混合背包
   假设存在无限个,一个,有限个的物品
1
2
3
   将多重背包用二进制优化转换为01背包
4
   接着按照完全背包和01背包做就行
5
   1.12
          区间 dp
   #include <bits/stdc++.h>
   using namespace std;
3
   typedef long long 11;
   typedef unsigned long long ull;
5 #define ACM_LOCAL
6
7
  const int N = 200 + 5;
   const int Mod = 1e9 + 7;
   int n, s[N], f1[N][N], f2[N][N];
10
   void solve() {
11
12
       cin >> n;
       for (int i = 1; i <= n; i++) cin >> s[i], s[i+n] = s[i];
13
       for (int i = 1; i <= 2*n; i++) s[i] += s[i-1];
14
15
       for (int len = 2; len <= n; len++) {</pre>
16
            for (int i = 1; i + len - 1 <= 2*n; i++) {
17
                int j = i + len - 1;
18
                f1[i][j] = 1e8;
19
                for (int k = i; k < j; k++) {
20
                    f1[i][j] = min(f1[i][j], f1[i][k] + f1[k+1][j] + s[j] - s[i-1]);
21
22
                    f2[i][j] = max(f2[i][j], f2[i][k] + f2[k+1][j] + s[j] - s[i-1]);
23
                }
24
            }
25
       int Max = 0, Min = 1e8;
26
       for (int i = 1; i + n - 1 \le 2*n; i++) {
27
           Max = max(Max, f2[i][i+n-1]);
28
           Min = min(Min, f1[i][i+n-1]);
29
30
31
       cout << Min << endl;</pre>
       cout << Max << endl;</pre>
32
   }
33
34
   signed main() {
35
       ios_base::sync_with_stdio(false);
36
37
       cin.tie(0);
38
       cout.tie(0);
   #ifdef ACM_LOCAL
39
       freopen("in.txt", "r", stdin);
freopen("out.txt", "w", stdout);
40
41
   #endif
```

```
43
       solve();
       return 0;
44
  }
45
          数位 dp
   1.13
   ll dfs(int pos, int pre, bool limit, bool lead) {
       if (pos == -1) return 1;
2
3
       if (!limit && !lead && dp[pos][...][...] != -1) return dp[pos][...][...];
4
       int End = limit ? a[pos] : 9;
       ll ans = 0;
5
       for (int i = 0; i \le End; i++) {
6
           *****//操作1
7
8
       if (!limit && !lead) dp[pos][...][...] = ans;
9
10
       return ans;
11
   ll calc(ll x) {
12
       int pos = 0;
13
       while (x) {
14
           a[pos++] = x \% 10;
15
16
           x /= 10;
17
       return dfs(pos-1, 0, true, true);
18
  }
19
   1.14
          四边形优化
1
  //
2 // Created by acer on 2020/11/23.
3 //
4
  //m(i,j)=min\{m(i,k-1),m(k,j)\}+w(i,j)(i \le k \le j)
   //满足对于 i≤i'<j≤j', 有 w(i',j)≤w(i,j'), 且对于 i≤i'<j≤j', 有 w(i,j)+w(i',j')≤w(i',j)+w(i,
   //则m函数具有相同性质,故可以简单的推出m的最优决策点有s(i,j) \le s(i,j+1) \le s(i+1,j+1)
7
   int main() {
8
       int f[100][100];
9
       int m[100][100];
10
11
       int n, INF = 100;
       for (int i = 1; i \le n; ++i) {
12
13
           m[i][i] = i;
14
       for (int len = 2; len <= n; ++len) // 枚举区间长度
15
           for (int l = 1, r = len; r <= n; ++l, ++r) { // 枚举长度为len的所有区间
16
               f[l][r] = INF;
17
               for (int k = m[l][r - 1]; k <= m[l + 1][r]; ++k)
18
                   if (f[l][r] > f[l][k] + f[k + 1][r] + w(l, r)) {
19
20
                       f[l][r] = f[l][k] + f[k + 1][r] + w(l, r); // 更新状态值
                       m[l][r] = k; // 更新(最小) 最优决策点
21
22
                   }
           }
23
24 }
```

1.15 完全背包

```
f[i][j] = max(f[i-1][j], f[i-1][j-v[i]] + w[i], f[i-1][j-2*v[i]] + 2*w[i], ....)
   f[i][j - v[i]] + w[i] = max(f[i-1][j-v[i]] + w[i], f[i-1][j-2*v[i]] + 2*w[i], .....)
3
   f[i][j] = max(f[i-1][j], f[i-1][j-v[i]]+w[i])
5
   ==> for (int i = v[i]; i <= m; i++)
7
8
   因此 j 使用当前的状态更新自己, 所以正序
   1.16
         悬线法
1 #include<cstdio>
2
3 #define N 2005
4 #define max(a, b) a>b?a:b
5 #define min(a, b) a<b?a:b</pre>
6 using namespace std;
7
  int up[N][N], left[N][N], right[N][N], ansa, ansb, a[N][N], m, n;
   int main() {
9
       scanf("%d%d", &n, &m);
10
11
       for (int i = 1; i <= n; i++)
           for (int j = 1; j <= m; j++)
    up[i][j] = 1, left[i][j] = j, right[i][j] = j, scanf("%d", &a[i][j]);// up</pre>
12
13
       初值,读入,left/right 最初值
14
       for (int i = 1; i <= n; i++)
15
           for (int j = 2; j <= m; j++)
16
               if (a[i][j] ^ a[i][j - 1])
17
                    left[i][j] = left[i][j - 1];
18
19
       for (int i = 1; i <= n; i++)
           for (int j = m; j > 1; j--)
20
               if (a[i][j] ^ a[i][j - 1])
21
                    right[i][j - 1] = right[i][j];//left/right初值,即(i,j)点向左/右的最大宽
22
       度
23
       for (int i = 1; i <= n; i++)
24
           for (int j = 1; j <= m; j++) {
25
               if (i > 1 && a[i][j] ^ a[i - 1][j])
26
                    up[i][j] = up[i - 1][j] + 1, left[i][j] = max(left[i][j],left[i - 1][j
27
       ]), right[i][j] = min(right[i][j], right[i - 1][j]);
               int a = right[i][j] - left[i][j] + 1;
28
               int b = min(a, up[i][j]);
29
30
               ansa = max(ansa, b * b);
               ansb = max(ansb, a * up[i][j]);
31
32
       printf("%d\n%d", ansa, ansb);
33
34 }
          有依赖的背包 (树形 dp)
   1.17
1
   类似分组背包的思想
2
3
   for (int j = m; j >= v[i])
       for (int k = 0; k <= j; k++)
4
           f[x][j] = max(f[x][j], f[x][j-k] + f[y][k]);
5
```

```
for (int i = m; i >= v[x]; i--) f[x][i-v[x]] + w[x]; for (int i = 0; i < v[x]; i++) f[x][i] = 0; 每个子节点为一个组,组内不同的背包容量转移为组内的不同元素

1.18 01 背包

f[i][j] = max(f[i-1][j], f[i-1][j-v[i]] + w[i])

B为 i 只用到前后的两次状态,因此可以压缩成一维的空间

f[j] = max(f[j], f[j-v[i]] + w[i])

并且 for (int j = m; j >= v[i]; j--)

B为f[j[v[i]]] + w[i]要用到i-1维的数据更新当前背包空间,所以j倒序
```

2 树形结构

2.1 lca(倍增)

```
#include <bits/stdc++.h>
   using namespace std;
   const int N = 1e5 + 10;
3
   vector<int> g[N];
5
6
   int f[N][30], dep[N];
7
   void dfs(int u, int fa) {
8
       dep[u] = dep[fa] + 1; f[u][0] = fa;
9
       for (int i = 1; i \le 19; i++) f[u][i] = f[f[u][i-1]][i-1];
10
       for (auto &v : g[u]) {
11
12
            if (v == fa) continue;
            dfs(v, u);
13
       }
14
   }
15
16
   int lca(int x, int y) {
    if (dep[x] > dep[y]) swap(x, y);
17
18
       for (int i = 19; i >= 0; i--) if (dep[y] - dep[x] >= (1 << i)) <math>y = f[y][i];
19
20
       if (x == y) return x;
21
       for (int i = 19; i >= 0; i--) {
22
            if (f[y][i] != f[x][i])
23
24
                y = f[y][i], x = f[x][i];
25
       return f[x][0];
26
27
   }
   2.2 lca (树链剖分)
   int lca(int u, int v) {
1
       while (top[u] != top[v]) {
2
3
            if (d[top[u]] > d[top[v]])
                u = fa[top[u]];
4
5
            else
6
                v = fa[top[v]];
7
       return d[u] > d[v] ? v : u;
8
   }
9
   2.3 次小生成树
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define INF 1e16
4 const int N = 1e5 + 200;
5 const int M = 6 * 1e5 + 300;
   int head[M], edge[M], Next[M], ver[M], tot, fa[M], n, m, father[N][32], deep[N];
7
   long long dp[2][N][32], val1, val2, ans_max, ans;
8
9
  struct node
10
   {
       int x, y, z, vis;
11
```

```
12 } s[M];
13
   int cmp(node a, node b)
14
15
   {
16
        return a.z < b.z;</pre>
17
   }
18
   struct Edge {
19
        void init2() {
20
            memset(head, 0, sizeof(head));
21
22
            tot = 0;
23
        }
        void add_edge(int a, int b, int c) {
24
            edge[++tot] = b;
25
            ver[tot] = c;
26
            Next[tot] = head[a];
27
            head[a] = tot;
28
29
30
        int find(int x) {
            return x == fa[x] ? x : fa[x] = find(fa[x]);
31
32
        void Kruskal() {
33
            sort(s + 1, s + 1 + m, cmp);
34
35
            for (int i = 1; i <= m; i++)</pre>
36
            {
                int a = find(s[i].x), b = find(s[i].y);
37
                if (a == b)
38
                     continue;
39
                s[i].vis = 1;
40
41
                fa[a] = b;
42
                ans += s[i].z;
                add_edge(s[i].x, s[i].y, s[i].z);
43
                add_edge(s[i].y, s[i].x, s[i].z);
44
            }
45
46
        void bfs(int root) {
47
            deep[root] = 0;
48
49
            queue<int> q;
50
            q.push(root);
            while (q.size()) {
51
                int x = q.front(), len = (int)log2(deep[x] + 1);
52
53
                q.pop();
                for (int i = head[x]; i; i = Next[i]) {
54
                     int y = edge[i];
55
                     if (y == father[x][0])
56
                         continue;
57
                     deep[y] = deep[x] + 1;
58
                     father[y][0] = x, dp[0][y][0] = ver[i], dp[1][y][0] = -INF;
59
                     q.push(y);
60
                     for (int t = 1; t <= len; t++) {
61
62
                         father[y][t] = father[father[y][t - 1]][t - 1];
63
                         if (dp[0][y][t - 1] != dp[0][father[y][t - 1]][t - 1]) {
64
                             dp[0][y][t] = max(dp[0][y][t - 1], dp[0][father[y][t - 1]][t -
       1]);
                             dp[1][y][t] = min(dp[0][y][t - 1], dp[0][father[y][t - 1]][t -
65
       1]);
                         }
66
67
                         else {
                             dp[0][y][t] = dp[0][y][t - 1];
68
```

```
dp[1][y][t] = max(dp[1][y][t - 1], dp[1][father[y][t - 1]][t -
69
        1]);
                          }
70
                      }
71
72
73
                 }
74
             }
75
         inline void update2(int x) {
76
77
             if (x > val1)
78
                 val2 = val1, val1 = x;
79
             else if (x > val2 \&\& x != val1)
                 val2 = x;
80
81
         inline void update(int x, int t) {
82
             update2(dp[0][x][t]);
83
             update2(dp[1][x][t]);
84
85
         inline void Lca(int x, int y) {
86
             val1 = val2 = -INF;
87
             if (deep[x] < deep[y])</pre>
88
89
                 swap(x, y);
             while (deep[x] > deep[y]) {
90
91
                 int t = (int)log2(deep[x] - deep[y]);
92
                 update(x, t), x = father[x][t];
93
             if(x == y)
94
95
                 return;
             for (int t = (int)log2(deep[x]); t >= 0; t--) {
96
                 if (father[x][t] != father[y][t]) {
97
98
                      update(x, t), update(y, t);
                      x = father[x][t];
99
100
                     y = father[y][t];
                 }
101
102
             }
             update(x, 0), update(y, 0);
103
104
105
    } g1;
106
107
    void solve() {
         scanf("%d%d", &n, &m);
108
         g1.init2();
109
         for (int i = 1; i <= m; i++) {
110
             int a, b, c;
111
             scanf("%d%d%d", &a, &b, &c);
112
             s[i].x = a, s[i].y = b, s[i].z = c;
113
             fa[i] = i;
114
115
         g1.Kruskal();
116
117
         g1.bfs(1);
         ans_max = INF;
118
119
         for (int i = 1; i <= m; i++) {
             if (!s[i].vis) {
120
                 g1.Lca(s[i].x, s[i].y);
121
122
                 if (val1 != s[i].z)
                      ans_max = min(ans_max, ans - val1 + s[i].z);
123
124
                 else
125
                      ans_max = min(ans_max, ans - val2 + s[i].z);
126
             }
```

```
127
        printf("%lld\n", ans_max);
128
    }
129
130
131
    int main() {
        ios_base::sync_with_stdio(false);
132
        cin.tie(nullptr);
133
        cout.tie(nullptr);
134
    #ifdef ACM_LOCAL
135
        freopen("in.txt", "r", stdin);
//freopen("out.txt", "w", stdout);
136
137
        signed test_index_for_debug = 1;
138
        char acm_local_for_debug = 0;
139
        do {
140
             if (acm_local_for_debug == '$') exit(0);
141
            if (test_index_for_debug > 20)
142
                throw runtime_error("Check the stdin!!!");
143
            auto start_clock_for_debug = clock();
144
145
            solve();
            auto end_clock_for_debug = clock();
146
            cout << "Test " << test_index_for_debug << " successful" << endl;</pre>
147
            cerr << "Test " << test_index_for_debug++ << " Run Time: "</pre>
148
                 << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC <<</pre>
149
        "s" << endl;
            cout << "----" << endl;
150
        } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
151
152
    #else
153
        solve();
    #endif
154
155
        return 0;
    }
156
    2.4 点分治
 1 #include <iostream>
 2 #include <algorithm>
 3 #include <cstdio>
 4 #include <cstring>
 5 #include <queue>
 6 #include <stack>
 7 #include <cmath>
 8 #include <bitset>
 9 #include <map>
 10
11 using namespace std;
    typedef long long 11;
12
13 const int N = 1e4 + 5, M = 2e5 + 5, INF = 0x3f3f3f3f;
14
15 struct Edge {
        int to, next, vi;
16
   } e[N << 1];
17
18
19 int h[N], cnt;
20
    void add(int u, int v, int w) {
21
        e[cnt].to = v;
22
        e[cnt].vi = w;
23
        e[cnt].next = h[u];
24
```

```
h[u] = cnt++;
25
  }
26
27
28 ll ans = 0;
   int root, sum, n;
30 int sz[N], mx[N], vis[N];
31
   void getroot(int x, int fa) {
32
33
        sz[x] = 1, mx[x] = 0;
        for (int i = h[x]; \sim i; i = e[i].next) {
34
35
            int y = e[i].to;
36
            if (vis[y] || y == fa) continue;
            getroot(y, x);
37
            sz[x] += sz[y];
38
            mx[x] = max(mx[x], sz[y]);
39
40
        mx[x] = max(mx[x], sum - sz[x]);
41
42
        if (mx[x] < mx[root]) root = x;
   }
43
44
45 int d[N], dep[N];
46
   void getd(int x, int fa) {
47
48
        d[++d[0]] = dep[x];
49
        for (int i = h[x]; \sim i; i = e[i].next) {
            int y = e[i].to;
50
            if (vis[y] | | y == fa) continue;
51
            dep[y] = dep[x] + e[i].vi;
52
            getd(y, x);
53
        }
54
   }
55
56
   int K;
57
58
   int cal(int x, int now) {
59
        dep[x] = now, d[0] = 0;
60
61
        getd(x, -1);
        sort(d + 1, d + d[0] + 1);
62
        int res = 0, l = 1, r = d[0];
63
        while (l < r) {
64
            if (d[1] + d[r] > K) r--;
65
            else res += r - 1, 1++;
66
67
68
        return res;
   }
69
70
   void work(int x) {
71
        ans += cal(x, 0);
72
        vis[x] = 1;
73
74
        for (int i = h[x]; \sim i; i = e[i].next) {
75
            int y = e[i].to;
76
            if (vis[y]) continue;
            ans -= cal(y, e[i].vi);
77
78
            sum = sz[y], root = 0;
            getroot(y, -1);
79
80
            work(root);
81
        }
82
   }
83
```

```
int main() {
84
        while (scanf("%d %d", &n, &K), n) {
85
            cnt = 0;
86
            memset(h, -1, sizeof h);
87
            memset(vis, 0, sizeof vis);
88
            for (int i = 1; i <= n - 1; i++) {
89
                int u, v, w;
90
                scanf("%d %d %d", &u, &v, &w);
91
92
                add(u, v, w), add(v, u, w);
93
            }
            root = 0, ans = 0, sum = n, mx[0] = INF;
94
95
            getroot(1, -1);
96
            work(root);
            printf("%lld\n", ans);
97
        }
98
   }
99
   2.5
         树的半径
1 #include <bits/stdc++.h>
2 using namespace std;
 3 const int N = 5000 + 5;
  int dis[N], n, m, h[N], cnt, maxd1[N], maxd2[N], mv[N], st, r;
   struct edge {
6
7
        int u, to, next, vi;
8
   }e[N<<1];
9
10
   void add(int u, int v, int w) {
        e[cnt].u = u;
11
12
        e[cnt].to = v;
13
        e[cnt].vi = w;
        e[cnt].next = h[u];
14
15
        h[u] = cnt++;
16
   }
17
   void dp(int u, int fa) {
19
        maxd1[u] = 0, maxd2[u] = 0;
20
        for (int i = h[u]; \sim i; i = e[i].next) {
21
            int v = e[i].to;
22
            if (v == fa) continue;
23
            dp(v, u);
            int tmp = \max d1[v] + e[i].vi;
24
            if (tmp > maxd1[u]) maxd2[u] = maxd1[u], maxd1[u] = tmp, mv[u] = v;
25
26
            else if (tmp > maxd2[u]) maxd2[u] = tmp;
27
        st = max(st, maxd1[u] + maxd2[u]);
28
   }
29
30
   void dfs(int u, int fa, int fr) {
31
        r = min(r, max(fr, maxd1[u]));
32
        for (int i = h[u]; \sim i; i = e[i].next) {
33
34
            int v = e[i].to;
35
            if (v == fa) continue;
            if (mv[u] == v) dfs(v, u, max(fr + e[i].vi, maxd2[u] + e[i].vi));
36
            else dfs(v, u, max(maxd1[u] + e[i].vi, fr + e[i].vi));
37
        }
38
39 }
```

```
40
   int main() {
41
42
       cin >> n;
       memset(h, -1, sizeof h);
43
       for (int i = 1; i \le n - 1; i++) {
44
           int x, y, z;
45
           cin >> x >> y >> z;
46
           add(x, y, z), add(y, x, z);
47
48
       dp(1, 0);
49
       r = 0x3f3f3f3f;
50
51
       dfs(1, 0, 0);
       cout << r << endl;</pre>
52
53 }
   2.6 树的直径
2 int f[N];
3 int ans;
   vector<int> g[N];
4
   void dp(int u, int fa) {
       for (auto v : g[u]) {
6
7
           if (v == fa) continue;
           dp(v, u);
8
           ans = max(ans, f[u] + f[v] + 1);
9
10
           f[u] = max(f[u], f[v] + 1);
11
   }
12
13
14
   //-----两次dfs
15
16
   int dfs(int u, int fa) {
17
18
       int maxpos = u;
       for (auto v : g[u]) {
   if (vis[v] | | v == fa) continue;
19
20
           dis[v] = dis[u] + 1;
21
           pre[v] = u;
22
           int z = dfs(v, u);
23
           if (dis[z] > dis[maxpos]) maxpos = z;
24
25
26
       return maxpos;
27 }
   2.7
         树哈希
1
2 vector<int> g[N];
3 int prime[N], siz[N], mx[N], cnt, rt, n;
4 bool is_prime[N];
  ull Hash[N];
5
6
   void get_prime(int n) {
       memset(is_prime, true, sizeof is_prime);
7
       for (int i = 2; i <= n; i++) {
8
           if (is_prime[i]) prime[++cnt] = i;
9
           for (int j = 1; j <= cnt && prime[j] * i <= n; j++) {</pre>
10
```

```
is_prime[prime[j] * i] = false;
11
                if (i % prime[j] == 0) break;
12
           }
13
       }
14
15
   }
   void getRt(int x, int fa) {
16
       siz[x] = 1, mx[x] = 0;
17
       for (auto v : g[x]) {
18
            if (v == fa) continue;
19
20
            getRt(v, x);
21
            siz[x] += siz[v];
22
           mx[x] = max(mx[x], siz[v]);
23
       }
       mx[x] = max(mx[x], n - siz[x]);
24
       if (mx[x] < mx[rt]) rt = x;
25
26
   ull getTreeHash(int x, int fa) {
27
28
       siz[x] = 1;
       ull res = 1;
29
30
       for (auto v : g[x]) {
            if (v == fa) continue;
31
            res += getTreeHash(v, x) * prime[siz[v]];
32
33
            siz[x] += siz[v];
34
35
       return res;
36 }
   2.8
         树链剖分
   const int N = 1e5 + 10;
   vector<int> q[N];
   int son[N], siz[N], dep[N], fat[N], dfn[N], rnk[N], top[N], tot;
   void dfs1(int u, int fa) {
       son[u] = -1; siz[u] = 1; dep[u] = dep[fa] + 1; fat[u] = fa;
5
6
        for (auto v : g[u]) {
7
            if (v == fa) continue;
8
            dfs1(v, u);
9
            siz[u] += siz[v];
10
            if (son[u] == -1 \mid | siz[v] > siz[son[u]]) son[u] = v;
11
       }
12
   }
13
   void dfs2(int u, int t) {
       rnk[dfn[u] = ++tot] = u; top[u] = t;
15
16
       if (son[u] == -1) return;
       dfs2(son[u], t);
17
       for (auto v : g[u]) {
18
            if (v != son[u] && v != fat[u]) dfs2(v, v);
19
       }
20
   }
21
   // -----结合线段树操作
   int querymax(int x, int y) {
23
       int ans = -1;
24
       while (top[x] != top[y]) {
25
            if (dep[top[x]] < dep[top[y]]) swap(x, y);</pre>
26
27
           ans = max(ans, st.query1(1, dfn[top[x]], dfn[x]));
28
           x = fa[top[x]];
       }
29
```

```
if (dfn[x] > dfn[y]) swap(x, y);
30
       ans = max(ans, st.query1(1, dfn[x], dfn[y]));//在点权转化成边权时, dfn[x]+1开始
31
       return ans;
32
33 }
34
   int querysum(int x, int y) {
35
36
       int ans = 0;
       while (top[x] != top[y]) {
37
            if (dep[top[x]] < dep[top[y]]) swap(x, y);</pre>
38
            ans += st.query2(1, dfn[top[x]], dfn[x]);
39
           x = fa[top[x]];
40
41
       }
       if (dfn[x] > dfn[y]) swap(x, y);
42
       ans += st.query2(1, dfn[x], dfn[y]);
43
       return ans;
44
   }
45
46
   void update(int x, int y, int c) {
47
       while (top[x] != top[y]) {
48
            if (dep[top[x]] < dep[top[y]]) swap(x, y);</pre>
49
            st.update(1, dfn[top[x]], dfn[x], c);
50
           x = fa[top[x]];
51
52
       if (dfn[x] > dfn[y]) swap(x, y);
53
54
       st.update(1, dfn[x], dfn[y], c);
55 }
   2.9 树上 k-th 祖先 O(q+nlogn)
   // Created by SANZONG on 2021/7/6.
3
4
   #include "bits/stdc++.h"
5
6
7
8 using namespace std;
9 const int maxn = 6e5;
10 int head[maxn << 1];</pre>
11 int cnt;
12 #define ui unsigned int
13 ui s;
14
   inline ui get(ui x) {
15
16
       x ^= x << 13;
       x ^= x >> 17;
17
18
       x ^= x << 5;
       return s = x;
19
20 }
21
22 #define int long long
23 struct node {
24
       int next;
25
       int to;
26 } a[maxn << 1];
27
28 void add(int u, int v) {
       a[++cnt].next = head[u];
```

```
a[cnt].to = v;
30
       head[u] = cnt;
31
   }
32
33
   int f[maxn][33];
34
   int dep[maxn];
35
   int son[maxn];
36
   int dlen[maxn];
                       //每个点所在的长链长度, 画一下就知道
37
   int g[maxn];
                      //log
38
39
   void dfs(int u) {
40
       dlen[u] = dep[u] = dep[f[u][0]] + 1;
41
       for (int i = 1; i \le 30; ++i) {
42
           f[u][i] = f[f[u][i - 1]][i - 1];
43
44
       for (int i = head[u]; i; i = a[i].next) {
45
           int v = a[i].to;
46
           if (v == f[u][0]) continue;
47
           dlen[v] = dep[v] = dep[u] + 1;
48
                                              //向下时先等于深度
           dfs(v);
49
           dlen[u] = max(dlen[u], dlen[v]); //逆推时一直取子树最大深度
50
           if (dlen[v] > dlen[son[u]]) son[u] = v;
51
       }
52
53
   }
54
   int U[maxn], D[maxn], id[maxn], cnt_t, top[maxn];
55
56
   void dfs_t(int u, int root) {
57
       id[u] = ++cnt_t;
                            //一条相同长链上的id是连续的。
58
       D[cnt_t] = u;
59
                               //dfs序为cnt_t的点的连顶向下数组。
       U[cnt_t] = root;
60
                              //dfs序为cnt_t的点的连顶向上数组。
       if (son[u]) {
61
           top[son[u]] = top[u];
62
           dfs_t(son[u], f[root][0]);
63
64
       for (int i = head[u]; i; i = a[i].next) {
65
           int v = a[i].to;
66
67
           if (v == f[u][0] \mid | v == son[u]) continue;
           top[v] = v;
68
                                        //短链
69
           dfs_t(v, v);
70
       }
   }
71
72
73
   int getkfa(int x, int k) {
74
       if (k == 0) return x;
75
       x = f[x][g[k]];
                           //直接跳到链上
                            //跳到f[x][g[k]]的消耗
76
       k -= 1 << g[k];
       k -= dep[x] - dep[top[x]]; //x 从x跳到顶点
77
       x = top[x];
78
79
       return k \ge 0? U[id[x]+k]: D[id[x]-k];
80 }
81
82 //int getkfa(int x,int k)
83 //{
84 //
         if (k == 0) return x;
85 //
         int d = dep[x] - k;
         for (int i = 30; i >= 0; --i) {
  //
86
87
   //
             if (dep[f[x][i]] > d) x = f[x][i];
88 //
```

```
89 //
           return f[x][0];
90 //}
    signed main() {
91
92
         ios::sync_with_stdio(0);
           freopen("in.txt","r",stdin);
93
94
         int n, q, root;;
95
         cin >> n >> q >> s;
         g[0] = -1;
96
         for (int i = 1; i \le n; ++i) { g[i] = g[i >> 1] + 1; }
97
         for (int i = 1; i <= n; ++i) {</pre>
98
             cin >> f[i][0];
99
100
             if (f[i][0] == 0) {
101
                 root = i;
102
             } else {
                 add(f[i][0], i);
103
104
                 add(i, f[i][0]);
             }
105
         }
106
         dep[root] = 1;
107
         dfs(root);
108
         top[root] = root;
109
         dfs_t(root, root);
110
    //
           for (int i = 1; i <= n; ++i) {
111
112 //
               cout << id[i] << endl;</pre>
113 //
114
         int ans = 0;
         int res = 0;
115
         for (int i = 1; i <= q; ++i) {
116
117
118
             int xi = ((get(s) \land ans) \% n) + 1;
   //
119
               cout << (get(s) ^ ans) << endl;</pre>
             int ki = (get(s) ^ ans) % dep[xi];
120
             ans = getkfa(xi, ki);
  cout << xi << ' ' << ki << ' ' << ans << endl;</pre>
121
   //
122
123
             res ^= (i * ans);
124
125
         cout << res << endl;</pre>
126 }
    2.10 树上差分 (边差分)
    边差分: c[x] += val, c[y] += val, c[lca(x, y)] -= 2*val;
    点差分: c[x] += val, c[y] += val, c[lca(x, y)] -= val, c[fa[lca(x, y)]] -= val;
           树上距离(lca tarjan)
    2.11
 1 #include <bits/stdc++.h>
 2 using namespace std;
 3 \quad const int N = 1e5 + 10;
 4 vector<int> query[N], query_id[N];
 5
 6
    struct edge {
 7
         int to, next, vi;
 8
    }e[N*2];
int T, n, q, h[N], cnt, ans[N], d[N], fa[N], vis[N];
```

```
11
   void add_query(int u, int v, int id) {
12
        query[u].push_back(v), query_id[u].push_back(id);
13
14
        query[v].push_back(u), query_id[v].push_back(id);
   }
15
16
   void add(int u, int v, int w) {
17
        e[cnt].to = v;
18
19
        e[cnt].vi = w;
        e[cnt].next = h[u];
20
21
        h[u] = cnt++;
22
   }
23
   int find(int x) {
24
        if (fa[x] == x) return x;
25
        else return fa[x] = find(fa[x]);
26
   }
27
28
   void tarjan(int x) {
29
30
        vis[x] = 1;
        for (int i = h[x]; \sim i; i = e[i].next) {
31
            int v = e[i].to;
32
            if (vis[v]) continue;
33
34
            d[v] = d[x] + e[i].vi;
35
            tarjan(v);
            fa[v] = x;
36
37
        }
38
        for (int i = 0; i < query[x].size(); i++) {
39
            int y = query[x][i], id = query_id[x][i];
40
41
            if (vis[y] == 2) {
                int lca = find(y);
42
                ans[id] = min(ans[id], d[x] + d[y] - 2 * d[lca]);
43
            }
44
45
        vis[x] = 2;
46
47
   }
48
   int main() {
49
        scanf("%d", &T);
50
        while (T--) {
51
            scanf("%d %d", &n, &q);
52
            for (int i = 1; i <= n; i++) {
53
54
                h[i] = -1;
                d[i] = 0;
55
                query[i].clear(), query_id[i].clear();
56
                fa[i] = i;
57
                vis[i] = 0;
58
            }
59
60
            cnt = 0;
            for (int i = 1; i \le n - 1; i++) {
61
                int x, y, z;
62
                scanf("%d %d %d", &x, &y, &z);
63
                add(x, y, z), add(y, x, z);
64
65
            for (int i = 1; i <= q; i++) {
66
                int x, y;
scanf("%d %d", &x, &y);
67
68
                if (x == y) ans[i] = 0;
69
```

```
else {
70
                     add_query(x, y, i);
71
72
                     ans[i] = 9999999999;
73
74
            }
            tarjan(1);
75
76
            for (int i = 1; i \le q; i++) {
77
                printf("%d\n", ans[i]);
78
79
        }
80 }
          树上距离(lca 倍增)
   2.12
   #include <bits/stdc++.h>
   using namespace std;
3
   const int N = 4e4 + 10;
   int h[N], dis[N], d[N], f[N][30];
5
6
   int T, n, m, cnt, s;
7
   struct edge {
8
9
        int to, next, vi;
  }e[N<<1];
10
11
12
  void add(int u, int v, int vi) {
        e[cnt].to = v;
13
14
        e[cnt].vi = vi;
        e[cnt].next = h[u];
15
        h[u] = cnt++;
16
17
   }
18
   void dfs(int x, int fa) {
19
        f[x][0] = fa, d[x] = d[fa] + 1;
20
        for (int i = 1; i \le 20; i++) f[x][i] = f[f[x][i-1]][i-1];
21
22
        for (int i = h[x]; \sim i; i = e[i].next) {
23
            int y = e[i].to;
            if (y != fa) dfs(y, x);
24
25
        }
26
   }
27
   int lca(int x, int y) {
28
        if (d[x] > d[y]) swap(x, y);
for (int i = 20; i >= 0; i--) if (d[f[y][i]] >= d[x]) y = f[y][i];
29
30
31
        if (x == y) return y;
32
33
        for (int i = 20; i >= 0; i--) {
            if (f[y][i] != f[x][i]) {
34
                y = f[y][i];
35
36
                x = f[x][i];
            }
37
38
39
        return f[x][0];
   }
40
41
42
   int main() {
        memset(h, -1, sizeof h);
43
        scanf("%d %d %d", &n, &m, &s);
44
```

```
for (int i = 1; i <= n - 1; i++) {
45
            int a, b, c;
46
            scanf("%d %d", &a, &b);
47
            add(a, b, 1), add(b, a, 1);
48
49
        dfs(s, 0);
50
        for (int i = 1; i <= m; i++) {
51
            int a, b;
52
            scanf("%d %d", &a, &b);
53
            printf("%d\n", lca(a, b));
54
55
        }
56
   }
          树上启发式合并
   2.13
1 #include <bits/stdc++.h>
   #define ACM_LOCAL
3 using namespace std;
4 typedef long long ll;
5 typedef pair<int, int> PII;
6 const int INF = 0x3f3f3f3f;
7 const int N = 2e5 + 10;
8 \text{ const int M} = 1e6 + 10;
9 const int MOD = 1e9 + 7;
int sz[N], son[N], h[N], tot, col[N], cnt[N], vis[N], mx;
11 ll sum, ans[N];
12 struct Edge {
        int to, next;
13
14
  }e[M];
15
16
   void add(int u, int v) {
        e[tot].to = v;
17
18
        e[tot].next = h[u];
19
        h[u] = tot++;
20 }
21
   void dfs(int u, int fa) {
22
23
        sz[u] = 1;
24
        for (int i = h[u]; ~i; i = e[i].next) {
25
            int v = e[i].to;
26
            if (v == fa) continue;
            dfs(v, u);
27
            sz[u] += sz[v];
if (!son[u] || sz[v] > sz[son[u]])
28
29
30
                son[u] = v;
31
        }
   }
32
33
   void count(int u, int fa, int k) {
34
        //统计子树信息
35
36
37
38
        for (int i = h[u]; \sim i; i = e[i].next) {
39
            int v = e[i].to;
40
            if (v == fa || vis[v]) continue;
41
            count(v, u, k);
42
        }
43
```

```
44 }
45
   void dsu(int u, int fa, int keep) {
46
       for (int i = h[u]; \sim i; i = e[i].next) {
47
            int v = e[i].to;
48
49
            if (v == fa || son[u] == v) continue;
            dsu(v, u, 0);//查询轻儿子
50
51
       if (son[u]) dsu(son[u], u, 1), vis[son[u]] = 1;//查询重儿子
52
       count(u, fa, 1);//统计子树信息
53
54
       //统计答案
55
       if (son[u]) vis[son[u]] = 0;
       if (!keep) count(u, fa, -1), mx = sum = 0;//撤回信息
56
57
   2.14 k 级祖先
   vector<int> g[N];
1
   int dep[N], f[N][30];
   void dfs(int x, int fa) {
3
       f[x][0] = fa; dep[x] = dep[fa] + 1;
4
       for (int i = 1; i \le 20; i++) {
5
6
            f[x][i] = f[f[x][i-1]][i-1];
7
       for (auto &v : g[x]) {
8
            if (v == fa) continue;
9
            dfs(v, x);
10
       }
11
   }
12
   int find_Kth(int x, int k) {
13
       int bit = 0;
14
       while (k) {
15
            if (k \& 1) x = f[x][bit];
16
            k >>= 1;
17
18
           bit++;
19
       }
20
       return x;
21 }
```

3 数据结构

3.1 Max XOR(tire)

```
#include <bits/stdc++.h>
2 using namespace std;
3 \quad const int N = 100010;
4 \quad const int M = 3000010;
5 int n, a[N];
6 int trie[M][2];
7
   int cnt;
8
   void insert(int x){
9
        int p = 0;
        for (int i = 30; \sim i; i--){
10
            int w = x >> i & 1;
11
12
            if (!trie[p][w]) trie[p][w] = ++cnt;
            p = trie[p][w];
13
        }
14
   }
15
16
   int query(int x){
17
18
        int p = 0;
        int res = 0;
19
20
        for (int i = 30; \sim i; i--){
21
            int w = x >> i \& 1;
22
23
            if (trie[p][w ^ 1]){
                res += (1 << i);
24
25
                p = trie[p][w \wedge 1];
            }
26
27
            else p = trie[p][w];
28
29
        return res;
   }
30
   3.2 splay(待完善)
   #include <bits/stdc++.h>
2
3
   #define ACM_LOCAL
4 #define fi first
5 #define se second
  #define pb push_back
   using namespace std;
   typedef long long ll;
9 typedef pair<int, int> PII;
10 const int N = 5e5 + 10, M = 5e5 + 10, INF = 0x3f3f3f3f;
11 const int MOD = 1e9 + 7;
12 int n, m, k, cnt;
13 int a[N];
14
   struct SplayTree {
15
         int sz[N], ch[N][2], cnt[N], fa[N], rev[N], val[N], ad[N];
16
17
         int tot, rt;
   #define rt ch[anc][0]
18
   #define keynode ch[ch[rt][1]][0]
19
20
         const int anc = 0;
         void clear(int x) \{sz[x] = cnt[x] = ch[x][0] = ch[x][1] = val[x] = ad[N] = 0;\}
21
```

```
void init() {tot = 0, clear(0);}
22
23
        int get(int x) {return ch[fa[x]][1] == x;}
24
        int newnode(int k) {
25
26
            int x = ++tot; clear(x);
27
            val[x] = k;
            sz[x] = cnt[x] = 1;
28
29
            return tot;
        }
30
31
32
        void setc(int x, int y, int d) { //基本点操作: 设置x的d号儿子为y
33
            ch[x][d] = y;
            if (y) fa[y] = x;
34
35
            if (x) push_up(x);
        }
36
37
       void reverse(int x) {
38
           if (x == 0) return;//Necessary
39
           swap(ch[x][0], ch[x][1]);
40
           rev[x] ^= 1;
41
42
       void add(int x, int k) {
43
           if (x == 0)return;
44
45
           val[x] += k;
46
           ad[x] += k;
       }
47
48
       void push_up(int x) {
49
           sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + cnt[x];
50
51
52
       void push_down(int x) {
53
           if (!x) return;
54
           if (rev[x]) {
55
               reverse(ch[x][0]);
56
               reverse(ch[x][1]);
57
58
               rev[x] = 0;
59
           if (ad[x]) {
60
               add(ch[x][0], ad[x]);
61
62
               add(ch[x][1], ad[x]);
               ad[x] = 0;
63
           }
64
65
       void rotate(int x) {
66
           int f = fa[x];
67
           int ff = fa[f];
68
           bool d = get(x);
69
           bool dd = get(f);
70
71
           setc(f, ch[x][!d], d); //第一步: 把与x原父节点同方向的子树取作原父节点f的子树
72
           setc(x, f, !d);
                                    //第二步: 使与x原父节点同方向的子树连接原父节点f
73
           setc(ff, x, dd);
                                    //第三步: 使得x替代原父节点f, 变为新父节点点ff的子树
74
75
76
       void splay(int x, int anc = 0) {//双旋
77
           if (x == 0)return;
           while (fa[x] != anc) {
78
79
               push_down(fa[x]);
80
               push_down(x);
```

```
if (fa[fa[x]] != anc)rotate(get(x) == get(fa[x]) ? fa[x] : x);
81
82
                rotate(x);
            }
83
84
        int rank(int k) {//排名
85
            int x = rt;
86
            while (1) {
87
                if (x == 0) return 0;
88
                if (k == val[x]) { splay(x); return x; }
89
                bool d = k > val[x];
90
                x = ch[x][d];
91
92
            }
            splay(x);
93
        }
94
95
        void ins(int k) {
96
             int x = rank(k);
97
98
             if (x) {cnt[x]++; sz[x]++; return;}
            int fa = anc; x = rt;
99
            bool d = 0;
100
            while (x) {
101
                fa = x;
102
                d = k > val[x];
103
104
                x = ch[x][d];
105
            }
            x = newnode(k);
106
            setc(fa, x, d);
107
            splay(x);
108
        }
109
110
        int Min(int x) {//子树中最小节点
111
            push_down(x);
112
113
            while (ch[x][0]) x = ch[x][0], push_down(x);
            return x;
114
115
        int Max(int x) {//子树中最大节点
116
117
            push_down(x);
118
            while (ch[x][1]) x = ch[x][1], push_down(x);
119
            return x;
120
121
        int kth(int k) {
                    //这里涉及到区间操作,我们在左右界各添加新节点,因此进入时要++k
122
            ++k;
123
            int x = rt;
124
            while (1) {
                push_down(x);
125
126
                if (sz[ch[x][0]] >= k) x = ch[x][0];
127
                else if (sz[ch[x][0]] + cnt[x] >= k) return x;
                else
128
                {
129
130
                     k = sz[ch[x][0]] + cnt[x];
131
                     x = ch[x][1];
132
                }
133
            }
134
135
        void del(int x) {
             splay(x);//转到根后便不再需要pushdown()
136
137
             if (ch[x][0] == 0)setc(anc, ch[x][1], 0);
                                                        //如果没有左子树,则直接把右子树放到树根
138
            else
            {
139
```

```
setc(anc, ch[x][0], 0);
                                                           //第一步: 把左子树放到树根
140
                 splay(Max(ch[x][0]), anc);
                                                           //第二步: 把左子树最大节点转到树根
141
                                                           //第三步: 把右子树接到树根上
                 setc(rt, ch[x][1], 1);
142
             }
143
        }
144
        //查找操作:返回树中[l,r]区间段的根节点
145
        int segment(int l, int r) {
    splay(kth(l - 1), anc);
146
147
             splay(kth(r + 1), rt);
148
             return keynode;
149
150
        }
151
        //分离操作:把[1,r]区间段从树中分离
        int split(int l, int r) {
152
             int x = segment(1, r); fa[x] = 0;
153
             setc(ch[rt][1], 0, 0);
154
             push_up(rt);
155
156
             return x;
        }
157
        //合并操作,把子树x插入到pos位置的右侧
158
        void inspos(int x, int pos) {
159
             segment(pos + 1, pos); //使得pos位于根, pos+1位于根的右子树
160
             setc(ch[rt][1], x, 0);
161
             push_up(rt);
162
163
        //中序遍历
164
        void ldr(int x) {
165
             if (ch[x][0]) ldr(ch[x][0]);
166
            printf("%d ", x);
if (ch[x][1]) ldr(ch[x][1]);
167
168
169
         void add(int l, int r, int k) {//区间加
170
             int x = segment(1, r);
171
172
             add(x, k);
             splay(x);
173
        }
174
175
176
        void reverse(int l, int r) {//区间翻转
177
             int x = segment(1, r);
             reverse(x);
178
179
             splay(x);
180
        }
181
182
183
184 }spy;
185
186
    void solve() {
        int n, opt, x;
187
188
        scanf("%d", &n);
189
        for (int i = 1; i <= n; i++) scanf("%d", &a[i]);</pre>
190
191 }
192
193
    int main() {
        ios_base::sync_with_stdio(false);
194
195
         cin.tie(0);
196
         cout.tie(0):
197
    #ifdef ACM_LOCAL
        freopen("input", "r", stdin);
198
```

```
freopen("output", "w", stdout);
199
    #endif
200
         solve();
201
         return 0;
202
203 }
    3.3 ST 表
    namespace ST {
 1
         int mi[N][21], ma[N][21], lg[N], a[N], gcd[N][21];
 2
 3
         int cmp1(int x, int y) {
             return a[x] < a[y] ? x : y;
 4
 5
 6
         int cmp2(int x, int y) {
             return a[x] > a[y] ? x : y;
 7
 8
 9
         void init(int n) {
             for (int i = 1; i <= n; i++) {</pre>
10
                 cin >> a[i];
11
                 gcd[i][0] = a[i];
12
                 ma[i][0] = mi[i][0] = i, lg[i] = log2(i);
13
14
             for (int i = 1; i <= 20; i++) {
15
                 for (int j = 1; j + (1 << i) - 1 <= n; j++) {
16
                     mi[j][i] = cmp1(mi[j][i - 1], mi[j + (1 << (i - 1))][i - 1]);
17
                     ma[j][i] = cmp2(ma[j][i - 1], ma[j + (1 << (i - 1))][i - 1]);
18
                      gcd[j][i] = \__gcd(gcd[j][i - 1], gcd[j + (1 << (i - 1))][i - 1]);
19
20
                 }
             }
21
22
         int qry_mi(int 1, int r) {
23
24
             int k = \lg[r - l + 1];
             return a[cmp1(mi[l][k], mi[r - (1 << k) + 1][k])];</pre>
25
26
         int qry_ma(int 1, int r) {
27
             int k = \lg[r - l + 1];
28
             return a[cmp2(ma[l][k], ma[r - (1 << k) + 1][k])];
29
30
         int qry_pmi(int l, int r) {
31
             int k = \lg[r - l + 1];
32
             return cmp1(mi[l][k], mi[r - (1 << k) + 1][k]);</pre>
33
34
         int gry_pma(int 1, int r) {
35
             int k = \lfloor q \lceil r - l + 1 \rceil;
36
37
             return cmp2(ma[l][k], ma[r - (1 << k) + 1][k]);
38
         int qry_gcd(int 1, int r) {
39
             int k = \lg[r - l + 1];
40
             return \_gcd(gcd[l][k], gcd[r - (1 << k) + 1][k]);
41
42
43
    }
    3.4
 1 #include <bits/stdc++.h>
 2 #define ACM_LOCAL
 3 using namespace std;
```

```
4 typedef long long ll;
5 typedef pair<int, int> PII;
  const int N = 1e5 + 100, M = 5e5 + 5, INF = 0x3f3f3f3f;
   mt19937 rnd(233);
   struct Treap {
        int lc[N], rc[N], val[N], key[N], sz[N];
9
10
        int cnt, root;
11
        inline int newnode(int v) {
12
            val[++cnt] = v;
13
14
            key[cnt] = rnd();
15
            sz[cnt] = 1;
            return cnt;
16
        }
17
18
        inline void update(int now) {
19
            sz[now] = sz[lc[now]] + sz[rc[now]] + 1;
20
        }
21
22
        void split(int now, int v, int &x, int &y) {
23
24
            if (!now) x = y = 0;
25
            else {
26
                if (val[now] <= v) {</pre>
27
                    x = now;
28
                     split(rc[now], v, rc[now], y);
29
                } else {
30
                    y = now;
                     split(lc[now], v, x, lc[now]);
31
32
                update(now);
33
34
            }
35
        }
36
37
        int merge(int x, int y) {
            if (!x | !y) return x + y;
38
            if (key[x] > key[y]) {
39
40
                rc[x] = merge(rc[x], y);
41
                update(x);
                return x;
42
43
            } else {
                lc[y] = merge(x, lc[y]);
44
                update(y);
45
46
                return y;
47
            }
        }
48
49
50
        int x, y, z;
51
        inline void insert(int val) {
52
53
            split(root, val, x, y);
54
            root = merge(merge(x, newnode(val)), y);
55
        }
56
        inline void del(int val) {
57
            split(root, val, x, z);
58
59
            split(x, val - 1, x, y);
60
            y = merge(lc[y], rc[y]);
61
            root = merge(merge(x, y), z);
        }
62
```

```
63
        inline int getrank(int v) { //查询v的排名
64
             split(root, v - 1, x, y);
65
             int rank = sz[x] + 1;
66
67
             root = merge(x, y);
             return rank;
68
69
        }
70
        inline int getnum(int rank) {//查询排名第rank的数
71
             int now = root;
72
73
             while (now) {
74
                 if (sz[lc[now]] + 1 == rank)
                     break;
75
                 else if (sz[lc[now]] >= rank)
76
                     now = lc[now];
77
78
                 else {
                     rank -= sz[lc[now]] + 1;
79
80
                     now = rc[now];
81
                 }
82
             }
             return val[now];
83
        }
84
85
86
        inline int pre(int v) {
             split(root, v - 1, x, y);
87
88
             int now = x;
            while (rc[now])
89
                 now = rc[now];
90
             int pre = val[now];
91
92
             root = merge(x, y);
93
             return pre;
94
        }
95
        inline int nxt(int v) {
96
             split(root, v, x, y);
97
             int now = y;
98
99
            while (lc[now])
100
                 now = lc[now];
             int nxt = val[now];
101
102
             root = merge(x, y);
103
             return nxt;
104
105
    }fhq;
          Treap(文艺平衡树)
 1 #include <bits/stdc++.h>
 2 #define ACM_LOCAL
 3 using namespace std;
 4 typedef long long ll;
 5 typedef pair<int, int> PII;
 6 const int N = 1e5 + 10, M = 1e5 + 10, INF = 0x3f3f3f3f;
 7
   const int MOD = 1e9 + 7;
   mt19937 rnd(233);
 8
    struct Treap {
 9
        int key[N], lc[N], rc[N], val[N], sz[N];
10
        bool reverse[N];
11
12
        11 sum[N], tag[N];
```

```
int cnt, rt;
13
14
        inline int newnode(int v) {
15
16
            val[++cnt] = v;
17
            sz[cnt] = 1;
            key[cnt] = rnd();
18
            reverse[cnt] = false;
19
            return cnt;
20
21
        inline void push_up(int now) {
22
23
            sz[now] = sz[lc[now]] + sz[rc[now]] + 1;
24
            sum[now] = sum[lc[now]] + sum[rc[now]] + val[now];
25
        inline void push_down(int now) {
26
            if (reverse[now]) {
27
                swap(lc[now], rc[now]);
28
                reverse[lc[now]] ^= 1;
29
                reverse[rc[now]] ^= 1;
30
                reverse[now] = false;
31
32
            if (tag[now]) {
33
                sum[lc[now]] += sz[lc[now]] * tag[now];
34
                val[lc[now]] += tag[now];
35
36
                tag[lc[now]] += tag[now];
37
                sum[rc[now]] += sz[rc[now]] * tag[now];
38
                val[rc[now]] += tag[now];
39
                tag[rc[now]] += tag[now];
40
41
            tag[now] = 0;
42
43
        void split(int now, int siz, int &x, int &y) {
44
            if (!now) x = y = 0;
45
            else {
46
                push_down(now);
47
                if (sz[lc[now]] < siz) {</pre>
48
49
                     x = now;
50
                     split(rc[now], siz-sz[lc[now]]-1, rc[now], y);
51
                } else {
52
                     split(lc[now], siz, x, lc[now]);
53
54
                push_up(now);
55
56
            }
57
58
        int merge(int x, int y) {
            if (!x | ! ! y) return x + y;
59
            if (key[x] < key[y]) {</pre>
60
                push_down(x);
61
                rc[x] = merge(rc[x], y);
62
63
                push_up(x);
64
                return x;
65
            } else {
                push_down(y);
66
                lc[y] = merge(x, lc[y]);
67
68
                push_up(y);
69
                return y;
70
            }
        }
71
```

```
void rev(int l, int r) {
72
             int x, y, z;
split(rt, l-1, x, y);
73
74
75
             split(y, r-l+1, y, z);
             reverse[y] ^= 1;
76
             rt = merge(merge(x, y), z);
77
78
        void ldr(int now) {
79
             if (!now) return;
80
81
             push_down(now);
             ldr(lc[now]);
82
             printf("%d", val[now]);
83
             ldr(rc[now]);
84
85
        void add(int l, int r, int v) {
86
             int x, y, z;
split(rt, l-1, x, y);
87
88
89
             split(y, r-l+1, y, z);
90
             tag[y] += v;
             sum[y] += sz[y] * v;
91
             val[y] += v;
92
             rt = merge(merge(x, y), z);
93
94
95
        ll query(int l, int r) {
96
             ll\ ans = 0;
97
             int x, y, z;
             split(rt, l-1, x, y);
98
             split(y, r-l+1, y, z);
99
             ans = sum[y];
100
             rt = merge(merge(x, y), z);
101
102
             return ans;
103
        void insert(int pos, int v) {
104
             int x, y, z;
105
             split(rt, pos-1, x, y);
106
107
             z = newnode(v);
108
             rt = merge(merge(x, z), y);
109
110 }fhq;
    3.6 并查集
   //带权并查集
    const int N = 1e5 + 10;
 3
    int fa[N], d[N];
    int find(int x) {
 4
        int par = fa[x];
 5
        fa[x] = find(fa[x]);
 6
 7
        d[x] += d[par];
    }
 8
 9
10
    //合并x和y,并计算路径长度
    void merge(int x, int y, int w) {//w代表 x->y 路径长
11
12
        int fx = find(x), fy = find(y);
        if (fx != fy) fa[fx] = fy, d[fx] = d[y] - d[x] + w;
13
        else {
14
             int\ dis = d[x] - d[y]; // 如果x和y已经存在关系, dis代表x到y的距离
15
16
```

```
17 }
18
  //种类并查集
19
20
21 find(x) --- x的同类
22 find(x+n) ---
23 find(x+n+n...) ---
24
25
  //建立虚点
   void merge(int x, int y) {//正常的合并操作
26
27
       int fx = find(x), fy = find(y);
28
       ans[fy] += ans[fx];
       num[fy] += num[fx];
29
       fa[fx] = fy;
30
   }
31
32
   void move(int x) {//id[x] 代表x目前的编号
33
       int fx = find(id[x]);
34
       ans[fx] -= x; //原来集合中减去x的贡献
35
       num[fx]--;
36
       id[x] = ++n;//建立新的节点
37
       ans[id[x]] = x;
38
       num[id[x]] = 1;
39
40
       fa[id[x]] = id[x];
41 }
42
   //可撤销并查集
43
   struct Undo_dsu {
44
       stack<PII> st;
45
       int fa[N], siz[N];
46
       void init() {
47
           while (st.size()) st.pop();
48
49
           for (int i = 1; i <= n; i++) fa[i] = i, siz[i] = 1;
50
       int find(int x) {return fa[x] == x ? x : find(fa[x]);}
51
       bool merge(int x, int y) {
52
53
           int fx = find(x), fy = find(y);
54
           if (fx == fy) return false;
           if (siz[fx] > siz[fy]) swap(fx, fy), swap(x, y);
55
           siz[fy] += siz[fx], fa[fx] = fy;
56
           st.push({fx, fy});
57
           return true;
58
59
       void undo() {
60
           PII now = st.top();
61
62
           fa[now.fi] = now.fi;
           siz[now.se] -= siz[now.fi];
63
           st.pop();
64
65
   }dsu;
         单调队列
   3.7
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1000010;
4 int q[N], a[N], n, k;
5 void deque_max() {
```

```
int hh = 1, tt = 0;
6
        for (int i = 1; i <= n; i ++) {
7
            while(hh <= tt && q[hh] + k <= i) hh ++;</pre>
8
9
            while(hh <= tt && a[q[tt]] < a[i]) tt --;</pre>
            q[++tt] = i;
10
            if (i >= k) cout << a[q[hh]] << " ";
11
12
        }
   }
13
14
   void deque_min() {
15
        int hh = 1, tt = 0;
16
17
        for (int i = 1; i <= n; i ++) {
            while(hh <= tt && q[hh] + k <= i) hh ++;</pre>
18
            while(hh <= tt && a[q[tt]] > a[i]) tt --;
19
20
            q[++tt] = i;
            if (i >= k) cout << a[q[hh]] << " ";
21
        }
22
23 }
         单调栈
   3.8
 1 #include <bits/stdc++.h>
   using namespace std;
3 const int N = 3e6 + 10;
4 int a[N], n, idx[N];
5 stack<int> stk;
   int main() {
6
        cin >> n;
7
        for (int i = 1; i <= n; i++) cin >> a[i];
8
9
        for (int i = 1; i <= n; i++) {
            while (stk.size() && a[stk.top()] < a[i]) {</pre>
10
                 idx[stk.top()] = i;
11
12
                 stk.pop();
            }
13
14
            stk.push(i);
15
        for (int i = 1; i <= n; i++) cout << idx[i] << " ";
16
17
         笛卡尔树
   3.9
   #include <bits/stdc++.h>
2 #define ACM_LOCAL
3 using namespace std;
4 typedef long long ll;
5 typedef pair<int, int> PII;
6 const int N = 3e5 + 10, M = 1e5 + 10, INF = 0x3f3f3f3f;
7 \quad const \quad int \quad MOD = 1e9 + 7;
8 int n, m, k;
   struct Car_tree {
        int lc[N], rc[N], stk[N], top, val[N];
10
        void init(int n) {
11
            for (int i = 0; i \le n; i++) lc[i] = rc[i] = 0;
12
13
        int build(int n) {
14
15
            int rt;
            for (int i = 1; i <= n; i++) {
16
```

```
scanf("%d", &val[i]);
17
                while (top && val[stk[top]] > val[i]) {
18
19
                    lc[i] = stk[top], top--;
20
21
                if (top) rc[stk[top]] = i;
22
                stk[++top] = i;
23
            }
24
            while (top) rt = stk[top--];
25
            return rt;
26
       }
27
  }tr;
          静态主席树
   3.10
   #include <bits/stdc++.h>
   using namespace std;
   typedef long long 11;
3
4 const int N = 2e5 + 5;
5
6
  int a[N], b[N], rt[N];
   int n, m;
7
   struct Hash {
8
9
       int b[N], tot;
10
       void init() {tot = 0;}
       void insert(int x) {b[++tot] = x;}
11
12
       void build() {
13
            sort(b+1, b+1+tot);
14
            tot = unique(b+1, b+tot+1) - (b+1);
15
       int pos(int x) {return lower_bound(b+1, b+tot+1, x) - b;}
16
17
   }ha;
   struct {
18
19
       int t[N << 5], lc[N << 5], rc[N << 5];</pre>
20
       int NodeNum = 0;
       ll sum[N << 5];
21
       int build(int 1, int r) {
22
23
            int num = ++NodeNum;
24
            if (l != r) {
25
                int mid = (l + r) >> 1;
26
                lc[num] = build(l, mid);
27
                rc[num] = build(mid + 1, r);
28
29
            return num;
30
31
        int update(int pre, int 1, int r, int x) {
            int num = ++NodeNum;
32
            lc[num] = lc[pre], rc[num] = rc[pre], t[num] = t[pre] + 1;
33
            if (l != r) {
34
                int mid = (l + r) \gg 1;
35
                if (x \le mid) lc[num] = update(lc[pre], l, mid, x);
36
37
                else rc[num] = update(rc[pre], mid + 1, r, x);
            }
38
39
            return num;
40
       int Qry_Kth_Num(int u, int v, int l, int r, int k) {
41
42
            if (l == r) return ha.b[l];
            int mid = (l + r) >> 1, num = t[lc[v]] - t[lc[u]];
43
            if (num >= k) return Qry_Kth_Num(lc[u], lc[v], l, mid, k);
44
```

```
else return Qry_Kth_Num(rc[u], rc[v], mid + 1, r, k-num);
45
46
       ĺl Qry_Kth_Sum(int u, int v, int l, int r, int k) {//k大数之和
47
           if (l == r) return 1ll*ha.b[l] * k;
48
           int mid = (l + r) >> 1, num = t[rc[v]] - t[rc[u]];
49
            if (num >= k) return Qry_Kth_Sum(rc[u], rc[v], mid+1, r, k);
50
           else return sum[rc[v]] - sum[rc[u]] + Qry_Kth_Sum(lc[u], lc[v], l, mid, k-num);
51
52
       int Binary_Search(int left, int right, int val) {//查找小于等于val的个数
53
            int l = 1, r = right - left + 1;
54
           while (l \ll r) {
55
56
               int mid = l + r \gg 1;
               int num = Qry_Kth_Num(rt[left - 1], rt[right], 1, ha.tot, mid);
57
               if (num > val) r = mid - 1;
58
               else l = mid + 1;
59
           }
60
61
           return r;
62
63
  }hjt;
   3.11
          珂朵莉树
1 #include <iostream>
2 #include <cstring>
3 #include <algorithm>
4 #include <stack>
5 #include <queue>
6 #include <map>
7 #include <vector>
8 #include <cstdio>
9 #include <cmath>
10 #include <ctime>
11 #include <bitset>
12 #include <unordered_map>
13 #include <string>
14 #include <set>
15 using namespace std;
16 #define ACM_LOCAL
17 #define IT set<node> ::iterator
18 inline int read(){
19
      int s=0, w=1;
20
      char ch=getchar();
      while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
21
      while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();</pre>
22
23
      return s*w;
24 }
25 const int N = 3e5 + 10;
26 const int INF = 0x3f3f3f3f;
27 const int MOD = 1e9 + 7;
28 typedef pair<int, int> PII;
29 typedef long long ll;
30 typedef unsigned long long ull;
31 int ans;
32 struct node {
33
       int 1, r;
34
       mutable ll val;
       node(int L, int R = -1, ll V = 0):l(L), r(R), val(V) {}
35
36
       bool operator < (const node &rhs) const {</pre>
```

```
return 1 < rhs.1;</pre>
37
        }
38
   };
39
40
   set<node> s;
41
   IT split(int pos) {
42
        IT it = s.lower_bound(node(pos));
43
        if (it != s.end() && it->l == pos) return it;
44
45
46
        int L = it -> l, R = it -> r;
        ll val = it->val;
47
48
        s.erase(it);
        s.insert(node(L, pos-1, val));
49
        return s.insert(node(pos, R, val)).first;
50
   }
51
52
   void push_down(int l, int r, ll val) {
53
        IT itr = split(r+1), itl = split(l);
54
        s.erase(itl, itr);
55
        s.insert(node(l, r, val));
56
  }
57
58
   void add(int 1, int r, 11 val) {
59
60
        IT itr = split(r+1), itl = split(l);
61
        for (; itl != itr; ++itl)
            itl->val += val;
62
63 }
64
   ll rank(int l, int r, int k) {
        vector<pair<ll, int>> vp;
66
67
        IT itr = split(r+1), itl = split(l);
        vp.clear();
68
69
        for (; itl != itr; ++itl) {
            vp.push_back(pair<ll, int>(itl->val, itl->r - itl->l + 1));
70
        }
71
        sort(vp.begin(), vp.end());
72
73
        for (vector<pair<ll,int> >::iterator it = vp.begin(); it != vp.end(); ++it) {
74
            k -= it->second;
            if (k \ll 0)
75
                return it->first;
76
        }
77
   }
78
79
80
   ll ksm(ll a, ll b, ll mod) {
        ll res = 1, ans = a \% \mod;
81
82
        while (b) {
            if (b & 1)
83
                res = res * ans % mod;
84
            ans = ans * ans \% mod;
85
86
            b >>= 1;
87
88
        return res;
89
   }
90
   11 sum(int l, int r, int ex, int mod) {
91
92
        IT itr = split(r+1), itl = split(l);
        11 \text{ res} = 0;
93
94
        for (; itl != itr; ++itl)
            res = (res + (ll)(itl->r - itl->l + 1) * ksm(itl->val, ll(ex), ll(mod))) % mod;
95
```

```
96
       return res;
97 }
   3.12
          可持久化并查集
1 #include <iostream>
2 #include <cstring>
3 #include <cstdio>
 4 #include <cmath>
5 #include <algorithm>
6 #include <queue>
7 #include <stack>
8 #include <string>
9 #include <set>
10 #include <map>
11 #include <bitset>
12 using namespace std;
13 #define ACM_LOCAL
14
15 const int INF = 0x3f3f3f3f;
16 const int N = 2e5 + 5;
17 typedef pair<int, int> PII;
18 typedef long long ll;
19 typedef unsigned long long ull;
20 typedef long double ld;
21
22
   struct node {
       int 1, r;
23
24
       int val;
   }t[N * 40 * 2];
25
26
27
   int cnt, tot, rootdep[N], rootfa[N], n, m;
28
29
   void build(int 1, int r, int &now) {
       now = ++cnt;
30
       if (l == r) {
31
32
            t[now].val = ++tot;
33
            return:
34
       }
       int mid = (l + r) \gg 1;
35
       build(l, mid, t[now].l);
36
       build(mid + 1, r, t[now].r);
37
   }
38
39
   void update(int l, int r, int ver, int &now, int pos, int val) {
40
       t[now = ++cnt] = t[ver];
41
42
       if (l == r) {
            t[now].val = val;
43
            return;
44
       }
45
       int mid = (l + r) \gg 1;
46
       if (pos <= mid) update(l, mid, t[ver].l, t[now].l, pos, val);</pre>
47
48
       else update(mid + 1, r, t[ver].r, t[now].r, pos, val);
   }
49
50
   int query(int l, int r, int &now, int pos) {
51
        if (l == r) return t[now].val;
52
       int mid = (l + r) >> 1;
53
```

```
if (pos <= mid) return query(l, mid, t[now].l, pos);</pre>
54
        else return query(mid + 1, r, t[now].r, pos);
55
   }
56
57
   int find(int ver, int x) {
58
        int fx = query(1, n, rootfa[ver], x);
59
        return fx == x ? x : find(ver, fx);
60
   }
61
62
   void merge(int ver, int x, int y) {
63
        x = find(ver-1, x);
64
65
        y = find(ver-1, y);
        if (x == y) {
66
            rootfa[ver] = rootfa[ver-1];
67
            rootdep[ver] = rootdep[ver-1];
68
69
        else {
70
            int depx = query(1, n, rootdep[ver-1], x);
71
            int depy = query(1, n, rootdep[ver-1], y);
72
            if (depx < depy) {</pre>
73
                update(1, n, rootfa[ver-1], rootfa[ver], x, y);
74
                rootdep[ver] = rootdep[ver-1];
75
76
77
            else if (depx > depy) {
78
                update(1, n, rootfa[ver-1], rootfa[ver], y, x);
                rootdep[ver] = rootdep[ver-1];
79
            }
80
            else {
81
                update(1, n, rootfa[ver-1], rootfa[ver], x, y);
82
                update(1, n, rootdep[ver-1], rootdep[ver], y, depy+1);
83
84
            }
85
        }
86
  }
          可持久化数组
   3.13
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ACM_LOCAL
4
  const int N = 1e6 + 5;
   typedef pair<int, int> PII;
   typedef long long 11;
6
7
   inline int read(){
8
9
       int s=0, w=1;
      char ch=getchar();
10
      while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
11
      while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();</pre>
12
      return s*w;
13
   }
   int NodeNum, n, m, a[N], rt[N];
15
16
   struct Node{
17
        int t[N << 5], lc[N << 5], rc[N << 5];</pre>
18
19
        void build(int 1, int r, int &now) {
20
            now = ++NodeNum;
21
22
            if (l == r) t[now] = a[l];
```

```
else {
23
                 int mid = (l + r) \gg 1;
24
                build(l, mid, lc[now]);
25
                build(mid + 1, r, rc[now]);
26
27
28
29
        void update(int pre, int l, int r, int &now, int pos, int val) {
            now = ++NodeNum;
30
            lc[now] = lc[pre], rc[now] = rc[pre], t[now] = t[pre];
31
            if (l == r) t[now] = val;
32
33
            else {
34
                int mid = (l + r) >> 1;
                if (mid >= pos) update(lc[pre], l, mid, lc[now], pos, val);
35
36
                else update(rc[pre], mid + 1, r, rc[now], pos, val);
            }
37
38
        void query(int pre, int l, int r, int &now, int pos) {
39
40
            now = ++NodeNum;
            lc[now] = lc[pre], rc[now] = rc[pre], t[now] = t[pre];
41
            if (l == r) printf("%d\n", t[now]);
42
            else {
43
                int mid = (l + r) >> 1;
44
                if (mid >= pos) query(lc[pre], l, mid, lc[now], pos);
45
46
                else query(rc[pre], mid + 1, r, rc[now], pos);
47
            }
48
        }
   }T;
49
50
51
   void solve() {
52
53
        n = read(), m = read();
        for (int i = 1; i <= n; i++) a[i] = read();</pre>
54
55
        T.build(1, n, rt[0]);
56
        for (int i = 1; i <= m; i++) {
57
            int opt, id, x, y;
58
59
            id = read(), opt = read();
            if (opt == 1) {
60
                x = read(), y = read();
61
62
                T.update(rt[id], 1, n, rt[i], x, y);
63
            }
            else {
64
                x = read();
65
66
                T.query(rt[id], 1, n, rt[i], x);
            }
67
68
        }
69
70 }
71
   signed main() {
73
        ios_base::sync_with_stdio(false);
74
        cin.tie(0);
75
        cout.tie(0);
76
   #ifdef ACM_LOCAL
        freopen("in.txt", "r", stdin);
freopen("out.txt", "w", stdout);
77
78
79
   #endif
80
        solve();
        return 0;
81
```

```
82 }
   3.14 平衡树(pb_ds
1 #include "bits/stdc++.h"
2 #include <ext/pb_ds/tree_policy.hpp>
3 #include <ext/pb_ds/assoc_container.hpp>
4 using namespace std;
5 using namespace __gnu_pbds;
6 #define int long long
7 tree<int, null_type, less<int>, rb_tree_tag, tree_order_statistics_node_update> t;
8 signed main()
9 {
10
       int n;
       cin >> n;
11
       while (n--)
12
13
           int k,s;
14
           cin >> k >> s;
15
           if (k == 1)
16
17
               cout << t.order_of_key(s)+1 << endl;</pre>
18
                                                              //order_of_key有几个比s小
19
20
           else if (k == 2){
                cout << *t.find_by_order(s-1) << endl;</pre>
                                                               //order是有几个比s小, s-1个比s小
21
           } else if (k == 3)
22
23
               if (t.find_by_order(t.order_of_key(s)-1) != t.end())
24
                    cout << *t.find_by_order(t.order_of_key(s)-1) << endl;</pre>
25
26
               else
                    cout << -2147483647 << endl;
27
           } else if (k == 4)
28
                if (t.upper_bound(s) != t.end())
29
                    cout << *t.upper_bound(s) << endl;</pre>
30
                                                               //未找到为t.end()
               else
31
                    cout << 2147483647 << endl;
32
           else if (k == 5)
33
34
               t.insert(s);
35
       }
36
  }
          扫描线
   3.15
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ACM_LOCAL
4 #define lc u << 1
5 #define rc u << 1 | 1
6 const int N = 2e5 + 5;
7 const int INF = 0x3f3f3f3f;
8 typedef unsigned long long ull;
9 typedef pair<int, int> PII;
10 typedef long long ll;
11
12 int n, m, p;
13 int v[N << 1];</pre>
```

14 **struct** node {

```
int 1, r;
15
        ll len;
16
        int cover;
17
18
   }t[N << 3];
19
   struct L{
20
21
        int x;
22
        int y1, y2;
23
        int state;
        bool operator < (L &rhs) {</pre>
24
25
            return x < rhs.x;</pre>
26
   }line[N << 1];
27
28
29
   void push_up(int u) {
        if (t[u].cover) t[u].len = t[u].r - t[u].l;
30
        else t[u].len = t[lc].len + t[rc].len;
31
   }
32
33
   void build(int u, int l, int r) {
34
        t[u].l = v[l], t[u].r = v[r];
35
        if (r - l <= 1) return;</pre>
36
37
        int mid = (l + r) \gg 1;
38
        build(lc, l, mid);
39
        build(rc, mid, r);
        push_up(u);
40
41 }
42
   void update(int u, int ql, int qr, int v) {
43
        if (ql \ll t[u].l \& qr \gg t[u].r) {
44
45
            t[u].cover += v;
            push_up(u);
46
47
            return;
48
        if (ql < t[rc].l) update(lc, ql, qr, v);</pre>
49
        if (qr > t[lc].r) update(rc, ql, qr, v);
50
51
        push_up(u);
52
   }
53
   void solve() {
54
        cin >> n;
55
        for (int i = 1; i <= n; i++) {
56
             int x1, y1, x2, y2;
57
            cin >> x1 >> y1 >> x2 >> y2;
58
            v[i] = y1, v[i+n] = y2;
59
60
            line[i] = \{x1, y1, y2, 1\};
61
            line[i + n] = \{x2, y1, y2, -1\};
62
        sort(v+1, v+(n<<1)+1);
63
64
        sort(line+1, line+(n<<1)+1);</pre>
65
        build(1, 1, n<<1);
66
        11 \text{ ans} = 0;
67
        for (int i = 1; i <= n<<1; i++) {
            ans += 111*t[1].len * (line[i].x - line[i-1].x);
68
            update(1, line[i].y1, line[i].y2, line[i].state);
69
70
71
        cout << ans << endl;</pre>
72
   }
73
```

```
signed main() {
       ios_base::sync_with_stdio(false);
75
76
       cin.tie(0);
       cout.tie(0);
77
   #ifdef ACM_LOCAL
78
       freopen("in.txt", "r", stdin);
freopen("out.txt", "w", stdout);
79
80
   #endif
81
       solve();
82
       return 0;
83
84 }
   3.16 树状数组
                  -----维树状数组
   //----
   struct BIT {
3
       int c[N];
       int lowbit(int x) {return x&-x;}
4
       void add(int x, int val) {for (int i = x; i < N; i += lowbit(i)) c[i] += val;}
5
6
       int ask(int x) {int res = 0; for (int i = x; i > 0; i -= lowbit(i)) res += c[i];
       return res;}
   };
7
   //-----二维树状数组
10 int c[N][N];
11
12 //单点修改,区间查询
  int lowbit(int x) {return x & -x;}
13
  void add(int x, int y, int v) {
       for (int i = x; i <= N; i += lowbit(i))</pre>
15
            for (int j = y; j \leftarrow N; j \leftarrow lowbit(j))
16
17
                c[i][j] += v;
18
   }
19
20 ll query(int x, int y) {
21
       11 \text{ res} = 0;
       for (int i = x; i > 0; i = lowbit(i))
22
            for (int j = y; j > 0; j -= lowbit(j))
23
24
                res += c[i][j];
25
       return res;
   }
26
27
28
   //区间修改,单点查询
   void add_s(int x1, int y1, int x2, int y2, int v) {
29
30
       add(x2+1, y2+1, v);
       add(x1, y1, v);
31
       add(x1, y2+1, -v);
32
       add(x2+1, y1, -v);
33
34
   }
   //区间修改,区间查询
   void add(int x,int y,int v) {
       for(int i = x; i<= n;i += lowbit(i))</pre>
37
38
            for(int j=y;j<=m;j+=lowbit(j)) {</pre>
                t1[i][j] += v;
39
                t2[i][j] += v*x;
40
                t3[i][j] += v*y;
41
42
                t4[i][j] += v*x*y;
           }
43
```

```
44
   }
   void modify(int x1,int y1,int x2,int y2,int v) {
45
46
       add(x1,y1,v);
47
       add(x1,y2+1,-v);
       add(x2+1,y1,-v);
48
49
       add(x2+1,y2+1,v);
50
   }
   int ask(int x, int y) {
51
52
       int res = 0;
53
        for(int i = x;i;i -= lowbit(i))
            for(int j=y;j;j -= lowbit(j))
54
55
                res += (x+1)*(y+1)*t1[i][j]-(y+1)*t2[i][j]-(x+1)*t3[i][j]+t4[i][j];
56
       return res;
   }
57
   int query(int x1,int y1,int x2,int y2) {
58
       return ask(x2,y2)-ask(x2,y1-1)-ask(x1-1,y2)+ask(x1-1,y1-1);
60
   }
   3.17
          树状数组套主席树
1 //单点修改动态区间k大
2 #include <bits/stdc++.h>
3 #define lowbit(i) i & -i
4 using namespace std;
5 typedef long long ll;
6 typedef pair<int, int> PII;
  const ll INF = 1e18;
7
8 \text{ const int } N = 1e5 + 10;
  const int M = 1e6 + 10;
  const int MOD = 1e9 + 7;
   struct Segtree {
11
12
       int ls, rs;
13
       int sum;
14 }t[N * 400];
  int tot, n, m, rt[N], a[N], cnt[2], tmp[2][20], b[N];
   void modify(int &now, int l, int r, int pos, int val) {
17
       if (!now) now = ++tot;
18
       if (l == r) {
19
            t[now].sum += val;
20
            return;
21
       }
       int mid = (l + r) \gg 1;
22
23
       if (pos <= mid) modify(t[now].ls, l, mid, pos, val);</pre>
       else modify(t[now].rs, mid + 1, r, pos, val);
24
       t[now].sum = t[t[now].ls].sum + t[t[now].rs].sum;
25
26
   void prepare_modify(int x, int val) {
27
28
       for (int i = x; i <= n; i += lowbit(i)) {</pre>
29
           modify(rt[i], 1, 10, a[x], val);//预处理出修改哪log棵主席树
       }
30
31
   }
   int query(int 1, int r, int k) {
32
33
       if (l == r) return l;
34
       int mid = (l + r) \gg 1, sum = 0;
       for (int i = 1; i <= cnt[1]; i++) sum += t[t[tmp[1][i]].ls].sum;</pre>
35
        for (int i = 1; i \le cnt[0]; i++) sum -= t[t[tmp[0][i]].ls].sum;
36
37
       if (k <= sum) {
            for (int i = 1; i <= cnt[1]; i++) tmp[1][i] = t[tmp[1][i]].ls;</pre>
38
```

```
for (int i = 1; i <= cnt[0]; i++) tmp[0][i] = t[tmp[0][i]].ls;</pre>
39
            return query(l, mid, k);
40
        } else {
41
            for (int i = 1; i <= cnt[1]; i++) tmp[1][i] = t[tmp[1][i]].rs;
for (int i = 1; i <= cnt[0]; i++) tmp[0][i] = t[tmp[0][i]].rs;</pre>
42
43
            return query(mid + 1, r, k - sum);
44
45
        }
   }
46
   int prepare_query(int l, int r, int k) {//处理出需要进行加减操作的log棵主席树
47
        memset(tmp, 0, sizeof tmp);
48
49
        memset(cnt, 0, sizeof cnt);
50
        for (int i = r; i > 0; i -= lowbit(i)) tmp[1][++cnt[1]] = rt[i];
        for (int i = l - 1; i > 0; i -= lowbit(i)) tmp[0][++cnt[0]] = rt[i];
51
        return query(1, 10, k);
52
   }
53
   void solve() {
54
55
        cin >> n >> m;
        for (int i = 1; i <= n; i++) {
56
57
            cin >> a[i];
58
        for (int i = 1; i <= n; i++) prepare_modify(i, 1);</pre>
59
        while(m--) {
60
            char op; cin >> op;
61
62
            if (op == '0') {
63
                 int l, r, k; cin >> l >> r >> k;
                 cout << prepare_query(l, r, k) << endl;</pre>
64
65
            } else {
66
                 int x, y; cin >> x >> y;
                 prepare_modify(x, -1);
67
68
                 a[x] = y;
69
                 prepare_modify(x, 1);
70
            }
71
        }
   }
72
73
74 //区间修改,区间K大
75 #include <bits/stdc++.h>
76 #define lowbit(i) i & -i
77 #define Debug(x) cout << x << endl
78 #define fi first
79 #define se second
80 using namespace std;
81 typedef long long ll;
82 typedef pair<int, int> PII;
83 const ll INF = 1e18;
84 const int N = 5e4 + 10;
85 const int M = 1e6 + 10;
86 const int MOD = 1e9 + 7;
87 struct Segment {
88
        int ls, rs;
        ll sum, lazt;
   } t[N * 400];
   int rt[N << 2], tot, n, m, b[N], len;</pre>
91
92
   void push_down(int now, int 1, int r) {
93
94
        if (t[now].lazt) {
95
            int mid = (l + r) >> 1;
96
            if (!t[now].ls) t[now].ls = ++tot;
            if (!t[now].rs) t[now].rs = ++tot;
97
```

```
t[t[now].ls].lazt += t[now].lazt;
98
             t[t[now].rs].lazt += t[now].lazt;
99
             t[t[now].ls].sum += 1ll*(mid - l + 1) * t[now].lazt;
100
             t[t[now].rs].sum += 1ll*(r - mid) * t[now].lazt;
101
102
             t[now].lazt = 0;
103
         }
104
    }
105
    void update(int &now, int ql, int qr, int l, int r) {
106
         if (!now) now = ++tot;
107
108
         if (ql \ll l \& qr \gg r) {
109
             t[now].sum += r - l + 1;
             t[now].lazt++;
110
             return;
111
112
         push_down(now, l, r);
113
         int mid = (l + r) >> 1;
114
115
         if (ql <= mid) update(t[now].ls, ql, qr, l, mid);</pre>
         if (qr > mid) update(t[now].rs, ql, qr, mid + 1, r);
116
         t[now].sum = t[t[now].ls].sum + t[t[now].rs].sum;
117
118 }
119
    void add(int u, int ql, int qr, int pos, int l, int r) {
120
121
         update(rt[u], ql, qr, 1, n);
122
         if (l == r) return;
         int mid = (l + r) \gg 1;
123
         if (pos \leftarrow mid) add(u \leftarrow 1, ql, qr, pos, l, mid);
124
         else add(u \ll 1 \mid 1, ql, qr, pos, mid + 1, r);
125
    }
126
127
    ll getsum(int &now, int ql, int qr, int l, int r) {
128
         if (!now) return 0;
129
130
         if (ql <= l && qr >= r) return t[now].sum;
         push_down(now, l, r);
131
         int mid = (l + r) \gg 1;
132
         11 \text{ ans} = 0;
133
134
         if (ql <= mid) ans += getsum(t[now].ls, ql, qr, l, mid);</pre>
135
         if (qr > mid) ans += getsum(t[now].rs, ql, qr, mid + 1, r);
136
         return ans;
137
    }
138
    int query(int u, int ql, int qr, ll k, int l, int r) {
139
         if (l == r) return b[l];
140
141
         int mid = (l + r) \gg 1;
         ll num = getsum(rt[u<<1|1], ql, qr, 1, n);</pre>
142
         if (k > num) return query(u<<1, ql, qr, k - num, l, mid);</pre>
143
144
         else return query(u << 1|1, ql, qr, k, mid + 1, r);
145 }
146
147
    struct Query {
148
         ll op, l, r, c;
149
    } q[N];
150
    void solve() {
151
         cin >> n >> m;
152
         for (int i = 1; i \le m; i++) {
153
             cin >> q[i].op;
154
             if (q[i].op == 1) {
155
                 cin >> q[i].l >> q[i].r >> q[i].c;
156
```

```
b[++len] = q[i].c;
157
158
            } else {
                 cin >> q[i].l >> q[i].r >> q[i].c;
159
            }
160
161
162
        sort(b + 1, b + len + 1);
        len = unique(b + 1, b + len + 1) - b - 1;
163
        for (int i = 1; i <= m; i++) {
164
            if (q[i].op == 1) {
165
                 q[i].c = lower\_bound(b + 1, b + len + 1, q[i].c) - b;
166
                 add(1, q[i].l, q[i].r, q[i].c, 1, len);
167
168
            } else {
                 printf("%d\n", query(1, q[i].l, q[i].r, q[i].c, 1, len));
169
170
            }
171
        }
172
    3.18
           替罪羊树 (平衡树)
 1 #include<bits/stdc++.h>
 2 #define lc (son[p][0])
 3 #define rc (son[p][1])
 4 using namespace std;
 5 //#define ACM_LOCAL
   const int N = 4e5 + 10;
 7
    struct SGT {
 8
        const double alpha = 0.75;
 9
        int st[N], top, tot, rt, son[N][2], fa[N], val[N], siz[N], all[N];
10
        bool in[N];
11
12
        inline int get() { return top ? st[top--] : ++tot; }
13
14
        void del(int t) { st[++top] = t; }
15
        inline void pushup(int p) { siz[p] = siz[lc] + siz[rc] + in[p], all[p] = all[lc] +
16
        all[rc] + 1; }
17
18
        inline bool check(int p) { return (all[lc] >= all[p] * alpha) || (all[rc] >= all[p]
         * alpha); }
19
        inline int newnode(int v = 0, int pa = 0) {
20
21
            int p = get();
            lc = rc = 0, val[p] = v, siz[p] = all[p] = 1, in[p] = 1, fa[p] = pa;
22
23
            return p;
24
        }
25
26
        inline void getpos(int p, vector<int> &v) {
27
            if (!p)return;
28
            getpos(lc, v);
            if (in[p])v.push_back(p);
29
30
            else del(p);
31
            getpos(rc, v);
32
        }
33
        inline int build(int l, int r, vector<int> v) {
34
35
             if (l >= r)return 0;
            int mid = l + r \gg 1, p = v[mid];
36
            lc = build(l, mid, v), rc = build(mid + 1, r, v), fa[lc] = fa[rc] = p;
37
```

```
return pushup(p), p;
38
        }
39
40
        inline void rebuild(int &p) {
41
42
            static vector<int> v;
            v.clear();
43
            int pa = fa[p];
44
            getpos(p, v);
45
            fa[(p = build(0, v.size(), v))] = pa;
46
        }
47
48
49
        inline int rank(int v) {
            int p = rt, ret = 1;
50
            while (p) {
51
                 if (v \le val[p])p = lc;
52
                 else ret += siz[lc] + in[p], p = rc;
53
54
55
            return ret;
        }
56
57
        inline int kth(int k) {
58
59
            int p = rt;
            while (p) {
60
61
                 if (siz[lc] + 1 == k \&\& in[p]) break;
62
                 if (siz[lc] >= k)p = lc;
                 else k \rightarrow siz[lc] + in[p], p = rc;
63
            }
64
65
            return val[p];
        }
66
67
        inline int insert(int &p, int v) {
68
            if (!p)return p = newnode(v), 0;
69
70
            ++siz[p], ++all[p];
71
            int ret;
            ret = insert(v <= val[p] ? lc : rc, v), pushup(p);</pre>
72
            if (check(p))ret = p;
73
74
            return ret;
75
        }
76
77
        inline void insert(int v) {
78
            int p = insert(rt, v);
            if (!p)return;
79
            if (p == rt)rebuild(rt);
80
81
            else {
                 int f = fa[p];
82
83
                 if (p == son[f][0])rebuild(son[f][0]);
84
                 else rebuild(son[f][1]);
            }
85
        }
86
87
88
         void erase(int p, int k) {
89
            --siz[p];
            if (in[p] \&\& k == siz[lc] + in[p]) {
90
                 in[p] = 0;
91
                 return;
92
93
            if (k <= siz[lc])erase(lc, k);</pre>
94
95
            else erase(rc, k - siz[lc] - in[p]);
        }
96
```

```
97
         void erase(int v) {
98
            erase(rt, rank(v));
99
            if (siz[rt] < alpha * all[rt])rebuild(rt);</pre>
100
101
102
         int pre(int v) { return kth(rank(v) - 1); }
103
104
         int suf(int v) { return kth(rank(v + 1)); }
105
106
    } tzy;
    3.19 线段树(区间最长连续段)
    int rt[N], cnt;//主席树版本
 1
    struct Tree {
 3
        int ls, rs;
        int lsum, rsum, tsum;
 4
    }hjt[N*50];
 5
 6
    void push_up(int now, int l, int r) {
        int mid = (l + r) \gg 1;
 7
        hjt[now].lsum = hjt[hjt[now].ls].lsum;
 8
        if (hjt[hjt[now].ls].lsum == mid - l + 1) hjt[now].lsum += hjt[hjt[now].rs].lsum;
 9
        hjt[now].rsum = hjt[hjt[now].rs].rsum;
 10
        if (hjt[hjt[now].rs].rsum == r - mid) hjt[now].rsum += hjt[hjt[now].ls].rsum;
11
        hjt[now].tsum = max(hjt[hjt[now].ls].tsum, hjt[hjt[now].rs].tsum);
12
        hjt[now].tsum = max(hjt[now].tsum, hjt[hjt[now].ls].rsum + hjt[hjt[now].rs].lsum);
13
14
    void modify(int &now, int pre, int l, int r, int pos, int val) {
15
16
        now = ++cnt;
        hit[now] = hit[pre];
17
        if (l == r) {
18
            hjt[now].lsum = hjt[now].rsum = hjt[now].tsum = 1;
19
20
            return:
        }
21
        int mid = (l + r) \gg 1;
22
        if (pos <= mid) modify(hjt[now].ls, hjt[pre].ls, l, mid, pos, val);</pre>
23
        else modify(hjt[now].rs, hjt[pre].rs, mid+1, r, pos, val);
24
25
        push_up(now, l, r);
26
    int query(int now, int l, int r, int ql, int qr) {
27
        if (ql <= l && qr >= r) return hjt[now].tsum;
28
29
        int mid = (l + r) >> 1;
        int ans = 0;
30
31
        if (ql <= mid) ans = max(ans, query(hjt[now].ls, l, mid, ql, qr));</pre>
32
        if (qr > mid) ans = max(ans, query(hjt[now].rs, mid+1, r, ql, qr));
        ans = max(ans, min(mid-ql+1, hjt[hjt[now].ls].rsum) + min(qr-mid, hjt[hjt[now].rs].
33
        lsum));
        return ans;
34
35 }
    3.20
           线段树 (最长子序和)
 1 #include<bits/stdc++.h>
 2 using namespace std;
 3 const int N=5e5+1000;
 4 int n,m,x,y;
 5 int a[N];
```

```
struct Node {
7
        int l,r;
8
        int sum;
        int lmax;
9
10
        int rmax;
        int tmax;
11
   }tr[N<<2];</pre>
12
13
   void push_up(Node &u,Node &l,Node &r) {
14
        u.sum=1.sum+r.sum;
15
16
        u.lmax=max(l.lmax,l.sum+r.lmax);
17
        u.rmax=max(r.rmax,r.sum+l.rmax);
        u.tmax=max(max(l.tmax,r.tmax),l.rmax+r.lmax);
18
19
   }
20
   void push_up(int u) {
21
        push_up(tr[u],tr[u<<1],tr[u<<1|1]);</pre>
22
23
24
   void build(int u,int l,int r) {
25
        tr[u].l = l, tr[u].r = r;
26
27
        if(l==r) {
28
             tr[u]={1,r,a[r],a[r],a[r],a[r]};
29
             return ;
30
        int mid = (l+r)>>1;
31
32
        build(u<<1, l, mid);</pre>
33
        build(u<<1|1, mid+1, r);
34
        push_up(u);
35
36
37
   void update(int u,int x,int v) {
38
39
        if(tr[u].l==tr[u].r) {
             tr[u]=\{x,x,v,v,v,v\};
40
             return ;
41
42
        }
43
        int mid = (tr[u].l + tr[u].r)>>1;
        if(x<=mid) update(u<<1, x, v);</pre>
44
        else update(u<<1|1, x, v);
45
46
        push_up(u);
47
    Node query(int u,int ql,int qr) {
48
49
        if(ql<=tr[u].l&&qr>=tr[u].r) return tr[u];
50
        int mid=tr[u].l+tr[u].r>>1;
51
        if(qr<=mid) return query(u<<1,ql,qr);</pre>
        else if(ql>mid) return query(u<<1|1,ql,qr);</pre>
52
        else
53
        {
54
55
             auto left=query(u<<1,ql,qr);</pre>
56
             auto right=query(u<<1|1,ql,qr);</pre>
57
            Node res;
            push_up(res,left,right);
58
59
             return res;
60
        }
61
   }
```

3.21 线段树

```
namespace Seg {
1
   #define ls (u<<1)
2
   #define rs (u<<1|1)</pre>
   #define mid ((l+r)>>1)
       ll sum[N<<2], tag[N<<2];
5
6
      void push_up(int u) {
7
           sum[u] = sum[u << 1] + sum[u << 1|1];
8
      void push_down(int u, int l, int r) {
9
10
           if (tag[u]) {
               tag[u << 1] += tag[u];
11
12
               tag[u << 1|1] += tag[u];
               sum[u << 1] += 1ll*(mid-l+1)*tag[u];
13
               sum[u << 1|1] += 1|1|*(r-mid)*tag[u];
14
               tag[u] = 0;
15
           }
16
17
       void modify(int u, int ql, int qr, int l, int r, int val) {
18
19
           if (ql \ll l \& qr \gg r) {
               sum[u] += 1ll*(r-l+1)*val;
20
               tag[u] += val;
21
22
               return;
           }
23
           push_down(u, l, r);
24
25
           if (ql <= mid) modify(ls, ql, qr, l, mid, val);</pre>
26
           if (qr > mid) modify(rs, ql, qr, mid+1, r, val);
27
           push_up(u);
28
      }
29
30
   #undef mid
   #undef ls
32
   #undef rs
33 }
34
  一些技巧
35
36
37
   1)找到最左端满足条件的位置:
38
39
   1. 二分+线段树 复杂度是nlogn^2
40
41
   2. 直接在线段树上找 复杂度是nlogn
42
   int query(int u, int L, int R, int val) {
43
44
        if (t[u].mi > val) return 2e9;
        if (t[u].l == t[u].r)return t[u].l;
45
        int mid = t[u].l + t[u].r >> 1;
46
        if (L <= t[u].l && R >= t[u].r) {
47
            if (t[u<<1].mi <= val) return query(u<<1, t[u].l, mid, val);</pre>
48
            else return query(u<<1|1, mid + 1, t[u].r, val);
49
        }
50
        else {
51
52
            if (R <= mid)return query(u<<1, L, R, val);</pre>
            else if (L > mid)return query(u<<1|1, L, R, val);</pre>
53
            else return min(query(u<<1, L, mid, val), query(u<<1|1, mid + 1, R, val));</pre>
54
55
        }
   }
56
57
```

```
2)势能线段树(均摊时间复杂度)
59
    1. 区间与, 单点修改, 区间max
60
61
62
    struct node {
63
         int 1, r;
64
         int Or, And, Max, tag;
    }t[N<<2];
65
    int a[N];
66
    void push_up(int u) {
67
         t[u].0r = t[u<<1].0r | t[u<<1|1].0r;
68
69
         t[u].And = t[u<<1].And & t[u<<1|1].And;
         t[u].Max = max(t[u<<1].Max, t[u<<1|1].Max);
70
    }
71
    void push_down(int u) {
72
         if (t[u].tag != 0) {
73
             t[u << 1].tag += t[u].tag;
74
75
             t[u << 1|1].tag += t[u].tag;
             t[u << 1].Max += t[u].tag;
76
             t[u << 1|1].Max += t[u].tag;
77
             t[u << 1].And += t[u].tag;
78
79
             t[u << 1|1].And += t[u].tag;
80
             t[u << 1].0r += t[u].tag;
             t[u << 1|1].0r += t[u].tag;
81
82
             t[u].tag = 0;
         }
83
    }
84
    void build(int u, int l, int r) {
85
         t[u].l = l, t[u].r = r;
86
         if (l == r) {
87
             t[u].0r = t[u].And = t[u].Max = a[l];
88
89
             return;
90
         }
         int mid = (l + r) \gg 1;
91
92
         build(u<<1, 1, mid);</pre>
         build(u<<1|1, mid+1, r);
93
94
         push_up(u);
95
    }
    void modify(int u, int ql, int qr, int val) {
96
         if ((t[u].0r & val) == t[u].0r) return;
97
         if (ql \le t[u].l \& qr \ge t[u].r \& (t[u].0r \& val) - t[u].0r == (t[u].And \& val) -
98
         t[u].And) {
             int tmp = (t[u].0r \& val) - t[u].0r;
99
100
             t[u].And += tmp;
             t[u].0r += tmp;
101
             t[u].Max += tmp;
102
103
             t[u].tag += tmp;
104
             return;
105
106
         push_down(u);
107
         int mid = (t[u].l + t[u].r) >> 1;
108
         if (ql <= mid) modify(u<<1, ql, qr, val);</pre>
         if (qr > mid) modify(u<<1|1, ql, qr, val);</pre>
109
110
         push_up(u);
111
    void update(int u, int pos, int val) {
112
113
         if (t[u].l == t[u].r) {
114
             t[u].Max = t[u].Or = t[u].And = val;
115
             return;
```

```
116
        push_down(u);
117
        int mid = (t[u].l + t[u].r) >> 1;
118
        if (pos <= mid) update(u<<1, pos, val);</pre>
119
        else update(u<<1|1, pos, val);</pre>
120
        push_up(u);
121
122
int query(int u, int ql, int qr) {
        if (ql \leftarrow t[u].l \& qr \rightarrow t[u].r) return t[u].Max;
124
        push_down(u);
125
126
        int mid = (t[u].l + t[u].r) >> 1;
127
        int ans = 0;
        if (ql \ll mid) ans = max(ans, query(u\ll1, ql, qr));
128
        if (qr > mid) ans = max(ans, query(u<<1|1, ql, qr));
129
130
        return ans;
131
132
133 2. 区间开根号,区间加,区间和
134
135
    势能=max-min
136
137
   3. 区间质因子问题
    3.22
          线段树 +dfs 序
 1 int in[N], out[N];
    vector<int> g[N];
    void dfs(int u) {
 3
        in[u] = ++tot;
 4
        for (auto i : g[u]) {
 5
 6
             dfs(i);
 7
        out[u] = tot;
 8
    }
 9
 10
    修改子树内信息[in[u], out[u]]
11
    3.23
           线段树标记永久化
 1 typedef long long ll;
    const int N = 1e5 + 5, M = 5e5 + 5, INF = 0x3f3f3f3f3f;
    const int MOD = 1e9 + 7;
 3
 5 int n, q, a[N];
 6
    struct Segment_tree {
 7
    #define ls u << 1
    #define rs u << 1 | 1</pre>
 9
        int L[N << 2], R[N << 2];</pre>
10
        11 \text{ sum}[N << 2], \text{ lazy}[N << 2];
11
12
13
        void push_up(int u) {
             sum[u] = sum[ls] + sum[rs] + (R[u] - L[u] + 1) * lazy[u];
14
15
16
        void build(int u, int l, int r) {
17
             L[u] = 1, R[u] = r;
18
```

```
if (l == r) {
19
                sum[u] = a[1];
20
21
                return;
22
23
            int m = (l + r) >> 1;
            build(ls, l, m);
24
25
            build(rs, m + 1, r);
            push_up(u);
26
27
       }
28
29
       void update(int u, int ql, int qr, ll k) {
30
            if (ql <= L[u] && R[u] <= qr) {</pre>
                sum[u] += (R[u] - L[u] + 1) * k;
31
32
                lazy[u] += k;
33
                return;
            }
34
            int m = (L[u] + R[u]) >> 1;
35
36
            if (ql <= m) update(ls, ql, qr, k);</pre>
            if (qr > m) update(rs, ql, qr, k);
37
38
            push_up(u);
       }
39
40
       11 query(int u, int ql, int qr, ll tg) {
41
42
            if (ql \le L[u] \& R[u] \le qr) return sum[u] + (R[u] - L[u] + 1) * tg;
43
            int m = (L[u] + R[u]) >> 1;
            ll ans = 0;
44
            if (ql <= m) ans += query(ls, ql, qr, tg + lazy[u]);</pre>
45
            if (qr > m) ans += query(rs, ql, qr, tg + lazy[u]);
46
            return ans;
47
       }
48
49
50
   #undef ls
   #undef rs
51
52 } tr;
   3.24 线段树合并 + 动态开点
1 #pragma GCC optimize (2)
2 #include <bits/stdc++.h>
3 #define ACM_LOCAL
4 using namespace std;
5 const int N = 1e5 + 5;
6
   const int M = 6e6 + 5;
   struct edge {
7
8
        int to, next;
   }e[2000010];
9
10
   struct tree {
11
12
       int 1, r;
       int maxx, id;
13
14
   }t[M];
15
16
   struct query {
17
       int x, y, z;
18
   }q[N];
19
  int tmp[N], n, m, h[N], cnt, son[N], root[N], siz[N], top[N], fa[N], tot, d[N], idx,
       all, ans[N], k;
```

```
21
22
   void add(int u, int v) {
        e[cnt].to = v;
23
        e[cnt].next = h[u];
24
25
        h[u] = cnt++;
   }
26
27
   void dfs1(int u) {
28
29
        son[u] = -1;
30
        siz[u] = 1;
31
        for (int i = h[u]; \sim i; i = e[i].next) {
32
            int v = e[i].to;
            if (!d[v]) {
33
                d[v] = d[u] + 1;
34
                fa[v] = u;
35
                dfs1(v);
36
37
                siz[u] += siz[v];
                if (son[u] == -1 \mid | siz[v] > siz[son[u]]) son[u] = v;
38
            }
39
40
        }
41 }
42
   void dfs2(int u, int t) {
43
44
        top[u] = t;
45
        if (son[u] == -1) return;
        dfs2(son[u], t);
46
        for (int i = h[u]; \sim i; i = e[i].next) {
47
            int v = e[i].to;
48
            if (v != son[u] && v != fa[u]) dfs2(v, v);
49
        }
50
   }
51
52
   int lca(int u, int v) {
53
        while (top[u] != top[v]) {
54
            if (d[top[u]] > d[top[v]])
55
                u = fa[top[u]];
56
57
            else
58
                v = fa[top[v]];
59
        return d[u] > d[v] ? v : u;
60
   }
61
62
   void push_up(int u) {
63
64
        int ls = t[u].l, rs = t[u].r;
        if (t[ls].maxx >= t[rs].maxx) t[u].maxx = t[ls].maxx, t[u].id = t[ls].id;
65
66
        else t[u].maxx = t[rs].maxx, t[u].id = t[rs].id;
67
   }
68
   void update(int &now, int l, int r, int pos, int val) {
69
70
        if (!now) now = ++idx;
71
        if (l == r \&\& l == pos){
72
            t[now].maxx += val;
            t[now].id = 1;
73
74
            return;
75
        int mid = (l + r) >> 1;
76
        if (pos <= mid) update(t[now].1, 1, mid, pos, val);</pre>
77
78
        else update(t[now].r, mid + 1, r, pos, val);
        push_up(now);
79
```

```
80 }
81
    void merge(int &x, int &y, int l, int r) {
82
83
         if (!x) return;
84
         if (!y) { y = x; return; }
85
         if (l == r) {
86
             t[y].maxx += t[x].maxx;
87
             return;
         }
88
89
         int mid = (l + r) >> 1;
         merge(t[x].1, t[y].1, 1, mid);
90
91
         merge(t[x].r, t[y].r, mid + 1, r);
92
         push_up(y);
    }
93
94
    void dfs_merge(int u, int fa) {
95
         for (int i = h[u]; ~i; i = e[i].next) {
96
97
             int v = e[i].to;
             if (v == fa) continue;
98
             dfs_merge(v, u);
99
             merge(root[v], root[u], 1, k);
100
101
         if (t[root[u]].maxx == 0) ans[u] = 0;
102
103
         else {
104
             ans[u] = tmp[t[root[u]].id];
105
106
    }
    void solve() {
107
         memset(h, -1, sizeof h);
108
         scanf("%d %d", &n, &m);
for (int i = 1; i <= n - 1; i++) {
109
110
111
             int x, y;
             scanf("%d %d", &x, &y);
112
             add(x, y), add(y, x);
113
114
         for (int i = 1; i <= m; i++) {
115
116
             scanf("%d %d %d", &q[i].x, &q[i].y, &q[i].z);
117
             tmp[++k] = q[i].z;
         }
118
         d[1] = 1;
119
120
         dfs1(1);
         dfs2(1, 1);
121
         sort(tmp + 1, tmp + 1 + k);
122
123
         k = unique(tmp + 1, tmp + 1 + k) - (tmp + 1);
         for (int i = 1; i <= m; i++) {
124
             q[i].z = lower\_bound(tmp + 1, tmp + 1 + k, q[i].z) - tmp;
125
126
             update(root[q[i].x], 1, k, q[i].z, 1);
             update(root[q[i].y], 1, k, q[i].z, 1);
127
             update(root[lca(q[i].x, q[i].y)], 1, k, q[i].z, -1);
128
129
             if (fa[lca(q[i].x, q[i].y)]) update(root[fa[lca(q[i].x, q[i].y)]], 1, k, q[i].z)
          -1);
130
131
         dfs_merge(1, 0);
132
         for (int i = 1; i \le n; i++) printf("%d\n", ans[i]);
133
134
135
    signed main() {
136
         ios_base::sync_with_stdio(false);
137
         cin.tie(nullptr);
```

```
138
        cout.tie(nullptr);
    #ifdef ACM_LOCAL
139
        freopen("in.txt", "r", stdin);
//freopen("out.txt", "w", stdout);
140
141
        signed test_index_for_debug = 1;
142
        char acm_local_for_debug = 0;
143
        do {
144
             if (acm_local_for_debug == '$') exit(0);
145
             if (test_index_for_debug > 20)
146
                 throw runtime_error("Check the stdin!!!");
147
             auto start_clock_for_debug = clock();
148
149
             solve();
             auto end_clock_for_debug = clock();
150
             cout << "Test " << test_index_for_debug << " successful" << endl;</pre>
151
             cerr << "Test " << test_index_for_debug++ << " Run Time: "</pre>
152
                 << double(end_clock_for_debug - start_clock_for_debug) / CLOCKS_PER_SEC <<</pre>
153
        "s" << endl;
                           -----" << endl;
             cout <<
154
        } while (cin >> acm_local_for_debug && cin.putback(acm_local_for_debug));
155
    #else
156
        solve();
157
    #endif
158
        return 0;
159
160
   }
    3.25
           Hash 表
 1 //手写哈希表
 2
    struct HashSet {
        const int mod = 1000009;
 3
 4
        struct node {
 5
             int k, v, nex;
        } buf[N];
 6
        int h[N], tot;
 7
 8
        void ins(int x) {
 9
             int pos = x \% mod;
 10
             for (int i=h[pos]; i; i=buf[i].nex) {
                 if (buf[i].k == x) { buf[i].v++; return ; }
11
12
13
             buf[++tot] = (node)\{x, 1, h[pos]\};
            h[pos] = tot;
14
15
        int find(int x) {
16
             int pos = x \% mod;
17
18
             for (int i=h[pos]; i; i=buf[i].nex) {
                 if (buf[i].k == x) return buf[i].v;
19
20
             }
21
             return 0;
22
    } H;
23
24
25 //支持单点修改的hash
26 const int N = 2e6 + 10, M = 1e6 + 10;
27 const int mod = 1e9 + 7;
28 char s[N];
29 int n, m;
30 ll f[N];
31 struct BIT {
```

```
ll c[N];
32
        int lowbit(int x) {return x&-x;}
33
        void add(int x, ll val) {
34
             for (int i = x; i \le n; i += lowbit(i)) c[i] = (c[i] + val) \% mod;
35
36
        ll qry(int x) {
37
             ll res = 0;
38
             for (int i = x; i > 0; i = lowbit(i)) res = (res + c[i]) % mod;
39
             return res;
40
41
        Il query(int l, int r) {
42
             return (qry(r) - qry(l-1) + mod) % mod;
43
44
        bool comp(int l1, int r1, int l2, int r2) {
45
             return query(l1, r1)*f[l2-l1] % mod == query(l2, r2);
46
47
        bool check(int l1, int r1, int l2, int r2) {
48
             if (l1 == r1) return true;
49
             int mid1 = (l1 + r1) >> 1;
50
             int mid2 = (12 + r2) >> 1;
51
            bool f1 = comp(l1, mid1, l2, mid2);
52
            bool f2 = comp(mid1+1, r1, mid2+1, r2);
53
             if (!f1 && !f2) return false;
54
55
             if (!f1) return check(l1, mid1, l2, mid2);
56
             else return check(mid1+1, r1, mid2+1, r2);
57
   }bit;
58
    void init() {
59
        f[0] = 1;
60
        for (int i = 1; i <= n; i++) f[i] = f[i-1] * 2333 % mod;
for (int i = 1; i <= n; i++) bit.add(i, s[i] * f[i] % mod);</pre>
61
62
63
   }
64
65 修改操作:将x位置的改为c
66 bit.add(x, (c-s[x]+mod)%mod*f[x]%mod);
67 	ext{ s[x]} = c;
```

4 数学

```
4.1 \mathbf{EX}_CRT
```

```
1 /***) О О ЙО ФО О О О ***/
2 #include<bits/stdc++.h>
3 using namespace std;
 4 #define ll long long
   const int MAXN = 2e5 + 5;
   ll exgcd(ll a, ll b, ll &x, ll &y) {
6
7
        if(b == 0) {
8
            x = 1; y = 0;
9
            return a;
10
        ll r = exgcd(b, a \% b, x, y);
11
        ll t = x; x = y; y = t - a / b * y;
12
        return r;
13
14 }
15
16
   int n;
   11 a[MAXN], b[MAXN];
17
18
   ll solve() {
19
20
        ll A = a[1], B = b[1], t, d, x, y;
        for(int i = 2; i \le n; i++) {
21
22
            d = exgcd(A, a[i], x, y);
            if((b[i] - B) % d) return -1;
23
            x = x * (b[i] - B) / d, t = a[i] / d, x = (x % t + t) % t;
24
            B = A * x + B;
25
            A = A / d * a[i];
26
            B %= A;
27
28
        while(B < 0) B += A;
29
        return (B + A) \% A;
30
   }
31
32
   int main() {
33
        scanf("%d",&n);
34
        for(int i = 1; i <= n; i++)</pre>
            scanf("%lld%lld", &a[i], &b[i]);
35
        printf("%lld\n",solve());
36
37
38
        return 0;
39 }
   4.2 \quad EX_Lucas
1
   // p不为质数,利用中国剩余定理结合求解
3
 4 #include <iostream>
5 using namespace std;
6
7
   typedef long long ll;
8
   const int N = 1e5 + 10;
9
10
   ll quick_pow(ll a, ll b, ll P) {
11
12
        ll ans = 1;
```

```
while(b) {
13
            if(b & 1)
14
                ans = ans * a \% P;
15
            a = a * a % P;
16
17
            b >>= 1;
18
       return ans % P;
19
   }
20
21
   ll ex_gcd(ll a, ll b, ll &x, ll &y) {
22
23
       ll res, t;
       if(!b) {
24
25
           x = 1;
           y = 0;
26
27
            return a;
       }
28
29
       res = ex_gcd(b, a \% b, x, y);
30
       t = x;
31
       x = y;
32
       y = t - (a / b) * y;
33
       return res;
34 }
35
36
   ll INV(ll a, ll mod) {
37
       11 x, y;
       ll d = ex_gcd(a, mod, x, y);
38
       return d ? (x % mod + mod) % mod : -1;
39
   }
40
41
   ll fac(ll n, ll P, ll pk) {// 阶乘除去质因子后模质数幂 (n / p^a) % pk
42
43
       if(!n) return 1;
       ll\ ans = 1;
44
45
       for(int i = 1;i < pk; i++) {// 第三部分: n!与p互质的乘积
46
            if(i % P)
                ans = ans * i % pk;
47
       }
48
49
       ans = quick_pow(ans, n / pk, pk) % pk; // 第三部分: n!与p互质的乘积,ans循环的次数为n/pk
50
       for(int i = 1;i <= n % pk; i++) {// 第四部分: 循环过后n!剩下的部分
            if(i \% P) ans = ans * i % pk;
51
52
       return ans * fac(n / P, P, pk) % pk; // 第一部分, p的幂, 个数为n/p;
53
                                                                             第二部分: (n/p)!
   }
54
55
   ll C(ll m, ll n, ll P, ll pk) {// 组合数模质数幂
56
       if(n < 0 \mid l \mid m < 0 \mid l \mid n > m) return 0;
57
       ll f1 = fac(m, P, pk), f2 = fac(n, P, pk), f3 = fac(m - n, P, pk), tmp = 0; // tmp
58
       = pk1 - pk2 - pk3
       for(ll i = m; i ; i /= P)
                                       tmp += i / P;
59
       for(ll i = n; i ; i /= P)
                                      tmp -= i / P;
60
       for(ll i = m - n; i ; i /= P) tmp -= i / P;
       return f1 * INV(f2, pk) % pk * INV(f3, pk) * quick_pow(P, tmp, pk) % pk;
62
63 }
64
   ll p[N], a[N];
65
   int cnt;
66
67
68
   11 CRT() {
69
       11 M = 1, ans = 0;
       for(int i = 1;i <= cnt; i++) M *= p[i];</pre>
70
```

```
for(int i = 1;i <= cnt; i++) {</pre>
71
72
            ll m = M / p[i];
            ans = (ans + a[i] * m % M * INV(m, p[i]) % M) % M;
73
74
75
        return (ans % M + M) % M;
    }
76
77
    11 EX_Lucas(ll m, ll n, ll P) {
78
        for(int i = 2; i * i <= P; i++) {
79
            if(P \% i == 0) {
80
81
                ll tmp = 1;
82
                while(P \% i == 0) {
                     tmp *= i;
83
                     P /= i;
84
85
                p[++cnt] = tmp;
86
                a[cnt] = C(m, n, i, tmp);
87
            }
88
89
        if(P > 1) {
90
            p[++cnt] = P;
91
            a[cnt] = C(m, n, P, P);
92
93
94
        return CRT();
95
    }
96 int main()
    {
97
        11 m, n, P;
98
        cin >> m >> n >> P;
99
        cnt = 0;
100
        cout << EX_Lucas(m, n, P) << endl;</pre>
101
102 }
    4.3 Lucas
    // p一定为质数
 1
 2
 3
    #include <iostream>
 4
    using namespace std;
 5
   typedef long long ll;
 6
 7
 8
    const int N = 1e7 + 10;
 9
10 ll p; // C(n,m) % p
11
13
14 void Init()
15
        f[0] = f[1] = inv[0] = inv[1] = invF[0] = invF[1] = 1;
16
        for(int i = 2; i <= p; i++)
17
18
            f[i] = f[i - 1] * i % p;
19
            inv[i] = (p - (p / i)) * inv[p % i] % p;
20
            invF[i] = invF[i - 1] * inv[i] % p;
21
        }
22
23 }
```

```
24
25 ll C(ll m, ll n)
26
        if(m < 0 \mid l \mid n < 0 \mid l \mid n > m)
27
28
            return 0;
        ll ans = f[m];
29
        ans = ans * invF[n] % p;
30
        ans = ans * invF[m - n] % p;
31
32
        return ans;
   }
33
34
35
   ll Lucas(ll m, ll n)
36
        if(n == 0)
37
            return 1;
38
        return Lucas(m / p, n / p) * C(m % p, n % p) % p; // 进制
39
   }
40
41
42 int main()
43  {
        11 m, n;
44
45
        cin >> m >> n >> p;
        Init();
46
47
        cout << Lucas(m, n) << endl;</pre>
48 }
   4.4 Miller abin
   // 二次探测定理:对素数p,满足x^2≡1(modp)的小于p的正整数px只有1或p-1.
2
3 #include <bits/stdc++.h>
4 using namespace std;
5 typedef long long ll;
6 const int N = 1e5 + 7;
7 const int times = 10;
8
  ll ksc(ll a, ll b, ll mod) {
9
10
        11 \text{ ans} = 0;
11
        while(b > 0) {
12
            if(b & 1) {
                ans = (ans + a) \% mod;
13
14
15
            a = (a << 1) \% mod;
16
            b >>= 1;
17
        return ans;
18
   }
19
20
   ll quick_pow(ll a, ll b, ll mod) {
21
22
        ll ans = 1, base = a;
        while(b != 0) {
23
            if(b & 1) {
24
25
                ans = ans * base % mod;
26
            base = base * base % mod;
27
28
            b >>= 1;
29
30
        return ans;
```

```
31 }
32
   bool Miller_Pabin(ll n)//Miller测试的主体结构
33
34
   {
        if(n < 2) return false;</pre>
35
        if(n == 2) return true;
36
        if(n & 1 == 0) return false;//对于偶数的优化
37
        ll k = 0,u = n - 1;//p为Miller测试的k, u为Miller测试的m
38
39
        while(u & 1 == 0){ // 把x拆成u*2^k
40
            u >>= 1;
41
42
            k++;
        }
43
        srand(time(NULL));
44
45
        ll x, pre; // pre为上次探测的x的值
46
47
        for(int i = 1; i <= times; i++) {</pre>
48
            x = rand() % (n - 1) + 1;
49
            x = quick_pow(x, u, n); // 先求出x^u(mod n)
50
            pre = x;
51
            for(int j = 1; j <= k; j++) {
52
                x = ksc(x, x, n);
53
54
                if(x == 1 \&\& pre != 1 \&\& pre != n - 1)
55
                     return false;
56
                pre = x;
            }
57
            if(x != -1)
58
            return false;
59
60
61
        return true;
   }
62
63
   int main() {
64
65
        ll n; cin >> n;
        cout << (Miller_Pabin(n) ? "Prime" : "Not a Prime") << endl;</pre>
66
67
        Min25 筛
   4.5
   #include <iostream>
2
3
   using namespace std;
   typedef long long ll;
5
6
7
   const int N = 1e5 + 10;
8
9
   namespace Min25 {
10
        int prime[N], id1[N], id2[N], flag[N], ncnt, m;
11
12
13
        ll g[N], sum[N], a[N], T, n;
14
        inline int ID(ll x) {
15
            return x \leftarrow T? id1[x] : id2[n / x];
16
        }
17
18
```

```
inline ll calc(ll x) {
19
            return x * (x + 1) / 2 - 1;
20
21
22
23
       inline ll f(ll x) {
24
            return x;
25
26
       inline void init() {
27
28
            ncnt = 0, m = 0;
29
            T = sqrt(n + 0.5);
30
            for (int i = 2; i <= T; i++) {
                if (!flag[i]) prime[++ncnt] = i, sum[ncnt] = sum[ncnt - 1] + i;
31
                for (int j = 1; j <= ncnt && i * prime[j] <= T; j++) {</pre>
32
                    flag[i * prime[j]] = 1;
33
                    if (i % prime[j] == 0) break;
34
                }
35
            }
36
            for (ll l = 1; l \ll n; l = n / (n / l) + 1) {
37
                a[++m] = n / 1;
38
                if (a[m] <= T) id1[a[m]] = m; else id2[n / a[m]] = m;
39
                g[m] = calc(a[m]);
40
41
            for (int i = 1; i <= ncnt; i++)</pre>
42
43
                for (int j = 1; j <= m && (ll)prime[i] * prime[i] <= a[j]; j++)</pre>
                    g[j] = g[j] - (ll)prime[i] * (g[ID(a[j] / prime[i])] - sum[i - 1]);
44
       }
45
46
       inline ll Solve(ll x) {
47
48
            if (x \ll 1) return x;
49
            return n = x, init(), g[ID(n)];
50
51
52 }
   4.6 村教筛
   #include <bits/stdc++.h>
2
3
   using namespace std;
4
5
   typedef long long 11;
6
   const int N = 1e6 + 10;
8
   unordered_map<int, ll> smu, sphi;
   bool isPrime[N];
9
10 int prime[N], num;
  11 mu[N], phi[N];
11
12
   void makeMobiusAndEuler(int siz) {
13
14
       mu[1] = phi[1] = 1;
       for (int i = 2; i <= siz; i++) {
15
            if (!isPrime[i]) prime[++num] = i, mu[i] = -1, phi[i] = i - 1;
16
            for (int j = 1; j <= num && i * prime[j] <= siz; j++) {
17
                isPrime[i * prime[j]] = 1;
18
19
                if (i % prime[j] == 0) {
                    mu[i * prime[j]] = 0;
20
                    phi[i * prime[j]] = phi[i] * prime[j];
21
```

```
22
                     break;
                }
23
                else {
24
                     phi[i * prime[j]] = phi[prime[j]] * phi[i];
25
26
                     mu[i * prime[j]] = -mu[i];
27
                }
            }
28
29
        for (int i = 1; i <= siz; i++) mu[i] += mu[i - 1], phi[i] += phi[i - 1];
30
31
   }
32
33
   11 getSmu(int n) {
        if (n < N) return mu[n];</pre>
34
        if (smu[n]) return smu[n];
35
        ll res = 1;
36
        for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
37
38
            r = n / (n / 1);
            res -= 1ll * (r - l + 1) * getSmu(n / l);
39
40
        return smu[n] = res;
41
   }
42
43
   ll getSphi(int n) {
44
        if (n < N) return phi[n];</pre>
45
46
        if (sphi[n]) return sphi[n];
        ll res = 1ll * n * (n + 1) / 2;
47
        for (unsigned int l = 2, r = 0; l <= n; l = r + 1) {
48
            r = n / (n / 1);
49
            res -= 1ll * (r - l + 1) * getSphi(n / l);
50
51
52
        return sphi[n] = res;
53 }
   4.7
         村教筛
   //筛大范围前缀和
  ll prime[N], mu[N], k;
   ll phi[N];
   bool is_prime[N];
   inline void init(int n) {
5
6
        memset(is_prime, true, sizeof is_prime);
        mu[1] = 1; phi[1] = 1;
7
        for (re int i = 2; i < n; ++i) {
8
            if (is_prime[i]) prime[++k] = i, mu[i] = -1, phi[i] = i-1;
9
            for (re int j = 1; j <= k && i * prime[j] < n; ++j) {
   is_prime[i * prime[j]] = false;</pre>
10
11
                if (i % prime[j] == 0) {
12
                     phi[i * prime[j]] = phi[i] * prime[j];
13
                     break;
14
15
                } else {
                     mu[i * prime[j]] = -mu[i];
16
                     phi[i * prime[j]] = phi[i] * (prime[j] - 1);
17
                }
18
            }
19
20
21
        for (re int i = 1; i < n; ++i) mu[i] += mu[i-1], phi[i] += phi[i-1];
22
   unordered_map<ll, ll> sum_mu, sum_phi;
```

```
inline ll GetSum_mu(ll n) {
       if (n <= 3e7) return mu[n];</pre>
25
       if (sum_mu[n]) return sum_mu[n];
26
       ll\ ans = 1;
27
       for (re ll l = 2, r; l <= n; l = r + 1) {
28
            r = min(n, n / (n / l));
29
            ans -= (r - l + 1) * GetSum_mu(n / l);
30
31
       return sum_mu[n] = ans;
32
   }
33
34
   inline ll GetSum_phi(ll n) {
35
       if (n <= 3e7) return phi[n];</pre>
36
       if (sum_phi[n]) return sum_phi[n];
37
       ll ans = n * (n + 1) / 2;
38
       for (re ll l = 2, r; l <= n; l = r + 1) {
39
            r = min(n, n / (n / 1));
40
            ans -= (r - l + 1) * GetSum_phi(n / l);
41
42
43
       return sum_phi[n] = ans;
44 }
         二次剩余
   4.8
   #include <iostream>
   #include <ctime>
3
4
   using namespace std;
5
   typedef long long ll;
6
7
8
9
   typedef struct{
       ll x, y; // 把求出来的w作为虚部, 则为a + bw
10
11
   }num;
12
13 ll quick_pow(ll a, ll b, ll p) {
14
       ll ans = 1;
15
       while(b) {
16
            if(b \& 1) ans = ans * a % p;
            a = a * a % p;
17
18
           b >>= 1;
19
20
       return ans % p;
21 }
22
23
   num num_mul(num a, num b, ll w, ll p) {// 复数乘法
24
       num ans = \{0, 0\};
25
       ans.x = (a.x * b.x \% p + a.y * b.y \% p * w \% p + p) \% p;
26
       ans.y = (a.x * b.y % p + a.y * b.x % p + p) % p;
27
       return ans;
28
29
   }
30
   ll num_pow(num a, ll b, ll w, ll p) { // 复数快速幂
31
       num ans = \{1, 0\};
32
       while(b) {
33
34
            if(b & 1)
```

```
ans = num_mul(ans, a, w, p);
35
36
           a = num_mul(a, a, w, p);
37
           b >>= 1;
38
39
       return ans.x % p;
   }
40
41
   ll legendre(ll a, ll p) { // 勒让德符号 = {1, -1, 0}
42
       return quick_pow(a, (p - 1) \gg 1, p);
43
   }
44
45
46
   ll Cipolla(ll n, ll p) {// 输入a和p, 是否存在x使得x^2 = a (mod p), 存在二次剩余返回x, 存在二次
       非剩余返回-1
                     注意: p是奇质数
47
       n \% = p;
       if(n == 0)
48
           return 0;
49
50
       if(p == 2)
           return 1;
51
52
       if(legendre(n, p) + 1 == p) // 二次非剩余
           return -1;
53
54
       ll a, w;
55
56
57
       while(true) {// 找出a, 求出w, 随机成功的概率是50%, 所以数学期望是2
58
           a = rand() \% p;
           w = ((a * a - n) % p + p) % p;
59
60
           if(legendre(w, p) + 1 == p) // 找到w, 非二次剩余条件
               break;
61
       }
62
       num x = \{a, 1\};
63
       return num_pow(x, (p + 1) >> 1, w, p) % p; // 计算x,一个解是x, 另一个解是p-x, 这里的w其实
64
       要开方,但是由拉格朗日定理可知虚部为0,所以最终答案就是对x的实部用快速幂求解
65
   }
66
   int main()
67
   {
68
69
       ll n, p;
       cin >> n >> p;
70
       srand((unsigned)time(NULL));
71
72
       cout << Cipolla(n, p) << endl;</pre>
73
       return 0;
74 }
   4.9
        反演相关
```

```
1 /*
2 莫比乌斯反演
3 g[n] = \sum_{d | n} f[d]
4 f[d] = \sum_{d | n} g[d] * mu[n / d]
5 二项式反演
6 g[n] = \sum{i = 1}^{n} C(n, i) * f[i]
7 f[n] = \sum{i = 1}^{n} C(n, i) * g[i] * (-1)^{n - i}
8 子集反演
9 f(S) = \sum_{T \belong S} g(T)
10 g(S) = \sum_{T \belong S} f(T) * (-1) ^ {|S| - |T|}
11 */
```

4.10 斐波那契

```
gcd(f[n], f[m]) = f[gcd(n, m)]
   斐波那契前n项和S[n] = f[n+2] - 1;
         光速幂
   4.11
1
   ll v_pow(ll a, ll b) {
2
       ll ans = 1;
3
       ll base = 65536, k = 1;
4
       while(1) {
5
            if((b \% (k * base)) / k == 0) break;
6
            ans = ans * quick_pow(a, (b % (k * base)) / k) % mod;
7
8
            a = quick_pow(a, base) % mod;
9
            k = k * base:
10
       }
11
       return ans;
12 }
          康托展开
   4.12
1 #include <bits/stdc++.h>//康拓展开
2 using namespace std;
3 typedef long long ll;
4 \quad const int N = 1e6 + 5;
5 const int Mod = 998244353;
  int T, n;
   11 a[N], fac[N], c[N];
   inline int lowbit(int x) { return x & -x; }
9
  void add(int x, int v) {
10
       for (int i = x; i <= n; i += lowbit(i)) c[i] += v;</pre>
11
12 int query(int x) {
13
       int res = 0;
       for (int i = x; i > 0; i \rightarrow lowbit(i)) res += c[i];
14
15
       return res;
16
   }
   void get_fac() {
17
       fac[1] = 1;
18
       for (int i = 2; i \le N; i++) fac[i] = fac[i - 1] * i \% Mod;
19
20
   11 cantor() {
21
       for (int i = n; i >= 1; i--) cin >> a[i];
22
       11 \text{ res} = 0;
23
       for (int i = 1; i <= n; i++) {
24
            res += 1ll*query(a[i]) * fac[i-1];
25
            res %= Mod;
26
27
            add(a[i], 1);
28
       return (res + 1) % Mod;
29
30 }
31
32
   int main() {
33
       get_fac();
34
       cin >> n;
```

```
cout << cantor() << endl;</pre>
35
   }
36
37 /*
38
39 ans = sum(a[i] * (i-1)!...) + 1
40 a[i] 代表第i个数比i到n的数之中大的个数
41
   */
42
43
44 //逆康托展开
45 #include <bits/stdc++.h>
46 using namespace std;
47 typedef long long ll;
48 const int N = 1e6 + 5;
49 const int Mod = 998244353;
50 int T, n, order;
51 ll fac[N];
52 vector<int> a, ans;
53 void get_fac() {
        fac[1] = fac[0] = 1;
54
        for (int i = 2; i \le N; i++) fac[i] = fac[i - 1] * i \% Mod;
55
56
   void decantor(int order, int n) {
57
        for (int i = 1; i <= n; i++) a.push_back(i);</pre>
59
        for (int i = n; i >= 1; i--) {
60
            int r = order % fac[i - 1];
61
            int t = order / fac[i - 1];
62
            order = r;
63
            ans.push_back(a[t]);
64
65
            a.erase(a.begin() + t);
        }
66
67
   }
   int main() {
68
        get_fac();
69
        cin >> order >> n;
70
71
        order--;
72
        for (auto x : ans) cout << x << " ";
73 }
   4.13
          快速幂
   ll mul(ll a, ll b) {
        ll z = (long double) a / mod * b;
        ll res = (unsigned long long) a * b - (unsigned long long) z * mod;
3
        return (res + mod) % mod;
 4
5
   }
   // 0(1) quick_mul, use long double
   inline ll quick_pow(ll ans, ll p, ll res = 1) {
        for(; p; p >>= 1, ans = mul(ans, ans) % mod)
8
            if(p \& 1) res = mul(res, ans) % mod;
9
        return res % mod;
10
11
   }
   double gcd(double a,double b) {
12
        if(fabs(b) < eps) return a;</pre>
13
        if(fabs(a) < eps) return b;</pre>
14
        return gcd(b, fmod(a,b));
15
16 }
```

```
int gcd(int a, int b) { return __gcd(a, b); }
18 ll gcd(ll a, ll b) { return __gcd(a, b); }
   4.14 莫比乌斯反演
   ll prime[N], mu[N], k;
2
  11 phi[N];
   bool is_prime[N];
3
   inline void init(int n) {
       memset(is_prime, true, sizeof is_prime);
5
       mu[1] = 1; phi[1] = 1;
6
       for (int i = 2; i < n; ++i) {
7
            if (is_prime[i]) prime[++k] = i, mu[i] = -1, phi[i] = i-1;
8
            for (int j = 1; j <= k && i * prime[j] < n; ++j) {
9
                is_prime[i * prime[j]] = false;
10
                if (i % prime[j] == 0) {
11
                    phi[i * prime[j]] = phi[i] * prime[j];
12
                    break:
13
                } else {
14
                    mu[i * prime[j]] = -mu[i];
15
                    phi[i * prime[j]] = phi[i] * (prime[j] - 1);
16
17
                }
            }
18
19
20
       for (int i = 1; i < n; ++i) mu[i] += mu[i-1], phi[i] += phi[i-1];
21
   }
22
   ll cal(int n, int m) {
23
24
       11 \text{ ans} = 0;
       n \neq d, m \neq d;
25
       int mx = min(n, m);
26
27
       for (ll l = 1, r; l \le mx; l = r + 1) {
            r = min(n / (n / 1), m / (m / 1));
28
           ans += 111 * (mu[r] - mu[l-1]) * (n / l) * (m / l);
29
30
31
       return ans;
32
  }
   4.15 逆元
   namespace INV {
2
       typedef long long 11;
3
4
       const int N = 2e5 + 10;
       const int mod = 1e9 + 7;
5
       int inv[N];
6
7
       11 x, y;
       ll gcd(il a, ll b) {
8
            return b ? gcd(b, a % b) : a;
9
10
       void get_inv(int n) {
11
12
            inv[0] = inv[1] = 1;
            for(int i = 2; i <= n; ++i)</pre>
13
                inv[i] = 1ll * (mod - mod / i) * inv[mod % i] % mod;
14
15
       il quick_pow(ll ans, ll p, ll res = 1) {
16
            ans %= mod;
17
```

```
p \% = mod - 1;
18
            for(; p; p >>= 1, ans = ans * ans % mod)
19
                if(p \& 1) res = res * ans % mod;
20
            return res % mod;
21
22
        ll inv1(ll ans) {
23
24
            return quick_pow(ans, mod - 2);
25
26
        ll ex_gcd(ll a, ll b, ll &x, ll &y) { //)[ ŷ¼,[ ] [ ] [ 烽
            if(!b) {
27
28
                x = 1, y = 0;
29
                return a;
30
            ll r = ex_gcd(b, a \% b, y, x);
31
            ll t = x;
32
33
            x = y;
            y = t - a / b * y;
34
35
            return r;
36
        ll inv2(ll a) { //0 0 a0 0 mod0 \mu0 0 0 0 £-0 0 0 ^2»´0 0 «\mu0 0 -1
37
            ll d = ex_gcd(a, mod, x, y);
38
39
            return d == 1 ? (x \% mod + mod) \% mod : -1;
        }
40
41 }
   4.16
          欧拉函数
int prime[N], phi[N], k;
   bool is_prime[N];
   void get_phi(int n) {
3
        memset(is_prime, true, sizeof is_prime);
4
        phi[1] = 1;
5
        for (int i = 2; i < n; i++) {
6
            if (is_prime[i]) prime[++k] = i, phi[i] = i - 1;
7
            for (int j = 1; j <= k && i * prime[j] < n; j++) {</pre>
8
                is_prime[i * prime[j]] = false;
9
                if (i % prime[j] == 0) {
10
11
                     phi[i * prime[j]] = phi[i] * prime[j];
                    break;
12
13
                phi[i * prime[j]] = phi[i] * (prime[j] - 1);
14
15
            }
16
        }
17
   }
18
19
   ll init(ll n) {
        ll m = (int) sqrt(n + 0.5);
20
        ll \ ans = n;
21
22
        for (ll i = 2; i <= m; ++ i) {
            if (n % i == 0) {
23
                ans = ans /i * (i - 1):
24
                while(n % i == 0) n /= i;
25
            }
26
27
28
        if (n > 1) ans = ans / n * (n - 1);
29
        return ans;
30 }
```

4.17 欧拉降幂

```
void init(int n) {
        memset(is_prime, true, sizeof is_prime);
2
        phi[1] = 1;
3
        for (int i = 2; i < n; i++) {
4
             if (is_prime[i]) prime[++k] = i, phi[i] = i - 1;
5
            for (int j = 1; j <= k && i * prime[j] < n; j++) {
    is_prime[i * prime[j]] = false;</pre>
6
7
                 if (i % prime[j] == 0) {
8
9
                     phi[i * prime[j]] = phi[i] * prime[j];
10
                     break:
11
                 phi[i * prime[j]] = phi[i] * (prime[j] - 1);
12
            }
13
        }
14
15
   ll mod(ll a, ll mm) {return a >= mm ? a % mm + mm : a;}
   ll ksm(ll a, ll b, ll mm) {
17
        ll res = 1, base = a;
18
        while (b) {
19
20
            if (b & 1) res = mod(res * base, mm);
21
            base = mod(base * base, mm);
22
23
24
        return res;
25
   il calc(ll a, ll b, ll mm) {
26
        if (b == 0 || mm == 1) return 1;
27
        else return ksm(a, calc(a, b-1, phi[mm]), mm);
28
29
30
   ll f(ll a, ll b, ll mm) {
        if (a == 0) return 0;
31
32
        ll ans = calc(a, b, mm) \% mm;
        return ans;
33
34 }
           筛法树
   4.18
1 ll fa_prime[N], fa_prime_edge[N];
  ll prime[N], cnt;
   bool is_prime[N];
   void prime_table(int n) {
        memset(is_prime, true, sizeof is_prime);
5
        cnt = 0;
6
        for (int i = 2; i < n; i++) {
7
             if (is_prime[i]) prime[++cnt] = i, fa_prime[i] = 1, fa_prime_edge[i] = i;
8
            for (int j = 1; j <= cnt && i * prime[j] < n; j++) {
   is_prime[i * prime[j]] = false;</pre>
9
10
                 fa_prime[i * prime[j]] = i;
11
                 fa_prime_edge[i * prime[j]] = prime[j];
12
                 if (i % prime[j] == 0) break;
13
            }
14
15
        }
16 }
```

4.19 算术基本定理

```
#include <bits/stdc++.h>
2
   using namespace std;
3
   typedef long long ll;
4
5
   ll get_Count(ll n) {
6
       ll\ ans = 1;
7
       for(int i = 2;i * i <= n; i++) {</pre>
8
9
            if(n \% i == 0) {
10
                int a = 0;
                while(n % i == 0) {
11
12
                    a++;
13
                    n \neq i;
                }
14
                ans *= (a + 1);
15
            }
16
17
18
       if(n > 1) ans *= 2;
19
       return ans;
20 }
21
22
   11 get_Sum(ll n) {
23
       ll \ ans = 1;
24
        for(int i = 2;i * i <= n; i++) {
25
            if(n \% i == 0) {
                11 a = 1;
26
                while(n \% i == 0) {
27
28
                    n \neq i;
                    a *= i;
29
30
                ans = ans * (a * i - 1) / (i - 1);
31
32
            }
33
34
       if(n > 1) ans *= (n + 1);
       return ans;
35
36 }
   4.20 质数、积性函数
1 /***0 0 0 /»0 0 0 -0 0 ***/
   //21474836470 0 0 0 0 0 0 0 0 0 £°105097565, 1e80 0 0 0 5e6 0 0 0 0
   void get_all(int n) {
3
4
       phi[1] = 1;
       for (int i = 2; i <= n; ++ i) {
5
6
            if (!vis[i]) prime[++ cnt] = i;
            for (int j = 1; j <= cnt \bar{k} prime[j] * i <= n; ++ j) {
7
                vis[i * prime[j]] = true;
8
   //
                p_num[i * prime[j]] = p_num[i] + p_num[prime[j]];
9
                if (i % prime[j] == 0) {
10
                    mu[i * prime[j] = 0;
11
                    phi[i * prime[j]] = phi[i] * prime[j];
12
                    break;
13
                } else {
14
                    mu[i * prime[j]] = -mu[i];
15
                    phi[i * prime[j]] = phi[i] * phi[prime[j]];
16
17
                }
           }
18
19
       }
```

```
20 }
21
   ll Euler(ll n) {
22
23
       ll res = n;
       for(int i = 2; i * i <= n; ++i) {</pre>
24
           if(n \% i == 0) res = res / i * (i - 1);
25
26
           while(n % i == 0) n /= i;
27
28
       if(n > 1) res = res / n * (n - 1);
29
       return res;
30 }
   4.21 线性筛
   const int N = 1e6 + 5;
3
   int prime[N];
   bool is_prime[N];
4
   void get_prime(){
6
       int k = 0;
       memset(is_prime, true, sizeof is_prime);
7
8
       is_prime[0] = is_prime[1] = false;
       for(int i = 2; i \le N; i++){
9
10
           if (is_prime[i]) prime[++k] = i;
           for(int j = 1; j <= k && i * prime[j] <= N; j++){
    is_prime[i * prime[j]] = false;</pre>
11
12
               if(i % prime[j] == 0) break;
13
           }
14
       }
15
   }
16
          ------//约数和
17
  int prime[N], cnt;
   bool is_prime[N];
20 ll sum[N], e[N];
21
   void init() {
22
       memset(is_prime, true, sizeof is_prime);
23
24
       sum[1] = 1;
       for (int i = 2; i < N; ++i) {
25
26
           if (is_prime[i]) {
27
               prime[++cnt] = i;
               sum[i] = i + 1;
28
               e[i] = 1;
29
30
           for (int j = 1; j <= cnt && 1ll*i * prime[j] < N; ++j) {</pre>
31
               is_prime[prime[j] * i] = false;
32
               if (i % prime[j] == 0) {
33
34
                   sum[i * prime[j]] = sum[i] * prime[j] + e[i];
                   e[i * prime[j]] = e[i];
35
                   break;
36
37
               sum[i * prime[j]] = sum[i] * (prime[j] + 1);
38
39
               e[i * prime[j]] = sum[i];
40
           }
41
       }
42 }
```

4.22 线性同余方程

```
/***D D D r D D 3D ***/
   void RemainderEquation(int a, int b, int n) {
        ll X, Y, d, res;
3
        11 min_res;
4
        d = gcd(a,n);
5
        exgcd(a, n, X, Y);
6
        if(b%d == 0) {
7
8
            X = X * (b / d) % n; //µõ½ 3½ 0 h½
            for(int i = 0; i < d; i++) {
9
                res = (X + (i * (b/d))) % n;
10
11
                cout << res << '\n';</pre>
                                        //0 0 0 0 0 0 0
12
            }
13
            \min_{res=(X%(n/d)+(n/d))%(n/d);}
            cout<<min_res<<endl;</pre>
                                         //O O O O C%O
14
        } else {
15
16
            cout << No Sulutions! << '\n';</pre>
17
18 }
   4.23
          原根
1
   #include <iostream>
3 #include <vector>
4
  using namespace std;
5
6
   typedef long long ll;
7
8
9 vector<ll> YG;
10
   ll p, n; // p是模数, n是p的欧拉函数值
11
12 ll gcd(ll a, ll b) {
        return b ? gcd(b, a % b) : a;
13
14 }
15
16
   ll quick_pow(ll a, ll b, ll p);
17
   ll phi(ll n) {
18
       11 \text{ ans} = n;
19
        for(int i = 2;i * i <= n; i++) {
20
            if(n % i == 0) {
21
                ans = ans - ans / i;
while(n % i == 0) {
22
23
24
                     n /= i;
25
                }
            }
26
27
        if(n > 1)
28
29
            ans = ans - ans / n;
30
        return ans;
31
   }
32
   vector<ll> PrimeFac(ll n) { // n的素因子
33
        vector<ll> fac;
34
35
        fac.clear();
```

```
for(ll i = 2;i * i <= n; i++) {</pre>
36
            if(n % i == 0) {
37
                fac.push_back(i);
38
                while(n % i == 0)
39
40
                    n \neq i;
            }
41
42
        if(n > 1)
43
            fac.push_back(n);
44
       return fac;
45
   }
46
47
   bool is_Protogen(ll p) { // 原根p = 2、4、p^k、2*p^k(p为非2的质数, k为任意数)
48
        if(p == 2 || p == 4) return true;
49
       if(p <= 1 || p % 4 == 0) return false;</pre>
50
       11 \text{ num} = 0;
51
       while(p % 2 == 0) // 2的倍数先筛掉
52
53
            p /= 2;
       for(int i = 3; i * i <= p; i++) { // p只能是一个非2的素数的倍数构成, 否则没有原根
54
            if(p \% i == 0) {
55
56
                num++;
                while(p % i == 0)
57
                    p /= i;
58
59
            }
60
       if(p > 1) num++;
61
       if(num == 1) return true;
62
       return false;
63
   }
64
65
   ll Protogen(ll p) {
66
       if(!is_Protogen(p)) // 先判断是否存在原根
67
            return -1;
68
       n = phi(p);
69
       if(p == 2) return 1;
70
       if(p == 3) return 2;
71
72
       if(p == 4) return 3;
73
       vector<ll> fac = PrimeFac(n); // f(p)的素因子
       for(int i = 2; i \le p - 1; i++) {
74
75
            if(gcd(i, p)!= 1) // n是模p的欧拉函数值, i要和n互质
                continue;
76
            bool flag = true;
77
            for(ll j = 0; j < fac.size(); j++) {</pre>
78
79
                if(quick_pow(i, n / fac[j] , p) == 1)
80
                    flaq = 0;
81
            if(flag) // i就是原根
82
83
                return i;
84
85
       return -1;
86
   }
87
   void Sum_Protogen(ll k) { // 找出n的所有原根
88
       YG.push_back(k);
89
       for(int i = 2;i < n; i++) {</pre>
90
            if(gcd(i, n) == 1) // i要与f(n)互质
91
92
                YG.push_back(quick_pow(k, i, p));
93
       }
94 }
```

```
95
     int main() {
96
         cin >> p;
97
         ll k = Protogen(p); // p的原根
98
99
         cout << k << endl;</pre>
         Sum_Protogen(k);
100
         for(int i = 0; i < YG.size(); i++) {</pre>
101
              cout << YG[i] << " ";
102
         }
103
         cout << endl;</pre>
104
105
         return 0;
106
    }
```

4.24 原根表

```
1
   mod
                                                   原根
2
   r*2^k+1
                               k
                                               g
                2
3
   3
        1
            1
   5
        1
            2
                2
4
   17
        1
            4
                3
5
            5
                5
   97
        3
6
   193 3
            6
                5
7
   257 1
            8
                3
8
                9
9
   7681
            15
                     17
   12289
            3
                12
                    11
10
   40961
            5
                13
                    3
11
   65537
            1
                16
                     3
12
   786433
            3
                18
                    10
13
14
   5767169 11
                19
                     3
                     3
15
   7340033 7
                20
   23068673
                11
                     21
                         3
16
17
   104857601
                25
                     22
                         3
   167772161
                5
                     25
                         3
18
                7
                         3
  469762049
                     26
19
                         3
  998244353
                119 23
20
                             这个数常用
                         3
                479 21
21
   1004535809
                             加起来不会爆int
                         31
   2013265921
                15
                     27
                         3
23
  2281701377
                17
                     27
                              这个数平方刚好不会爆11
24
   3221225473
                3
                     30
                         5
25
   75161927681 35
                     31
                         3
                         7
26
   77309411329 9
                     33
27
   206158430209
                     3
                         36
                             22
                             7
28
   2061584302081
                     15
                         37
                             3
29
   2748779069441
                     5
                         39
30
   6597069766657
                     3
                         41
                             5
                         42
                             5
31
   39582418599937
                    9
                         43
                             5
  79164837199873
                    9
32
                         44
  263882790666241 15
                             7
33
34
   1231453023109121
                         35
                             45
                                 3
                         19
                                 3
35
  1337006139375617
                             46
                                 5
36
  3799912185593857
                         27
                             47
  4222124650659841
                         15
                             48
                                 19
37
   7881299347898369
                         7
                             50
                                 6
38
   31525197391593473
                         7
                             52
                                 3
39
40
   180143985094819841
                         5
                             55
                                 6
   1945555039024054273
                         27
                             56
                                 5
   4179340454199820289 29
```

4.25 整除分块

```
int calc(int n, int m) {
1
2
       //sum_{i = 1} ^ {m} n / i
3
       //向下取整
4
       for (int l = 1, r; l <= m; l = r + 1) {
5
           if (n / 1) r = min(m, n / (n / 1));
           else r = m;
6
7
           //[l, r]之间的 n / l 都相等
       }
8
9
       //向上取整
10
       for (int l = 1, r; l <= m; l = r + 1) {
11
           int t = (n + l - 1) / l;
12
           if (t == 1) r = m;
13
           else r = min(m, (n - 1) / (t - 1));
14
           //[l, r]之间的 (n + l - 1) / l 都相等
15
       }
16
17
18 }
```

4.26 整数拆分

```
1 #include <bits/stdc++.h>
   using namespace std;
2
3
   typedef long long ll;
4
5
6
   //递归
   ll PartitionCount(ll n, ll m)
7
8
   {
9
        if(n == 1 | l m == 1)
10
            return 1;
        else if(n < m)</pre>
11
            return PartitionCount(n , n);
12
        else if(n == m)
13
            return PartitionCount(n , n - 1) + 1;
15
16
         return PartitionCount(n - m , m) + PartitionCount(n , m - 1);
17
   }
18
   //DP
19
20
   ll dp[10005][10005];
21
22
   void Partition_DP(ll n, ll m)
23
   {
        for(ll i = 1;i <= n + 1; i++)
24
25
            for(ll j = 1; j <= m + 1; j++)
26
27
28
                if(i == 1 || j == 1)
                     dp[i][j] = 1;
29
                else if(i == j)
30
                     dp[i][j] = 1 + dp[i][j - 1];
31
                else if(i < j)</pre>
32
33
                     dp[i][j] = dp[i][i];
34
                else
                     dp[i][j] = dp[i - j][j] + dp[i][j - 1];
35
```

```
36
             }
         }
37
38 }
   4.27
          整数分块
   void cal(ll a) {
2
        for (ll l = 1, r; l <= a; l = r + 1) {
3
            r = min(a, (a / (a / 1)));
            ans = (ans + (r - l + 1) \% MOD * (a / l));
4
        }
5
6
   }
          BSGS
   4.28
1 /***BSGS***/
   int A,B,C; // A^x = B \pmod{C}
   struct Hashmap \{//10 \ 0 \ \pm 0 \ 0 \ 0 \ map
        static const int Ha = 999917, maxe = 46340;
        int E,lnk[Ha],son[maxe+5],nxt[maxe+5],w[maxe+5];
5
6
        int top,stk[maxe+5];
        void clear() {E=0; while (top) lnk[stk[top--]]=0;}
7
        void Add(int x,int y) \{son[++E]=y;nxt[E]=lnk[x];w[E]=((1<<30)-1)*2+1;lnk[x]=E;\}
8
        bool count(int y) {
9
10
            int x=y%Ha;
            for (int j=lnk[x];j;j=nxt[j])
11
12
                if (y==son[j]) return true;
13
            return false;
14
        int& operator [] (int y) {
15
16
            int x=y%Ha;
            for (int j=lnk[x];j;j=nxt[j])
17
                if (y==son[j]) return w[j];
18
            Add(x,y);stk[++top]=x;return w[E];
19
        }
20
   } f;
21
   int exgcd(int a,int b,int &x,int &y) {
22
        if (!b) { x = 1; y = 0; return a; }
23
        int r = exgcd(b, a \% b, x, y), t = x; x = y; y = t - a / b * y;
24
25
        return r;
26
   int BSGS(int A,int B,int C) {
27
        if (C==1) if (!B) return A!=1; else return -1;
28
        if (B==1) if (A) return 0; else return -1;
29
        if (4\%C==0) if (!B) return 1; else return -1; //4 \square \square \square \square \square
30
        int m=ceil(sqrt(C)),D=1,Base=1;f.clear();
31
        for (int i=0;i<=m-1;i++) {//0 00 A^j 0 0 10 10 10 ±0
32
            f[Base]=min(f[Base],i);
33
            Base=((11)Base*A)%C;
34
35
        for (int i=0;i<=m-1;i++) {</pre>
36
37
            int x,y,r=exgcd(D,C,x,y);
38
            if (f.count(x)) return i*m+f[x]; // \square \square \% \square
39
            D=((ll)D*Base)%C;
40
41
        return -1;
42
```

```
43 }
   4.29
          Simpson 积分
1
   inline double f(double x) {
       return 0.0;
3
4
   double simpson(double l, double r) {
5
       double mid = (l + r) / 2;
6
       return (r - 1) * (f(1) + 4 * f(mid) + f(r)) / 6;
7
8
   double asr(double 1, double r, double ans, double eps) {
9
       double mid = (l + r) / 2;
10
       double fl = simpson(l, mid), fr = simpson(mid, r);
11
12
       if (abs(fl + fr - ans) \le 15 * eps)
           return fl + fr + (fl + fr - ans) / 15;
13
       return asr(1, mid, fl, eps * 0.5) + asr(mid, r, fr, eps * 0.5);
14
15 }
   4.30 2 维计算几何
1 const double pi = acos(-1.0);
3 const double eps = 1e-8;
4 const int maxp = 1010;
5 //`Compares a double to zero`
6
  int sqn(double x) {
       if(fabs(x) < eps)return 0;</pre>
7
8
       return (x > 0? 1: -1);
9 }
10 //sauare of a double
inline double sqr(double x) { return x * x; }
12 /*
   * Point
13
    * Point()
                             - Empty constructor
15
    * Point(double _x,double _y) - constructor
    * input()
                          - double input
16
17
    * output()
                          - %.2f output
18
    * operator ==
                          - compares x and y
    * operator <
                          - compares first by x, then by y
19
    * operator -
                          - return new Point after subtracting curresponging x and y
20
                          - cross product of 2d points
    * operator ^
21
    * operator *
22
                          - dot product
    * len()
                           - gives length from origin
23
                          - gives square of length from origin
    * len2()
24
    * distance(Point p) - gives distance from p
* operator + Point b - returns new Point after adding curresponging x and y
25
26
    * operator * double k - returns new Point after multiplieing x and y by k
27
    * operator / double k - returns new Point after divideing x and y by k
28
    * rad(Point a,Point b)- returns the angle of Point a and Point b from this Point
30
    * trunc(double r)
                         - return Point that if truncated the distance from center to r
    * rotleft()
                          - returns 90 degree ccw rotated point
31
    * rotright()
                           - returns 90 degree cw rotated point
32
    * rotate(Point p,double angle) - returns Point after rotateing the Point centering at
       p by angle radian ccw
```

```
35 struct Point {
36
        double x,y;
        Point() {}
37
        Point(double \_x, double \_y) { x = \_x; y = \_y; }
38
        void input() { cin >> x >> y; }
39
        bool operator == (Point b)const {
40
            return sgn(x - b.x) == 0 \&\& sgn(y - b.y) == 0;
41
42
        bool operator < (Point b)const {</pre>
43
            return sgn(x - b.x) == 0? sgn(y - b.y) < 0: x < b.x;
44
45
46
        Point operator -(const Point &b)const {
            return Point(x - b.x, y - b.y);
47
        }
48
        double operator ^(const Point &b)const {//200
49
            return x * b.y - y * b.x;
50
51
        double operator *(const Point &b)const {//µ□ □
52
            return x * b.x + y * b.y;
53
54
        double len() {//·µ»[ ¤¶[
55
            return hypot(x, y);//¿編 🛘 🖺
56
57
        double len2() { //\mu» \square  = 10 \ \square  = 10 \ \square
58
59
            return x * x + y * y;
60
        double distance(Point p) { //\cdot \mu \gg 0 }\mu 0 10 0
61
            return hypot(x - p.x, y - p.y);
62
63
        Point operator +(const Point &b)const {
64
             return Point(x + b.x, y + b.y);
65
66
        Point operator *(const double &k)const {
67
            return Point(x * k, y * k);
68
69
        Point operator /(const double &k)const {
70
71
            return Point(x / k, y / k);
72
        //`¼0 0 0 pa °0 pb ul+0 `
73
        double rad(Point a, Point b) {
74
            Point p = *this;
75
            return fabs(atan2( fabs((a-p)^(b-p)),(a-p)^*(b-p));
76
77
        //`»¯0 ³¤¶0 0 rµ0 0 0 –`
78
        Point trunc(double r) {
79
80
            double l = len();
            if(!sgn(l)) return *this;
81
            r \neq 1;
82
            return Point(x*r,y*r);
83
84
        //`0 0 <sup>6</sup>0 0 0 0 ¶900 0 `
85
        Point rotleft() { return Point(-y,x); }
86
        //`.<sup>6</sup>0 0 0 ¶900 0 `
87
        Point rotright() { return Point(y,-x); }
88
        //`0 \ 0 \ 0 \ p\mu 0 \ 0 \ 0 \ h 0 \ 0 \ 0 \ angle 0
89
        Point rotate(Point p,double angle) {
90
            Point v = (*this) - p;
91
            double c = cos(angle), s = sin(angle);
92
```

```
return Point(p.x + v.x * c - v.y * s,
93
                           p.y + v.x * s + v.y * c);
94
         }
95
96 };
    /*
97
     * Stores two points
98
     * Line()
                                         - Empty constructor
99
     * Line(Point _s,Point _e)
                                         - Line through _s and _e
100
                                         - checks if two points are same
     * operator ==
101
     * Line(Point p,double angle) - one end p , another end at angle degree * Line(double a,double b,double c) - Line of equation ax + by + c = 0
102
103
104
     * input()
                                        - inputs s and e
     * adjust()
                                         - orders in such a way that s < e
105
     * length()
106
                                         - distance of se
                                         - return 0 <= angle < pi
     * angle()
107
                                         - 3 if point is on line
     * relation(Point p)
108
                                           1 if point on the left of line
109
                                           2 if point on the right of line
110
       pointonseg(double p)
                                         - return true if point on segment
111
                                        - return true if they are parallel
       parallel(Line v)
112
       segcrossseg(Line v)
                                         - returns 0 if does not intersect
113
                                           returns 1 if non-standard intersection
114
                                           returns 2 if intersects
115
116
     * linecrossseg(Line v)
                                         - line and sea
117
     * linecrossline(Line v)
                                         - 0 if parallel
                                          1 if coincides
118
                                           2 if intersects
119
     * crosspoint(Line v)
                                         - returns intersection point
120
      * dispointtoline(Point p)
                                        - distance from point p to the line
121
      * dispointtoseg(Point p)
                                       - distance from p to the segment
122
     * disseqtoseg(Line v)
                                       - distance of two segment
123
     * lineprog(Point p)
124
                                        - returns projected point p on se line
       symmetrypoint(Point p)
                                     - returns reflection point of p over se
125
126
127
     */
    struct Line {
128
129
         Point s, e;
130
         Line() {}
         Line(Point _s,Point _e) { s = _s; e = _e; }
131
132
         bool operator ==(Line v) {
133
             return (s == v.s) && (e == v.e);
134
         //`.[]¾[] h.[] [] [] [] 6½[] anglej¶[] [] [] ,0 <= angle < pi`
135
         Line(Point p, double angle) {
136
137
             s = p;
             if(sgn(angle-pi/2) == 0) e = (s + Point(0,1));
138
139
             else e = (s + Point(1, tan(angle)));
140
         //ax + by + c = 0
141
142
         Line(double a, double b, double c) {
143
             if(sgn(a) == 0) {
144
                 s = Point(0, -c/b);
145
                 e = Point(1, -c/b);
             else\ if(sgn(b) == 0) {
146
                 s = Point(-c/a, 0);
147
                 e = Point(-c/a, 1);
148
149
             } else {
                 s = Point(0, -c/b);
150
                 e = Point(1,(-c-a)/b);
151
```

```
152
            }
153
        void input() { s.input(); e.input(); }
154
155
        void adjust() { if(e < s) swap(s, e); }</pre>
156
        //O O O O D ¤¶O
        double length() { return s.distance(e); }
157
        //`·µ»□ □ □ □ □ □ □ 6½□ 0 <= angle < pi
158
        double angle() {
159
            double k = atan2(e.y-s.y,e.x-s.x);
160
161
             if(sgn(k) < 0) k += pi;
             if(sqn(k-pi) == 0) k -= pi;
162
             return k;
163
164
        }
        165
        int relation(Point p) {
166
            int c = sgn((p-s)^{(e-s)});
167
            if(c < 0) return 1;
168
            else if(c > 0) return 2;
169
170
            else return 3;
171
        // µ0 0 0 0 0 0 0 0 0 0 0 0 0
172
        bool pointonseg(Point p) {
173
            return sgn((p - s) \wedge (e - s)) == 0 \&\&
174
                    sgn((p - s) * (p - e)) <= 0;
175
176
        //`}O O -O O O (¶O O O O O O O O O O )`
177
        bool parallel(Line v) {
178
            return sgn((e - s) \wedge (v.e - v.s)) == 0;
179
180
        181
        int segcrossseg(Line v) {
182
             int d1 = sgn((e - s) \wedge (v.s - s));
183
             int d2 = sgn((e - s) \wedge (v.e - s));
184
            int d3 = sgn((v.e - v.s) \wedge (s - v.s));
185
            int d4 = sgn((v.e - v.s) \wedge (e - v.s));
186
            if( (d1 \wedge d2) == -2 &\& (d3 \wedge d4) == -2 )return 2;
187
             return (d1 == 0 \& sqn((v.s - s) * (v.s - e)) <= 0) | |
188
                    (d2 == 0 \&\& sgn((v.e - s) * (v.e - e)) <= 0) | |
189
                    (d3 == 0 \&\& sgn((s - v.s) * (s - v.e)) <= 0) | |
190
                    (d4 == 0 \&\& sgn((e - v.s) * (e - v.e)) <= 0);
191
        }
192
        //`□ □ □ □ □ □ □ □ □ w□ `2 ¹淶□ □ `1 ·n淶□ □ `0 ²»□ □ `
193
        int linecrossseg(Line v) {
194
195
             int d1 = sgn((e - s) \wedge (v.s - s));
            int d2 = sgn((e - s) \wedge (v.e - s));
196
            if((d1 \wedge d2) == -2) return 2;
197
            return (d1 == 0 || d2 ==0 );
198
199
        //`}0 0 0 0 0 `0 0 0 0 `1 0 0 0 `2 0 0 `
200
201
        int linecrossline(Line v) {
202
             if((*this).parallel(v))
203
                 return v.relation(s) == 3;
204
            return 2;
205
        //`O O }O O L'»µO O `DZO O }O O O »O О лО О О `
206
207
        Point crosspoint(Line v) {
            double a1 = (v.e - v.s) \wedge (s - v.s);
208
209
            double a2 = (v.e - v.s) \wedge (e - v.s);
            return Point((s.x * a2 - e.x * a1) / (a2 - a1),
210
```

```
211
                         (s.y * a2 - e.y * a1) / (a2 - a1));
212
        //μ湍□ □ l'O □ □ □
213
        double dispointtoline(Point p) {
214
215
            return fabs((p - s) ^ (e - s)) / length();
216
217
        //u湍0 0 0 l'0 0 0
        double dispointtoseg(Point p) {
218
            if(sgn((p - s) * (e - s)) < 0 | | sgn((p - e) * (s - e)) < 0)
219
220
                return min(p.distance(s), p.distance(e));
221
            return dispointtoline(p);
222
        }
        //`·u»0 0 0 0 ½0 0 0 10 0 0 ` j0 0 0 0 10 0 0 0 0 0 £-0 0 ½0 0 0 0 0 0 0 0 0 0 0
223
224
        double dissegtoseg(Line v) {
225
            return min(min(dispointtoseg(v.s), dispointtoseg(v.e)),
226
                       min(v.dispointtoseg(s), v.dispointtoseg(e)));
227
        //`·µ»0 0 p0 0 0 0 0 0 0 0 ÿ`
228
229
        Point lineprog(Point p) {
            return s + (((e - s) * ((e - s) * (p - s))) / ((e - s).len2()));
230
231
        //`\mu»0 0 p<sup>1</sup>0 0 0 0 K0 \Xi0 0 `
232
233
        Point symmetrypoint(Point p) {
            Point q = lineprog(p);
234
            return Point(2 * q.x - p.x, 2 * q.y - p.y);
235
        }
236
237
    };
    //[
238
239
    struct circle {
        Point p;
240
241
        double r;
242
        circle() {}
        circle(Point _p,double _r) { p = _p; r = _r; }
243
        circle(double x,double y,double _r) {
244
245
            p = Point(x,y);
246
            r = _r;
        }
247
        //`0 0 ½0 0 0 0 0 0 0 °0 0 0 }0 0 0 0 д10 õ½0 0 0 0 `
248
        circle(Point a,Point b,Point c) {
249
            Line u = Line((a+b)/2,((a+b)/2)+((b-a).rotleft()));
250
            Line v = Line((b+c)/2,((b+c)/2)+((c-b).rotleft()));
251
252
            p = u.crosspoint(v);
253
            r = p.distance(a);
254
        }
        255
        0 0 0 0 ±0
        circle(Point a, Point b, Point c, bool t) {
256
257
            Line u, v;
258
            double m = atan2(b.y - a.y, b.x - a.x),
                   n = atan2(c.y - a.y, c.x - a.x);
259
260
            u.s = a;
            u.e = u.s + Point(cos((n + m) / 2), sin((n + m) / 2));
261
262
            v.s = b;
            m = atan2(a.y - b.y, a.x - b.x),
263
264
            n = atan2(c.y - b.y, c.x - b.x);
265
            v.e = v.s + Point(cos((n + m) / 2), sin((n + m) / 2));
266
            p = u.crosspoint(v);
267
            r = Line(a, b).dispointtoseg(p);
        }
268
```

```
269
        bool operator == (circle v) {
270
271
             return (p == v.p) && sgn(r - v.r) == 0;
272
273
        bool operator < (circle v)const {</pre>
             return ((p < v.p) | | ((p == v.p) && sgn(r - v.r) < 0));
274
275
        double area() { return pi * r * r; }
276
        double circumference() {//[ [ x
277
278
             return 2*pi*r;
279
        //`µ0 0 0 µĹ0 0 `0 0 0 0 `1 0 0 0 `2 0 0 0 `
280
        int relation(Point b) {
281
             double dst = b.distance(p);
282
283
             if(sgn(dst - r) < 0) return 2;</pre>
             else if(sqn(dst - r) == 0) return 1;
284
             return 0;
285
286
        //`O O O O O µĹO O `±ŧO O O O O ĵ‰O O O ľO O O □µĹO O `
287
288
         int relationseg(Line v) {
289
             double dst = v.dispointtoseg(p);
290
             if(sgn(dst - r) < 0) return 2;</pre>
291
             else if(sgn(dst - r) == 0) return 1;
292
             return 0;
293
        }
        //`O O O O O µĹO O `±tO O O O O ÔO O CO O O O □µĹO O `
294
        int relationline(Line v) {
295
             double dst = v.dispointtoline(p);
296
             if(sgn(dst - r) < 0) return 2;
297
             else if(sgn(dst - r) == 0) return 1;
298
299
             return 0;
300
        }
        //`}O uĹO O `5 O O O O `4 O O O O `3 O O `2 O O O O `1 O ¬O `
301
         int relationcircle(circle v) {
302
             double d = p.distance(v.p);
303
             if(sgn(d - r - v.r) > 0) return 5;
304
             if(sgn(d - r - v.r) == 0) return 4;
305
             double l = fabs(r - v.r);
306
             if(sgn(d - r - v.r) < 0 \& sgn(d - l) > 0) return 3;
307
308
             if(sgn(d - 1) == 0) return 2;
             if(sqn(d - 1) < 0) return 1;
309
310
        }
         //`□ □ } □ □ μĽ»μ徥μ»□ 0±□ 'û□ н»μ徥μ»□ 1□ □ n □ 徥2□ □ } □ □ `
311
        int pointcrosscircle(circle v, Point &p1, Point &p2) {
312
             int rel = relationcircle(v);
313
314
             if(rel == 1 || rel == 5)return 0;
             double d = p.distance(v.p);
315
             double l = (d*d+r*r-v.r*v.r)/(2*d);
316
             double h = sqrt(r*r-l*l);
317
318
             Point tmp = p + (v.p-p).trunc(1);
             p1 = tmp + ((v.p-p).rotleft().trunc(h));
319
320
             p2 = tmp + ((v.p-p).rotright().trunc(h));
             if(rel == 2 || rel == 4) return 1;
321
322
             return 2;
323
        //`0 0 0 0 0 0 μL'»μ循·μ»0 »μ0 0 0 0 `
324
        int pointcrossline(Line v,Point &p1,Point &p2) {
325
             if(!(*this).relationline(v)) return 0;
326
```

```
327
             Point a = v.lineprog(p);
             double d = v.dispointtoline(p);
328
             d = sqrt(r * r - d * d);
if(sgn(d) == 0) { p1 = p2 = a; return 1; }
329
330
             p1 = a + (v.e-v.s).trunc(d);
331
332
             p2 = a - (v.e-v.s).trunc(d);
             return 2;
333
334
         //`μõ½¹□ a,b}μ缉°□□ r1μ□ } □ □ `
335
         int gercircle(Point a,Point b,double r1,circle &c1,circle &c2) {
336
             circle x(a,r1),y(b,r1);
337
338
             int t = x.pointcrosscircle(y,c1.p,c2.p);
339
             if(!t) return 0;
             c1.r = c2.r = r1;
340
341
             return t;
         }
342
         //`μο̃½O O O O O UO O O У¬¹O μO q,°□O r1μO O
343
         int getcircle(Line u,Point q,double r1,circle &c1,circle &c2) {
344
345
             double dis = u.dispointtoline(q);
             if(sgn(dis - r1 * 2) > 0) return 0;
346
             if(sgn(dis) == 0) {
347
                 c1.p = q + ((u.e - u.s).rotleft().trunc(r1));
348
349
                 c2.p = q + ((u.e - u.s).rotright().trunc(r1));
                 c1.r = c2.r = r1;
350
351
                 return 2;
352
             Line u1 = Line((u.s + (u.e-u.s).rotleft().trunc(r1)),
353
                             (u.e + (u.e-u.s).rotleft().trunc(r1)));
354
             Line u2 = Line((u.s + (u.e-u.s).rotright().trunc(r1)),
355
356
                             (u.e + (u.e-u.s).rotright().trunc(r1)));
             circle cc = circle(q,r1);
357
358
             Point p1,p2;
             if(!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2);
359
             c1 = circle(p1,r1);
360
             if(p1 == p2) { c2 = c1; return 1; }
361
362
             c2 = circle(p2,r1);
363
             return 2;
364
         }
         //`rf0 0 0 0 0 u,v0 0 0 y¬°□0 r1µ0 0 `
365
         int getcircle(Line u,Line v,double r1,circle &c1,circle &c2,circle &c3,circle &c4)
366
             if(u.parallel(v))return 0;//}[ [ [ [ [ [
367
368
             Line u1 = Line(u.s + (u.e-u.s).rotleft().trunc(r1),
                             u.e + (u.e-u.s).rotleft().trunc(r1));
369
370
             Line u2 = Line(u.s + (u.e-u.s).rotright().trunc(r1),
371
                             u.e + (u.e-u.s).rotright().trunc(r1));
             Line v1 = Line(v.s + (v.e-v.s).rotleft().trunc(r1),
372
373
                             v.e + (v.e-v.s).rotleft().trunc(r1));
             Line v2 = Line(v.s + (v.e-v.s).rotright().trunc(r1),
374
                             v.e + (v.e-v.s).rotright().trunc(r1));
375
             c1.r = c2.r = c3.r = c4.r = r1;
376
377
             c1.p = u1.crosspoint(v1);
378
             c2.p = u1.crosspoint(v2);
379
             c3.p = u2.crosspoint(v1);
380
             c4.p = u2.crosspoint(v2);
381
             return 4;
382
         }
         //`^{h}0 \square0 0 0 cx,cy0 0 0 y_{\neg}^{\circ}\square0 r1\mu0 0 `
383
         int getcircle(circle cx,circle cy,double r1,circle &c1,circle &c2) {
384
```

```
385
            circle x(cx.p, r1 + cx.r), y(cy.p, r1 + cy.r);
386
             int t = x.pointcrosscircle(y, c1.p, c2.p);
             if(!t) return 0;
387
388
            c1.r = c2.r = r1;
            return t;
389
390
        }
391
        392
        int tangentline(Point q,Line &u,Line &v) {
393
             int x = relation(q);
394
             if(x == 2) return 0;
395
             if(x == 1) {
396
                u = Line(q, q + (q - p).rotleft());
397
398
                V = U;
                return 1;
399
            }
400
            double d = p.distance(q);
401
            double l = r * r / d;
402
            double h = sqrt(r * r - l * l);
403
            u = Line(q, p + ((q - p).trunc(l) + (q - p).rotleft().trunc(h)));
404
            v = Line(q, p + ((q - p).trunc(l) + (q - p).rotright().trunc(h)));
405
406
            return 2;
407
        //`0 0 }0 0 0 µ0 0 0 0 `
408
        double areacircle(circle v) {
409
             int rel = relationcircle(v);
410
             if(rel >= 4) return 0.0;
411
            if(rel <= 2) return min(area(), v.area());</pre>
412
            double d = p.distance(v.p);
413
            double hf = (r + v.r + d) / 2.0;
414
            double ss = 2 * sqrt(hf * (hf - r) * (hf - v.r) * (hf - d));
415
            double a1 = acos((r * r + d * d - v.r * v.r) / (2.0 * r * d));
416
417
            a1 = a1 * r * r;
            double a2 = acos((v.r * v.r + d * d - r * r) / (2.0 * v.r * d));
418
            a2 = a2 * v.r * v.r;
419
            return a1 + a2 - ss;
420
421
        //`0 0 }0 0 0 μ0 0 0 0 (¾«¶Φ0 0 )(0 0 Clong double)`
422
423
        double areacircle2(circle v) {
            double a = hypot(p.x-v.p.x,p.y-v.p.y),b=r,c=v.r;
424
            double s1 = pi * r* r, s2 = pi * v.r * v.r;
425
             if(sgn(a - b - c) >= 0) return 0;
426
            if(sgn(a + min(b,c) - max(b,c)) \le 0) return min(s1, s2);
427
428
            else {
                double cta1 = 2 * acos((a * a + b * b - c * c) / (2 * a * b));
429
                double cta2 = 2 * acos((a * a + c * c - b * b) / (2 * a * c));
430
                 return cta1 / (2 * pi) * s1 - 0.5 * sin(cta1) * b * b +
431
                        cta2 / (2 * pi) * s2 - 0.5 * sin(cta2) * c * c;
432
            }
433
434
        //`0 0 0 °0 0 0 ½0 0 0 pabµ0 0 0 0 0 `
435
436
        double areatriangle(Point a, Point b) {
            if(sgn((p - a) \land (p - b)) == 0) return 0.0;
437
            Point q[5];
438
            int len = 0;
439
            q[len ++] = a;
440
            Line l(a,b);
441
442
            Point p1,p2;
443
             if(pointcrossline(l, q[1], q[2]) == 2) {
```

```
if(sgn((a - q[1]) * (b - q[1])) < 0) q[len ++] = q[1];
444
                 if(sgn((a - q[2]) * (b - q[2])) < 0) q[len ++] = q[2];
445
446
             q[len ++] = b;
447
             if(len == 4 \& sgn((q[0] - q[1]) * (q[2] - q[1])) > 0) swap(q[1], q[2]);
448
             double res = 0;
449
             for(int i = 0; i < len - 1; ++ i) {</pre>
450
                 if(relation(q[i]) == 0 | | relation(q[i + 1]) == 0) {
451
                     double arg = p.rad(q[i], q[i + 1]);
452
                     res += r * r * arg / 2.0;
453
454
                 } else res += fabs((q[i] - p) \wedge (q[i + 1] - p)) / 2.0;
455
             }
456
             return res;
        }
457
    };
458
459
460
       n,p Line l for each side
       input(int _n)
                                               - inputs _n size polygon
461
       add(Point q)
                                               - adds a point at end of the list
462
                                               - populates line array
463
       getline()
                                               - comparision in convex_hull order
464
       cmp
     * norm()
                                              - sorting in convex_hull order
465
     * getconvex(polygon &convex)
                                              - returns convex hull in convex
466
467
     * Graham(polygon &convex)
                                              - returns convex hull in convex
468
     * isconvex()
                                              - checks if convex
     * relationpoint(Point q)
                                              - returns 3 if q is a vertex
469
                                                         2 if on a side
470
                                                         1 if inside
471
                                                         0 if outside
472
       convexcut(Line u,polygon &po)
                                               - left side of u in po
473
                                              - returns side length
474
       gercircumference()
475
       getarea()
                                              - returns area
476
       getdir()
                                              - returns 0 for cw, 1 for ccw
       getbarycentre()
                                              - returns barycenter
477
478
479
     */
480
    struct polygon {
481
        int n;
        Point p[maxp];
482
        Line l[maxp];
483
        void input(int _n) {
484
485
             n = _n;
             for(int i = 0; i < n; i++)
486
487
                 p[i].input();
488
489
        void add(Point q) {
490
            p[n ++] = q;
491
        void getline() {
492
493
             for(int i = 0; i < n; i++) {
494
                 l[i] = Line(p[i],p[(i+1)%n]);
495
             }
496
497
        struct cmp {
             Point p;
498
             cmp(const Point &p0) \{ p = p0; \}
499
500
             bool operator()(const Point &aa,const Point &bb) {
501
                 Point a = aa, b = bb;
                 int d = sgn((a-p)^{(b-p)});
502
```

```
if(d == 0) {
503
                                                  return sgn(a.distance(p)-b.distance(p)) < 0;</pre>
504
505
506
                                        return d > 0;
507
                             }
508
                    };
                    //`½0 0 м«½0 0 0 0 0 `
509
                    //`0 0 0 0 0 0 C0 0 ½0 0 0 0 0 ½ģĵ0 `
510
                    //`\[ \] \[ \Cappa \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \[ \] \
511
                    void norm() {
512
513
                              Point mi = p[0];
514
                              for(int i = 1; i < n; i++) mi = min(mi, p[i]);</pre>
                              sort(p, p + n, cmp(mi));
515
                    }
516
                    //`µõ½□ °□ `
517
                    //`μοχμο 0 °0 0 0 0 0 ĵ0 0 0 0 0$\sim$n-1μ0 `
518
                    //`}0 0 0 °0 µķ½''`
519
                    //`□ □ □ □ □ □ □ Ö □ □C□ □ □ □ □ □ □ □ e耧μ役»□ □ □ ²□ □ □ □ □ □ □ □ □ □ □ `
520
                    void getconvex(polygon &convex) {
521
522
                              sort(p,p+n);
523
                              convex.n = n;
                              for(int i = 0; i < min(n,2); i++) {</pre>
524
                                       convex.p[i] = p[i];
525
526
                              if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//\mathbb{I} \mathbb{I}
527
                              if(n <= 2)return;</pre>
528
529
                              int &top = convex.n;
                              top = 1;
530
531
                              for(int i = 2; i < n; i++) {
                                       while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)</pre>
532
533
                                                  top--:
534
                                       convex.p[++top] = p[i];
                              }
535
                              int temp = top;
536
                              convex.p[++top] = p[n-2];
537
                              for(int \bar{i} = n-3; i >= 0; i--) {
538
                                       while(top != temp && sgn((convex.p[top]-p[i])^(convex.p[top-1]-p[i])) <= 0)</pre>
539
                                                  top--;
540
                                       convex.p[++top] = p[i];
541
542
                              if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//\mathbb{I} \mathbb{I}
543
                              convex.norm()://`[ 4uő½u[ ] [ 。<sup>6</sup> ] [ î宿 ] [ ] [ ] [ ] <sup>6</sup> ] [ `
544
545
                    //`uõ½0 °0 u0 0 0 0 n0 0 ½·"`
546
                    void Graham(polygon &convex) {
547
                              norm();
548
                              int &top = convex.n;
549
550
                              top = 0;
551
                              if(n == 1) {
552
                                       top = 1;
                                       convex.p[0] = p[0];
553
554
                                       return;
555
                              if(n == 2) {
556
557
                                       top = 2;
                                       convex.p[0] = p[0];
558
                                       convex.p[1] = p[1];
559
                                       if(convex.p[0] == convex.p[1])top--;
560
                                        return;
561
```

```
}
562
            convex.p[0] = p[0];
563
            convex.p[1] = p[1];
564
565
            top = 2;
566
            for(int i = 2; i < n; i++) {
                while( top > 1 && sgn((convex.p[top-1]-convex.p[top-2])^(p[i]-convex.p[top
567
        -2])) <= 0)
                     top--:
568
                convex.p[top++] = p[i];
569
570
            if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n--;//\square \square
571
572
        }
        //`O жO O Dz»O O O μO `
573
        bool isconvex() {
574
            bool s[3];
575
            memset(s,false,sizeof(s));
576
            for(int i = 0; i < n; i++) {
577
                int j = (i+1)\%n;
578
579
                int k = (j+1)%n;
                s[sgn((p[j]-p[i])^(p[k]-p[i]))+1] = true;
580
                if(s[0] && s[2])return false;
581
            }
582
            return true;
583
        }
584
        585
        int relationpoint(Point q) {
586
            for(int i = 0; i < n; i++) {</pre>
587
                 if(p[i] == q)return 3;
588
            }
589
            getline();
590
            for(int i = 0; i < n; i++) {
591
                if(l[i].pointonseg(q))return 2;
592
593
594
            int cnt = 0;
            for(int i = 0; i < n; i++) {</pre>
595
                int j = (i+1)\%n;
596
                int k = sgn((q-p[j])^{p[i]-p[j]);
597
598
                int u = sgn(p[i].y-q.y);
599
                int v = sgn(p[j].y-q.y);
                if(k > 0 \&\& u < 0 \&\& v >= 0)cnt++;
600
                if(k < 0 \&\& v < 0 \&\& u >= 0)cnt--;
601
            }
602
            return cnt != 0;
603
        }
604
        //`0 0 0 u0 и0 0 ¶0 0 0 0 0 0 °
605
606
        //`0 0 0 0 0 0 1 1/0 0
        void convexcut(Line u,polygon &po) {
607
            608
            top = 0;
609
            for(int i = 0; i < n; i++) {</pre>
610
                 int d1 = sgn((u.e-u.s)^(p[i]-u.s));
611
612
                int d2 = sgn((u.e-u.s)^(p[(i+1)%n]-u.s));
                if(d1 >= 0)po.p[top++] = p[i];
613
                if(d1*d2 < 0)po.p[top++] = u.crosspoint(Line(p[i],p[(i+1)%n]));
614
            }
615
616
        //`µõ½🛮 🖟 ¤`
617
618
        double getcircumference() {
619
            double sum = 0;
```

```
for(int i = 0; i < n; i++) {
620
                 sum += p[i].distance(p[(i+1)%n]);
621
622
623
            return sum;
624
        //`µõ½0 0 0 `
625
626
        double getarea() {
627
            double sum = 0;
             for(int i = 0; i < n; i++) {</pre>
628
629
                 sum += (p[i]^p[(i+1)%n]);
630
631
            return fabs(sum)/2;
        }
632
        //`uõ½½0 0 `1 ±0 '0 0 <sup>6</sup>0 □0±0 '. <sup>6</sup>0 0 `
633
        bool getdir() {
634
635
            double sum = 0;
636
             for(int i = 0; i < n; i++)</pre>
                 sum += (p[i]^p[(i+1)%n]);
637
638
             if(sqn(sum) > 0)return 1;
639
            return 0;
        }
640
        //`uõ½0 0 0 °
641
        Point getbarycentre() {
642
            Point ret(0,0);
643
            double area = 0;
644
            for(int i = 1; i < n-1; i++) {
645
                 double tmp = (p[i]-p[0])^(p[i+1]-p[0]);
646
                 if(sqn(tmp) == 0)continue;
647
648
                 area += tmp;
649
                 ret.x += (p[0].x+p[i].x+p[i+1].x)/3*tmp;
                 ret.y += (p[0].y+p[i].y+p[i+1].y)/3*tmp;
650
651
            if(sgn(area)) ret = ret/area;
652
653
            return ret;
        }
654
        //`¶0 0 0 0 0 0 ½»µ0 0 0 0 `
655
        double areacircle(circle c) {
656
            double ans = 0;
657
            for(int i = 0; i < n; i++) {</pre>
658
659
                 int j = (i+1)\%n;
                 if(sgn((p[j]-c.p)^(p[i]-c.p)) >= 0)
660
                     ans += c.areatriangle(p[i],p[j]);
661
662
                 else ans -= c.areatriangle(p[i],p[j]);
663
664
            return fabs(ans);
665
        }
        //`¶0 0 0 0 0 0 ±0 0 `
666
        667
        int relationcircle(circle c) {
668
            getline();
669
            int x = 2;
670
             if(relationpoint(c.p) != 1)return 0;//0 0 U»0 0 2
671
             for(int i = 0; i < n; i++) {</pre>
672
                 if(c.relationseg(l[i])==2)return 0;
673
674
                 if(c.relationseg(l[i])==1)x = 1;
675
            }
676
            return x;
677
        }
678 };
```

```
679 //`AB X AC`
    double cross(Point A,Point B,Point C) {
680
        return (B-A)^(C-A);
681
682
    //`AB*AC`
683
    double dot(Point A,Point B,Point C) {
684
        return (B-A)*(C-A);
685
686 }
687 //`0 0 C¾0 0 0 0 0 0 2 0 `
688 //` A \pm0 0 0 0 0 0 ° 0 (¶0 0 0 0 0 0 0 0 0 0 0 )`
    double minRectangleCover(polygon A) {
689
        690
691
        if(A.n < 3)return 0.0;
692
        A.p[A.n] = A.p[0];
        double ans = -1;
693
694
        int r = 1, p = 1, q;
695
        for(int i = 0; i < A.n; i++) {</pre>
            696
697
            while (sqn(cross(A,p[i],A,p[i+1],A,p[r+1]) - cross(A,p[i],A,p[i+1],A,p[r]))
        >= 0 )
                r = (r+1)%A.n;
698
            //`;"30 A.p[i] - A.p[i+1]\frac{1}{2}0 0 0 0 0 0 0 n0 0 0 \mu_1^20 `
699
            while(sqn( dot(A.p[i],A.p[i+1],A.p[p+1]) - dot(A.p[i],A.p[i+1],A.p[p]) ) >= 0 )
700
701
                 p = (p+1)%A.n;
            if(i == 0)q = p;
702
            //`¿"³0 A.p[i] - A.p[i+1]½0 0 0 0 °0 0 0 0 uî0 `
703
704
            while(sqn(dot(A,p[i],A,p[i+1],A,p[q+1]) - dot(A,p[i],A,p[i+1],A,p[q])) <= 0)
705
                 q = (q+1)%A.n;
706
            double d = (A.p[i] - A.p[i+1]).len2();
707
            double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
708
                          (dot(A.p[i],A.p[i+1],A.p[p]) - dot(A.p[i],A.p[i+1],A.p[q]))/d;
709
            if(ans < 0 \mid l \mid ans > tmp)ans = tmp;
        }
710
        return ans;
711
712
    }
713 //`0 0 0 0 0 0 ¶0 0 0 0 `
714 //`¶D D D D D D D D ŬD D å¬D D a1a2uD D D D `
715
    vector<Point> convexCut(const vector<Point> &ps,Point q1,Point q2) {
        vector<Point>qs;
716
        int n = ps.size();
717
        for(int i = 0; i < n; i++) {
718
            Point p1 = ps[i], p2 = ps[(i+1)%n];
719
            int d1 = sgn((q2-q1)^{p1-q1}), d2 = sgn((q2-q1)^{p2-q1});
720
            if(d1 >= 0)
721
                qs.push_back(p1);
722
            if(d1 * d2 < 0)
723
                qs.push_back(Line(p1,p2).crosspoint(Line(q1,q2)));
724
        }
725
726
        return as:
727
    //`°□□□添`
728
    struct halfplane:public Line {
729
        double angle;
730
        halfplane() {}
731
        //`±0 '0 0 -s->e0 0 <sup>6</sup>0 0 (0 0 0 )µİ0 0 0 0 `
732
733
        halfplane(Point _s,Point _e) {
734
            s = _s;
735
            e = _e;
```

```
736
        halfplane(Line v) {
737
738
             s = v.s;
739
             e = v.e;
740
        void calcangle() {
741
742
            angle = atan2(e.y-s.y,e.x-s.x);
743
        bool operator <(const halfplane &b)const {</pre>
744
             return angle < b.angle;</pre>
745
746
747
    };
    struct halfplanes {
748
        int n;
749
        halfplane hp[maxp];
750
        Point p[maxp];
751
        int que[maxp];
752
753
        int st,ed;
        void push(halfplane tmp) {
754
            hp[n++] = tmp;
755
756
        //z0 0
757
        void unique() {
758
759
             int m = 1;
760
             for(int i = 1; i < n; i++) {</pre>
                 if(sgn(hp[i].angle-hp[i-1].angle) != 0)
761
762
                     hp[m++] = hp[i];
                 else if(sgn( (hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s) > 0)
763
                     hp[m-1] = hp[i];
764
             }
765
766
             n = m;
767
        bool halfplaneinsert() {
768
             for(int i = 0; i < n; i++)hp[i].calcangle();</pre>
769
             sort(hp,hp+n);
770
             unique();
771
772
             que[st=0] = 0;
773
             que\lceil ed=1 \rceil = 1;
            p[1] = hp[0].crosspoint(hp[1]);
774
             for(int i = 2; i < n; i++) {</pre>
775
                 while(st<ed && sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))<0)ed--;</pre>
776
                 while(st<ed && sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))<0)st++;</pre>
777
                 que[++ed] = i;
778
779
                 if(hp[i].parallel(hp[que[ed-1]]))return false;
                 p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
780
781
            while(st<ed && sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[que[st]].s))<0)ed--;</pre>
782
            783
784
             if(st+1>=ed)return false;
785
             return true;
786
        //`µõ½□ □ □ □ □ □ 添µõ½μ□ □ ¶□ □ □ □ `
787
        //`[ [ Ç[ [ ] [ ] halfplaneinsert() [ qµ»[ true`
788
        void getconvex(polygon &con) {
789
             p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
790
             con.n = ed-st+1;
791
792
             for(int j = st, i = 0; j \le ed; i++, j++)
                 con.p[i] = p[j];
793
```

```
}
794
             double minRectangleCover(polygon A) {
795
             796
             if(A.n < 3)return 0.0;
797
             A.p[A.n] = A.p[0];
798
             double ans = -1;
799
             int r = 1, p = 1, q;
800
             for(int i = 0; i < A.n; i++) {</pre>
801
                 //`¿¨³O O O A.p[i] - A.p[i+1]O O O μĵO `
802
                 while( sgn( cross(A.p[i],A.p[i+1],A.p[r+1]) - cross(A.p[i],A.p[i+1],A.p[r])
803
         \rangle = 0
804
                     r = (r+1)\%A.n;
                 //`;"30 A.p[i] - A.p[i+1]½0 0 0 0 0 0 0 n0 0 μĵ0 `
805
                 while(sgn( dot(A.p[i],A.p[i+1],A.p[p+1]) - dot(A.p[i],A.p[i+1],A.p[p]) ) >=
806
         0)
                     p = (p+1)%A.n;
807
                 if(i == 0)q = p;
808
                 //`¿"³0 A.p[i] - A.p[i+1]½0 0 0 0 0 0 0 0 μĵ0 `
809
                 while(sgn(dot(A.p[i],A.p[i+1],A.p[q+1]) - dot(A.p[i],A.p[i+1],A.p[q])) <=</pre>
810
        0)
                     q = (q+1)%A.n;
811
                 double d = (A.p[i] - A.p[i+1]).len2();
812
                 double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
813
                               (dot(A.p[i],A.p[i+1],A.p[p]) - dot(A.p[i],A.p[i+1],A.p[q]))/d;
814
815
                 if(ans < 0 | l | ans > tmp)ans = tmp;
             }
816
817
             return ans;
818
        circle minCircleCover(int n, Point p[], Point P = Point(0, 0)) {
819
             random_shuffle(p, p + n); double r2 = 0;
820
             for(int i = 0; i < n; ++ i) {
   if((p[i] - P).len2() > r2) {
821
822
                     P = p[i], r2 = 0;
823
                     for(int j = 0; j < i; ++ j) {
824
                         if((p[j]-P).len2() > r2) {
825
                              P = (p[i]+p[j])/2, r2 = (p[j]-P).len2();
826
                              for(int k = 0; k < j; ++ k) {
827
828
                                  if((p[k]-P).len2() > r2) {
829
                                      P = circle(p[i], p[j], p[k]).p, r2 = (p[k] - P).len2();
                                  }
830
                              }
831
                         }
832
                     }
833
834
                 }
835
836
             return circle(P, sqrt(r2));
837
        }
838 };
           3 维计算几何
    4.31
    struct Point3 {
 1
        double x, y, z;
 2
 3
        Point3(double xx = 0, double yy = 0, double zz = 0) { x = xx, y = yy, z = zz; }
        void input() { cin >> x >> y >> z; }
 4
        void output(void) { cout << fixed << setprecision(3) << x << ' ' << y << ' ' << z</pre>
 5
        << '\n'; }
        double len(void) { return sqrt(x * x + y * y + z * z); }
 6
```

```
double len2(void) { return x * x + y * y + z * z; }
  7
                       double dis(const Point3 &b) const { return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(z-b.y)*(y-b.y)+(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b.y)*(y-b
  8
                      b.z)*(z-b.z)); }
                       bool operator ==(const Point3 &b) const { return sqn(x-b.x)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)==0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y)=0\&sqn(y-b.y
  9
                      z-b.z)==0; }
                       bool operator <(const Point3 &b) const {</pre>
10
                                   if(sgn(x-b.x)!=0) return x < b.x;
11
                                   if(sgn(y-b.y)!=0) return y < b.y;</pre>
12
                                   return sgn(z-b.z) < 0;
13
14
                       Point3 operator + (const Point3 &b) const { return Point3(x+b.x,y+b.y,z+b.z); }
15
16
                       Point3 operator - (const Point3 &b) const { return Point3(x-b.x,y-b.y,z-b.z); }
                       Point3 operator * (const double &k) const { return Point3(x*k,y*k,z*k); }
17
                       Point3 operator / (const double &k) const { return Point3(x/k,y/k,z/k); }
18
                       Point3 operator ^ (const Point3 &b) const { return Point3(y*b.z-z*b.y,z*b.x-x*b.z,x
19
                      *b.y-y*b.x); }
                       double operator * (const Point3 &b) const { return x*b.x+y*b.y+z*b.z; }
20
21 };
          struct CH3D {
22
                       struct face {
23
                                   int a, b, c;//±0 '0 °0 h,0 0 0 0 0 0 0 0 0 10 0
24
25
                                   26
27
                       int n;//30 '¶¥µ0 0 0
                       Point3 P[maxn];
28
                       int num;//0 °0 ±0 0 0 0 0 0 ½0 0 0 0
29
                       face F[maxn<<3];//0 °0 ±0 0 0 0 0 ½0 0 0
30
                       int g[maxn][maxn];
31
                       Point3 cross(const Point3 &a,const Point3 &b,const Point3 &c) { return (b-a)^(c-a);
32
                       //0 0 ½0 0 0 0 0 0 *2
33
34
                       double area_triangle(Point3 a,Point3 b,Point3 c) { return ((b-a)^(c-a)).len(); }
                       //0 0 0 0 0 0 0 0 0 0 0 0 0 0 *6
35
                       double volume_four(Point3 a,Point3 b,Point3 c,Point3 d) { return ((b-a)^(c-a))*(d-a
36
                      ); }
                       //O O £°µO O O O O O O
37
                       double dblcmp(Point3 &p,face &f) {
38
39
                                   Point3 p1=P[f.b]-P[f.a];
                                   Point3 p2=P[f.c]-P[f.a];
40
                                   Point3 p3=p-P[f.a];
41
42
                                   return (p1^p2)*p3;
43
                       void deal(int p,int a,int b) {
44
                                   int f=q[a][b];
45
                                   face add;
46
                                   if(F[f].ok) {
47
                                               if(sgn(dblcmp(P[p],F[f]))>0)
48
                                                            dfs(p,f);
49
                                               else {
50
                                                            add.a=b; add.b=a; add.c=p;
51
52
                                                            add.ok=true;
53
                                                            g[p][b]=g[a][p]=g[b][a]=num;
54
                                                            F[num++]=add;
                                               }
55
                                   }
56
57
                       58
                       void dfs(int p,int now) {
59
                                   F[now].ok = false;
60
```

```
deal(p,F[now].b,F[now].a);
61
             deal(p,F[now].c,F[now].b);
62
             deal(p,F[now].a,F[now].c);
63
64
         bool same(int s,int t) {
65
             Point3 &a=P[F[s].a];
66
             Point3 &b=P[F[s].b];
67
             Point3 &c=P[F[s].c];
68
             int d1=sgn(volume_four(a,b,c,P[F[t].a]));
69
             int d2=sqn(volume_four(a,b,c,P[F[t].b]));
70
             int d3=sgn(volume_four(a,b,c,P[F[t].c]));
71
72
             return (d1==0)&&(d2==0)&&(d3==0);
73
         //11½"O O O O °O
74
         void create() {
75
             num=0;
76
 77
             face add;
             //±0 0 j0 κ0 依120 0
78
             //********
79
             bool flag=true;
80
             for(int i=1;i<n;i++) {</pre>
81
                  if(!(P[0]==P[i])) {
82
                      swap(P[1],P[i]);
83
                      flag=false;
84
85
                      break:
                  }
86
87
             if(flag) return ;
88
             flag=true;
89
             for(int i=2;i<n;i++) {</pre>
90
                  if(sgn(((P[1]-P[0])^(P[i]-P[0])).len())>0) {
91
                      swap(P[2],P[i]);
92
                      flag=false;
93
                      break;
94
                  }
95
96
97
             if(flag) return ;
98
             flag=true;
             for(int i=3;i<n;i++) {</pre>
99
                  if(sgn(((P[1]-P[0])^(P[2]-P[0]))*(P[i]-P[0]))!=0) {
100
                      swap(P[3],P[i]);
101
                      flag=false;
102
103
                      break;
                  }
104
105
             if(flag) return ;
106
107
             for(int i=0;i<4;i++) {</pre>
108
                  add.a=(i+1)\%4;
109
                  add.b=(i+2)\%4;
110
111
                  add.c=(i+3)\%4;
112
                  add.ok=true;
                  if(sgn(dblcmp(P[i],add))>0) swap(add.b,add.c);
113
                  g[add.a][add.b]=g[add.b][add.c]=g[add.c][add.a]=num;
114
                  F[num++]=add;
115
116
             for(int i=4;i<n;i++) {</pre>
117
118
                  for(int j=0;j<num;j++) {</pre>
                      if(F[j].ok&&sgn(dblcmp(P[i],F[j]))>0) {
119
```

```
120
                           dfs(i,j);
                           break;
121
                      }
122
                  }
123
124
             }
             int tmp=num;
125
126
             num=0;
             for(int i=0;i<tmp;i++)</pre>
127
                  if(F[i].ok)
128
                       F[num++]=F[i];
129
130
         }
         //±0 0 0
131
         double area(void) {
132
             double res=0;
133
             if(n==3) {
134
                  Point3 p=cross(P[0],P[1],P[2]);
135
                  return p.len()/2;
136
137
             for(int i=0;i<num;i++)</pre>
138
                  res+=area_triangle(P[F[i].a],P[F[i].b],P[F[i].c]);
139
             return res/2.0;
140
         }
141
         //0 0 0
142
143
         double volume(void) {
144
             double res=0;
             Point3 tmp=Point3(0,0,0);
145
             for(int i=0;i<num;i++)</pre>
146
                  res+=volume_four(tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
147
             return abs(res/6.0);
148
149
         //±0 0 0 0 0 ½0 0 0 0 0 0
150
         int sum_of_triangle(void) {
151
             return num;
152
153
         //±0 0 0 0 0 0 0 0 0 0
154
         int sum_of_polygon(void) {
155
156
             int res=0;
157
              for(int i=0;i<num;i++) {</pre>
                  bool flag=true;
158
                  for(int j=0;j<i;j++) {</pre>
159
160
                       if(same(i,j)) {
                           flag=false;
161
                           break;
162
163
                       }
                  }
164
165
                  res+=flag;
166
             }
             return res;
167
         }
168
         //0 0 0 0
169
170
         Point3 bary_center(void) {
171
             Point3 ans=Point3(0,0,0);
             Point3 o=Point3(0,0,0);
172
173
             double all=0;
             for(int i=0;i<num;i++) {</pre>
174
                  double vol=volume_four(o,P[F[i].a],P[F[i].b],P[F[i].c]);
175
176
                  ans=ans+(((o+P[F[i].a]+P[F[i].b]+P[F[i].c])/4.0)*vol);
177
                  all+=vol;
             }
178
```

```
179
             ans=ans/all;
180
             return ans;
181
         //µ溫0 0 l'0 0 0
182
         double dis_point_to_face(Point3 p,int i) {
183
             double tmp1=abs(volume_four(P[F[i].a],P[F[i].b],P[F[i].c],p));
184
             double tmp2=((P[F[i].b]-P[F[i].a])^(P[F[i].c]-P[F[i].a])).len();
185
             return tmp1/tmp2;
186
187
        }
188
    };
           三分套三分
    4.32
    int n; double maxx = -double(inf);
    double xmax = -10000000, xmin = 10000000, ymax = -10000000, ymin = 10000000;
 3
    double check(Point P) {
 4
 5
        double minn = double(inf);
        for(int i = 1; i <= n; ++ i) {
 6
 7
             minn = min(minn, P.rad(p[i], p[i + 1]));
 8
 9
        return minn;
    }
10
11
    double Find(double x) {
12
13
        double yl = ymin, yr = ymax, midl, midr;
14
        while(yr - yl > eps) {
15
             midl = (yl + yl + yr) / 3.0;
             midr = (yl + yr + yr) / 3.0;
16
             if(check(Point(x, midl)) < check(Point(x, midr))) yl = midl;</pre>
17
18
             else yr = midr;
19
20
        maxx = max(maxx, check(Point(x, yl)));
21
        return check(Point(x, yl));
22
    }
23
24
    inline void solve() {
        cin >> n; for(int i = 1; i <= n; ++ i) {
25
26
             p[i].input();;
27
             xmax = max(xmax, p[i].x);
28
             xmin = min(xmin, p[i].x);
29
            ymax = max(ymax, p[i].y);
             ymin = min(ymin, p[i].y);
30
        p[n + 1] = p[1];
31
32
        double xl = xmin, xr = xmax, midl, midr;
33
        while(xr - xl > eps) {
34
             midl = (xl + xl + xr) / 3.0;
35
             midr = (xl + xr + xr) / 3.0;
36
             if(Find(midl) < Find(midr)) xl = midl;</pre>
37
38
             else xr = midr;
39
40
        cout << fixed << setprecision(10) << maxx * 180.0 / pi << endl;</pre>
41
    }
```

4.33 三维几何

```
1 #include <math.h>
   #define eps 1e-8
3 #define zero(x) (((x)>0?(x):-(x))<eps)
4 struct point3{double x,y,z;};
5 struct line3{point3 a,b;};
6 struct plane3{point3 a,b,c;};
7 //计算 cross product U x V
   point3 Cross(point3 u,point3 v){
8
9
       point3 ret;
10
       ret.x=u.y*v.z-v.y*u.z;
       ret.y=u.z*v.x-u.x*v.z;
11
12
       ret.z=u.x*v.y-u.y*v.x;
13
       return ret;
   }
14
   //计算 dot product U . V
15
   double Dot(point3 u,point3 v){
17
       return u.x*v.x+u.y*v.y+u.z*v.z;
18 }
19 //矢量差 U - V
20 point3 subt(point3 u,point3 v){
21
       point3 ret;
22
       ret.x=u.x-v.x;
23
       ret.y=u.y-v.y;
24
       ret.z=u.z-v.z;
25
       return ret;
26 }
   //取平面法向量
27
   point3 pvec(plane3 s){
28
29
       return Cross(subt(s.a,s.b),subt(s.b,s.c));
30
   point3 pvec(point3 s1,point3 s2,point3 s3){
31
       return Cross(subt(s1,s2),subt(s2,s3));
32
33 }
   //两点距离,单参数取向量大小
34
   double distance(point3 p1,point3 p2){
35
       return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y)+(p1.z-p2.z)*(p1.z-p2.z)
36
       );
37 }
   //向量大小
38
   double vlen(point3 p){
39
       return sqrt(p.x*p.x+p.y*p.y+p.z*p.z);
40
41 }
   //判三点共线
42
  int dots_inline(point3 p1,point3 p2,point3 p3){
43
       return vlen(Cross(subt(p1,p2),subt(p2,p3)))<eps;</pre>
44
45 }
46 //判四点共面
47 int dots_onplane(point3 a,point3 b,point3 c,point3 d){
       return zero(Dot(pvec(a,b,c),subt(d,a)));
48
49 }
  //判点是否在线段上,包括端点和共线
   int dot_online_in(point3 p,line3 l){
52
       return zero(vlen(Cross(subt(p,1.a),subt(p,1.b))))&&(1.a.x-p.x)*(1.b.x-p.x)<eps&&</pre>
53
              (1.a.y-p.y)*(1.b.y-p.y) < eps&&(1.a.z-p.z)*(1.b.z-p.z) < eps;
54
   int dot_online_in(point3 p,point3 l1,point3 l2){
55
56
       return zero(vlen(Cross(subt(p,l1),subt(p,l2))))&(l1.x-p.x)*(l2.x-p.x)<eps&
57
              (l1.y-p.y)*(l2.y-p.y)<eps&&(l1.z-p.z)*(l2.z-p.z)<eps;
58
  }
```

```
//判点是否在线段上,不包括端点
    int dot_online_ex(point3 p,line3 l){
60
        return dot_online_in(p,l)&&(!zero(p.x-l.a.x)||!zero(p.y-l.a.y)||!zero(p.z-l.a.z))&&
61
62
                (!zero(p.x-l.b.x)||!zero(p.y-l.b.y)||!zero(p.z-l.b.z));
63
    int dot_online_ex(point3 p,point3 l1,point3 l2){
64
        return dot_online_in(p,l1,l2)&&(!zero(p.x-l1.x)||!zero(p.y-l1.y)||!zero(p.z-l1.z))
65
        &&
               (!zero(p.x-l2.x)||!zero(p.y-l2.y)||!zero(p.z-l2.z));
66
    }
67
    //判点是否在空间三角形上,包括边界,三点共线无意义
68
    int dot_inplane_in(point3 p,plane3 s){
        return zero(vlen(Cross(subt(s.a,s.b),subt(s.a,s.c)))-vlen(Cross(subt(p,s.a),subt(p,
70
        s.b)))-
                    vlen(Cross(subt(p,s.b),subt(p,s.c)))-vlen(Cross(subt(p,s.c),subt(p,s.a)
71
        )));
    }
72
    int dot_inplane_in(point3 p,point3 s1,point3 s2,point3 s3){
73
        return zero(vlen(Cross(subt(s1,s2),subt(s1,s3)))-vlen(Cross(subt(p,s1),subt(p,s2)))
74
                    vlen(Cross(subt(p,s2),subt(p,s3)))-vlen(Cross(subt(p,s3),subt(p,s1))));
75
    }
76
    //判点是否在空间三角形上,不包括边界,三点共线无意义
77
    int dot_inplane_ex(point3 p,plane3 s){
78
79
        return dot_inplane_in(p,s)&&vlen(Cross(subt(p,s.a),subt(p,s.b)))>eps&&
               vlen(Cross(subt(p,s.b),subt(p,s.c)))>eps&&vlen(Cross(subt(p,s.c),subt(p,s.a)
80
        ))>eps;
81
    int dot_inplane_ex(point3 p,point3 s1,point3 s2,point3 s3){
82
        return dot_inplane_in(p,s1,s2,s3)&&vlen(Cross(subt(p,s1),subt(p,s2)))>eps&&
83
               vlen(Cross(subt(p,s2),subt(p,s3)))>eps&&vlen(Cross(subt(p,s3),subt(p,s1)))>
84
        eps;
85
    }
    //判两点在线段同侧,点在线段上返回 0,不共面无意义
86
    int same_side(point3 p1,point3 p2,line3 l){
87
        return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))>eps
88
89
    }
    int same_side(point3 p1,point3 p2,point3 l1,point3 l2){
90
        return Dot(Cross(subt(l1,l2),subt(p1,l2)),Cross(subt(l1,l2),subt(p2,l2)))>eps;
91
    }
92
    //判两点在线段异侧,点在线段上返回 0,不共面无意义
93
    int opposite_side(point3 p1,point3 p2,line3 l){
94
        return Dot(Cross(subt(l.a,l.b),subt(p1,l.b)),Cross(subt(l.a,l.b),subt(p2,l.b)))<-</pre>
95
96
97
    int opposite_side(point3 p1,point3 p2,point3 l1,point3 l2){
        return Dot(Cross(subt(l1,l2),subt(p1,l2)),Cross(subt(l1,l2),subt(p2,l2)))<-eps;</pre>
98
99
   //判两点在平面同侧,点在平面上返回 0
100
101
    int same_side(point3 p1,point3 p2,plane3 s){
102
        return Dot(pvec(s),subt(p1,s.a))*Dot(pvec(s),subt(p2,s.a))>eps;
103
    int same_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3){
104
        return Dot(pvec(s1,s2,s3),subt(p1,s1))*Dot(pvec(s1,s2,s3),subt(p2,s1))>eps;
105
106
107
    //判两点在平面异侧,点在平面上返回 0
108
    int opposite_side(point3 p1,point3 p2,plane3 s){
        return Dot(pvec(s), subt(p1, s.a))*Dot(pvec(s), subt(p2, s.a))<-eps;</pre>
109
```

```
110
    }
    int opposite_side(point3 p1,point3 p2,point3 s1,point3 s2,point3 s3){
111
        return Dot(pvec(s1,s2,s3),subt(p1,s1))*Dot(pvec(s1,s2,s3),subt(p2,s1))<-eps;</pre>
112
113
114
    //判两直线平行
    int parallel(line3 u,line3 v){
115
        return vlen(Cross(subt(u.a,u.b),subt(v.a,v.b)))<eps;</pre>
116
117
    int parallel(point3 u1,point3 u2,point3 v1,point3 v2){
118
119
        return vlen(Cross(subt(u1,u2),subt(v1,v2)))<eps;</pre>
120 }
121
    //判两平面平行
    int parallel(plane3 u,plane3 v){
122
        return vlen(Cross(pvec(u),pvec(v)))<eps;</pre>
123
124
    int parallel(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
125
126
        return vlen(Cross(pvec(u1,u2,u3),pvec(v1,v2,v3)))<eps;</pre>
127
    }
128
    //判直线与平面平行
    int parallel(line3 l,plane3 s){
129
        return zero(Dot(subt(l.a,l.b),pvec(s)));
130
131
    int parallel(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
132
        return zero(Dot(subt(l1,l2),pvec(s1,s2,s3)));
133
134
    }
135
    //判两直线垂直
    int perpendicular(line3 u,line3 v){
136
137
        return zero(Dot(subt(u.a,u.b),subt(v.a,v.b)));
138
    int perpendicular(point3 u1,point3 u2,point3 v1,point3 v2){
139
        return zero(Dot(subt(u1,u2),subt(v1,v2)));
140
141
    //判两平面垂直
142
   int perpendicular(plane3 u,plane3 v){
143
        return zero(Dot(pvec(u),pvec(v)));
144
145
    int perpendicular(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
146
147
        return zero(Dot(pvec(u1,u2,u3),pvec(v1,v2,v3)));
148
    }
    //判直线与平面平行
149
    int perpendicular(line3 l,plane3 s){
150
        return vlen(Cross(subt(l.a,l.b),pvec(s)))<eps;</pre>
151
152
153
    int perpendicular(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
        return vlen(Cross(subt(l1,l2),pvec(s1,s2,s3)))<eps;</pre>
154
155
    //判两线段相交,包括端点和部分重合
156
    int intersect_in(line3 u,line3 v){
157
        if (!dots_onplane(u.a,u.b,v.a,v.b))
158
            return 0;
159
160
        if (!dots_inline(u.a,u.b,v.a)||!dots_inline(u.a,u.b,v.b))
161
             return !same_side(u.a,u.b,v)&!same_side(v.a,v.b,u);
162
        return dot_online_in(u.a,v)||dot_online_in(u.b,v)||dot_online_in(v.a,u)||
        dot_online_in(v.b,u);
163
164
    int intersect_in(point3 u1,point3 u2,point3 v1,point3 v2){
        if (!dots_onplane(u1,u2,v1,v2))
165
166
             return 0;
167
        if (!dots_inline(u1,u2,v1)||!dots_inline(u1,u2,v2))
```

```
168
            return !same_side(u1,u2,v1,v2)&&!same_side(v1,v2,u1,u2);
169
        return
                dot_online_in(u1,v1,v2)||dot_online_in(u2,v1,v2)||dot_online_in(v1,u1,u2)||
170
        dot_online_in(v2,u1,u2);
171
172
    //判两线段相交,不包括端点和部分重合
    int intersect_ex(line3 u,line3 v){
173
        return dots_onplane(u.a,u.b,v.a,v.b)&&opposite_side(u.a,u.b,v)&&opposite_side(v.a,v
174
        .b,u);
    }
175
176
    int intersect_ex(point3 u1,point3 u2,point3 v1,point3 v2){
177
        return dots_onplane(u1,u2,v1,v2)&&opposite_side(u1,u2,v1,v2)&&opposite_side(v1,v2,
        u1,u2);
178
      }
    //判线段与空间三角形相交,包括交于边界和(部分)包含
179
    int intersect_in(line3 l,plane3 s){
180
        return !same_side(l.a,l.b,s)&&!same_side(s.a,s.b,l.a,l.b,s.c)&&
181
               !same_side(s.b,s.c,l.a,l.b,s.a)&&!same_side(s.c,s.a,l.a,l.b,s.b);
182
183
    int intersect_in(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
184
        return !same_side(l1,l2,s1,s2,s3)&&!same_side(s1,s2,l1,l2,s3)&&
185
               !same_side(s2,s3,l1,l2,s1)&&!same_side(s3,s1,l1,l2,s2);
186
187
188
    //判线段与空间三角形相交,不包括交干边界和(部分)包含
189
    int intersect_ex(line3 l,plane3 s){
        return opposite_side(l.a,l.b,s)&&opposite_side(s.a,s.b,l.a,l.b,s.c)&&
190
191
               opposite_side(s.b,s.c,l.a,l.b,s.a)&&opposite_side(s.c,s.a,l.a,l.b,s.b);
192
    int intersect_ex(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
193
        return opposite_side(l1,l2,s1,s2,s3)&&opposite_side(s1,s2,l1,l2,s3)&&
194
195
               opposite_side(s2,s3,l1,l2,s1)&&opposite_side(s3,s1,l1,l2,s2);
196
197
    //计算两直线交点,注意事先判断直线是否共面和平行!
    //线段交点请另外判线段相交(同时还是要判断是否平行!)
198
    point3 intersection(line3 u,line3 v){
199
200
        point3 ret=u.a;
201
        double t=((u.a.x-v.a.x)*(v.a.y-v.b.y)-(u.a.y-v.a.y)*(v.a.x-v.b.x))
202
                 /((u.a.x-u.b.x)*(v.a.y-v.b.y)-(u.a.y-u.b.y)*(v.a.x-v.b.x));
        ret.x+=(u.b.x-u.a.x)*t;
203
204
        ret.y+=(u.b.y-u.a.y)*t;
205
        ret.z+=(u.b.z-u.a.z)*t;
        return ret;
206
207
208
    point3 intersection(point3 u1,point3 u2,point3 v1,point3 v2){
        point3 ret=u1;
209
210
        double t=((u1.x-v1.x)*(v1.y-v2.y)-(u1.y-v1.y)*(v1.x-v2.x))
211
                 /((u1.x-u2.x)*(v1.y-v2.y)-(u1.y-u2.y)*(v1.x-v2.x));
        ret.x+=(u2.x-u1.x)*t;
212
213
        ret.y+=(u2.y-u1.y)*t;
214
        ret.z+=(u2.z-u1.z)*t;
215
        return ret;
216
    //计算直线与平面交点,注意事先判断是否平行,并保证三点不共线!
217
218
    //线段和空间三角形交点请另外判断
219
    point3 intersection(line3 l,plane3 s){
220
        point3 ret=pvec(s);
221
        double t=(ret.x*(s.a.x-l.a.x)+ret.y*(s.a.y-l.a.y)+ret.z*(s.a.z-l.a.z))/
222
                 (ret.x*(l.b.x-l.a.x)+ret.y*(l.b.y-l.a.y)+ret.z*(l.b.z-l.a.z));
        ret.x=l.a.x+(l.b.x-l.a.x)*t;
223
```

```
224
        ret.y=l.a.y+(l.b.y-l.a.y)*t;
225
        ret.z=l.a.z+(l.b.z-l.a.z)*t;
226
        return ret;
227
    }
    point3 intersection(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
228
        point3 ret=pvec(s1,s2,s3);
229
        double t=(ret.x*(s1.x-l1.x)+ret.y*(s1.y-l1.y)+ret.z*(s1.z-l1.z))/
230
                  (ret.x*(l2.x-l1.x)+ret.y*(l2.y-l1.y)+ret.z*(l2.z-l1.z));
231
        ret.x=11.x+(12.x-11.x)*t;
232
        ret.y=l1.y+(l2.y-l1.y)*t;
233
234
        ret.z=l1.z+(l2.z-l1.z)*t;
235
        return ret;
236
    }
237
    //计算两平面交线,注意事先判断是否平行,并保证三点不共线!
    line3 intersection(plane3 u,plane3 v){
238
        line3 ret;
239
        ret.a=parallel(v.a,v.b,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(
240
        v.a,v.b,u.a,u.b,u.
241
                c);
        ret.b=parallel(v.c,v.a,u.a,u.b,u.c)?intersection(v.b,v.c,u.a,u.b,u.c):intersection(
242
        v.c,v.a,u.a,u.b,u.
243
                c);
        return ret;
244
245
    }
246
    line3 intersection(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
247
        ret.a=parallel(v1,v2,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v1,v2,u1,
248
        u2,u3);
        ret.b=parallel(v3,v1,u1,u2,u3)?intersection(v2,v3,u1,u2,u3):intersection(v3,v1,u1,
249
        u2,u3);
250
        return ret;
251
252
    //点到直线距离
    double ptoline(point3 p,line3 l){
253
        return vlen(Cross(subt(p,1.a),subt(l.b,l.a)))/distance(l.a,l.b);
254
255
256
    double ptoline(point3 p,point3 l1,point3 l2){
257
        return vlen(Cross(subt(p,l1),subt(l2,l1)))/distance(l1,l2);
258
    }
    //点到平面距离
259
260
    double ptoplane(point3 p,plane3 s){
        return fabs(Dot(pvec(s),subt(p,s.a)))/vlen(pvec(s));
261
262
263
    double ptoplane(point3 p,point3 s1,point3 s2,point3 s3){
        return fabs(Dot(pvec(s1,s2,s3),subt(p,s1)))/vlen(pvec(s1,s2,s3));
264
265
    }
    //直线到直线距离
266
    double linetoline(line3 u,line3 v){
267
        point3 n=Cross(subt(u.a,u.b),subt(v.a,v.b));
268
269
        return fabs(Dot(subt(u.a,v.a),n))/vlen(n);
270
    }
271
    double linetoline(point3 u1,point3 u2,point3 v1,point3 v2){
272
        point3 n=Cross(subt(u1,u2),subt(v1,v2));
273
        return fabs(Dot(subt(u1,v1),n))/vlen(n);
274
    //两直线夹角 cos 值
275
    double angle_cos(line3 u,line3 v){
276
277
        return Dot(subt(u.a,u.b),subt(v.a,v.b))/vlen(subt(u.a,u.b))/vlen(subt(v.a,v.b));
278 }
```

```
double angle_cos(point3 u1,point3 u2,point3 v1,point3 v2){
280
                 return Dot(subt(u1,u2),subt(v1,v2))/vlen(subt(u1,u2))/vlen(subt(v1,v2));
281
282
        //两平面夹角 cos 值
        double angle_cos(plane3 u,plane3 v){
283
284
                 return Dot(pvec(u),pvec(v))/vlen(pvec(u))/vlen(pvec(v));
285
        double angle_cos(point3 u1,point3 u2,point3 u3,point3 v1,point3 v2,point3 v3){
286
                 return Dot(pvec(u1,u2,u3),pvec(v1,v2,v3))/vlen(pvec(u1,u2,u3))/vlen(pvec(v1,v2,v3))
287
288
        }
289
        //直线平面夹角 sin 值
        double angle_sin(line3 l,plane3 s){
290
                 return Dot(subt(l.a,l.b),pvec(s))/vlen(subt(l.a,l.b))/vlen(pvec(s));
291
292
        double angle_sin(point3 l1,point3 l2,point3 s1,point3 s2,point3 s3){
293
                 return Dot(subt(l1,l2),pvec(s1,s2,s3))/vlen(subt(l1,l2))/vlen(pvec(s1,s2,s3));
294
295
        }
296
297
       // 球体相交
        double vol_ints(double x1, double y1, double z1, double r1, double x2, double y2,
298
                double z2, double r2) {
                 double sum = 4.00 / 3.00 * PI * r1 * r1 * r1 + 4.00 / 3.00 * PI * r2 * r2 * r2;
299
                 double ans = 0;
300
301
                 double dis = sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2) + (z1 - z2) * (z1 - z2) 
                  z2)); //球心距离
                 if (dis >= r1 + r2) //没有交到的地方
302
303
304
                         ans = 0;
                 } else if (dis + r1 <= r2)//重合
305
306
                         ans = (4.00 / 3.00) * PI * r1 * r1 * r1;
307
308
                 } else if (dis + r2 <= r1) {
                         ans = (4.00 / 3.00) * PI * r2 * r2 * r2;
309
                 } else //相交
310
311
                         double cal = (r1 * r1 + dis * dis - r2 * r2) / (2.00 * dis * r1);
312
                         double h = r1 * (1 - cal);
313
                         ans += (1.00 / 3.00) * PI * (3.00 * r1 - h) * h * h;
314
                         cal = (r2 * r2 + dis * dis - r1 * r1) / (2.00 * dis * r2);
315
                         h = r2 * (1.00 - cal);
316
                         ans += (1.00 / 3.00) * PI * (3.00 * r2 - h) * h * h;
317
318
319
                 return ans;
320
        }
         4.34
                      多项式 \ln_e x p_n o w
   1 #include <bits/stdc++.h>
   2 using namespace std;
   3 typedef long long ll;
   4 const double PI = acos(-1);
   5 const int N = 1e5 + 10;
   6
        struct Complex {
   7
   8
                 double x, y;
                 Complex(double a = 0, double b = 0): x(a), y(b) {}
   9
                 Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
 10
```

```
Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
11
                 Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
12
                  rhs.y + y * rhs.x); }
                 Complex conj() { return Complex(x, -y); }
13
       } w[N];
14
15
      ll mod, inv2;
16
       int tr[N];
17
     ll F[N], G[N];
18
19
     ll quick_pow(ll a, ll b);
21
22 int getLen(int n);
23
24 void FFT(Complex *A, int len);
25
     inline void MTT(ll *x, ll *y, ll *z, int len);
26
27
28 void Get_Inv(ll *f, ll *g, int n);
29
       void Get_Der(ll *f, ll *g, int len) { for(int i = 1; i < len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len; i++) g[i - 1] = f[i] * i % len
30
                  mod; g[len - 1] = 0; }
31
       void Get_Int(ll *f, ll *g, int len) { for(int i = 1;i < len; i++) g[i] = f[i - 1] *</pre>
32
                quick_pow(i, mod - 2) % mod; g[0] = 0; }
33
      void Get_Ln(ll *f, ll *g, int n);
34
35
       void Get_Exp(ll *f, ll *g, int n);
36
37
       void Get_Pow(ll *f, ll *g, int n, ll k);
38
39
       void Get_Sqrt(ll *f, ll *g, int n) {
40
                 static ll a[N];
41
42
                 Get_Ln(f, a, n);
                 for(int i = 0; i < n; i++) a[i] = a[i] * inv2 % mod;
43
                 Get_Exp(a, g, n);
44
45
                 int len = getLen(n);
                 for(int i = n; i < len; i++) g[i] = 0;
46
                 for(int i = 0; i < len; i++) a[i] = 0;
47
       }
48
                       二次剩余处理边界不为 1
        4.35
 1
 2 #include <bits/stdc++.h>
 3 using namespace std;
      typedef long long ll;
      const double PI = acos(-1);
 6
       const int N = 1e5 + 10;
 7
 8
 9
       struct Complex {
10
                 double x, y;
                 Complex(double a = 0, double b = 0): x(a), y(b) {}
11
                 Complex operator + (const Complex &rhs) { return Complex(x + rhs.x, y + rhs.y); }
12
                 Complex operator - (const Complex &rhs) { return Complex(x - rhs.x, y - rhs.y); }
13
```

```
Complex operator * (const Complex &rhs) { return Complex(x * rhs.x - y * rhs.y, x *
14
        rhs.y + y * rhs.x); }
       Complex conj() { return Complex(x, -y); }
15
16 } w[N];
17
  ll mod, inv2;
18
   int tr[N];
19
20 ll F[N], G[N];
21
22 ll quick_pow(ll a, ll b);
23
24 typedef struct{
25
       ll x, y; // 把求出来的w作为虚部,则为a + bw
26
27
   num num_mul(num a, num b, ll w, ll p) {// 复数乘法
28
       num ans = \{0, 0\};
29
       ans.x = (a.x * b.x % p + a.y * b.y % p * w % p + p) % p;
30
       ans.y = (a.x * b.y % p + a.y * b.x % p + p) % p;
31
32
       return ans;
33 }
34
   ll num_pow(num a, ll b, ll w, ll p) { // 复数快速幂
35
36
       num ans = \{1, 0\};
37
       while(b) {
           if(b & 1)
38
39
               ans = num_mul(ans, a, w, p);
           a = num_mul(a, a, w, p);
40
           b >>= 1;
41
42
43
       return ans.x % p;
   }
44
45
   ll legendre(ll a, ll p) { // 勒让德符号 = {1, -1, 0}
46
       return quick_pow(a, (p - 1) >> 1);
47
48
49
50
   ll Cipolla(ll n, ll p) {// 输入a和p, 是否存在x使得x^2 = a (mod p), 存在二次剩余返回x, 存在二次
       非剩余返回-1
                      注意: p是奇质数
51
       n \% = p;
52
       if(n == 0)
           return 0;
53
54
       if(p == 2)
           return 1;
55
56
       ll a, w;
57
       while(true) {// 找出a, 求出w, 随机成功的概率是50%, 所以数学期望是2
58
           a = rand() \% p;
59
           W = ((a * a - n) \% p + p) \% p;
60
61
           if(legendre(w, p) + 1 == p) // 找到w, 非二次剩余条件
62
               break:
63
       }
64
       num x = \{a, 1\};
65
       return num_pow(x, (p + 1) >> 1, w, p) % p; // 计算x, 一个解是x, 另一个解是p-x, 这里的w其实
       要开方,但是由拉格朗日定理可知虚部为0,所以最终答案就是对x的实部用快速幂求解
66
   }
67
68
   int getLen(int n);
69
```

```
70 void FFT(Complex *A, int len);
71
   inline void MTT(ll *x, ll *y, ll *z, int len);
72
73
   void Get_Inv(ll *f, ll *g, int n);
74
75
   void Get_Sqrt(ll *f, ll *g, int n) {
76
       if(n == 1) { ll t = Cipolla(f[0], mod); g[0] = min(mod - t, t); return ; }
77
       Get_Sqrt(f, g, (n + 1) >> 1);
78
79
80
       int len = getLen(n);
81
       static ll c[N], invg[N];
       for(int i = 0; i < len; i++) c[i] = i < n ? f[i] : 0;
82
       Get_Inv(g, invg, n);
83
       MTT(c, invg, c, len);
84
       for(int i = 0; i < n; i++) g[i] = inv2 * (c[i] + g[i]) % mod;
85
        for(int i = n; i < len; i++) g[i] = 0;
86
       for(int i = 0; i < len; i++) c[i] = invg[i] = 0;
87
   }
88
89
   int main() {
90
       inv2 = quick_pow(2, mod - 2);
91
92
       int n;
93
       cin >> n;
94
       for(int i = 0; i < n; i++) cin >> F[i];
       Get_Sqrt(F, G, n);
95
       for(int i = 0;i < n; i++) cout << G[i] << " ";</pre>
96
97 }
   4.36 x 不连续、暴力插值
1
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 const double PI = acos(-1);
  const int N = 3e5 + 10;
7
8
   ll mod;
9
   11 X[N], Y[N];
10
   ll quick_pow(ll a, ll b);
11
12
   11 Lagrange(ll *x, ll *y, int n, int k) {
13
       11 \text{ ans} = 0;
14
        for(int i = 0; i < n; i++) {
15
            11 s1 = 1, s2 = 1;
16
            for(int j = 0; j < n; j++) {
17
18
                if(i == j) continue;
                s1 = s1 * (k - x[j] + mod) % mod;
19
                s2 = s2 * (x[i] - x[j] + mod) % mod;
20
21
22
           ans = (ans + 1ll * y[i] * s1 % mod * quick_pow(s2, mod - 2) % mod) % mod;
23
24
       return ans;
25
   }
26
27 int main() {
```

```
int n, k;
28
29
        cin >> n >> k;
        for(int i = 0;i < n; i++) cin >> X[i] >> Y[i];
30
        cout << Lagrange(X, Y, n, k) << endl;</pre>
31
32 }
   4.37 x 连续、前缀优化
1
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 const int N = 1e5 + 10;
6
7 11 mod;
8 11 F[N];
9 11 pre[N], suf[N];
10 ll fac[N], invf[N];
11
12
13 ll quick_pow(ll a, ll b);
14
   void init() {
15
16
        fac[0] = 1;
        for(int i = 1; i < N; i++) fac[i] = fac[i - 1] * i % mod;
17
        invf[N - 1] = quick_pow(fac[N - 1], mod - 2);
18
        for(int i = N - 1; i >= 1; i --) invf[i - 1] = invf[i] * i % mod;
19
   }
20
21
   ll Lagrange(ll *f, int k, int n) {
        if(k <= n) return f[k];</pre>
23
24
        pre[0] = suf[n] = 1;
        for(int i = 1; i <= n; i++) pre[i] = pre[i - 1] * (k - i + 1) % mod;
25
        for(int i = n; i >= 1; i--) suf[i - 1] = suf[i] * (k - i) % mod;
26
        11 ans = 0;
27
        for(int i = 0;i <= n; i++) {</pre>
28
            int opt = (n - i) & 1 ? -1 : 1;
ans = (ans + 1ll * opt * pre[i] % mod * suf[i] % mod * invf[i] % mod * invf[n -
29
30
        i] % mod * f[i] % mod + mod) % mod;
31
        return f[k] = ans;
32
33 }
34
   int main() {
35
36
        init();
37
        int n, k;
38
        cin >> n >> k;
        for(int i = 0; i <= n; i++) cin >> F[i];
39
        cout << Lagrange(F, k, n) << endl;</pre>
40
41
42 }
   4.38 FFT 加速带有通配符字符串匹配
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
```

```
// p[x] = \sum_{i=0}^{m-1} A[i]^3 * B[x-m+i+1] + \sum_{i=0}^{m-1} A[i] * B[x-m+i+1]^3 -
         2 * \sum_{i=0}^{m-1} A[i]^2 * B[x-m+i+1]^2
6
   const int N = 1e6 + 1e5;
7
8
   ll qpow(ll a, ll b, ll mod) {
9
10
        ll\ ans = 1;
        while(b) {
11
            if(b \& 1) ans = ans * a % mod;
12
13
            a = a * a % mod;
14
            b >>= 1;
15
16
        return ans % mod;
   }
17
18
19 const ll G = 3;
20 const ll invG = qpow(G, mod - 2, mod);
21 int tr[N];
22
23
   void NTT(ll *A, int len, int type) {
24
        for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
25
        for (int i = 2; i <= len; i <<= 1) {
            int mid = i / 2;
27
            ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
            for (int k = 0; k < len; k += i) {
28
                ll w = 1;
29
                for (int l = k; l < k + mid; l++) {
30
                    ll t = w * A[l + mid] % mod;
31
                    A[l + mid] = (A[l] - t + mod) \% mod;
32
                    A[1] = (A[1] + t) \% mod;
33
34
                    w = w * Wn \% mod;
35
                }
            }
36
37
        if (type == -1) {
38
39
            ll invn = qpow(len, mod - 2, mod);
40
            for (int i = 0; i < len; i++)
                A[i] = A[i] * invn % mod;
41
        }
42
   }
43
44
   void mul(ll *a, ll *b, int n) {
45
        int len = 1; while (len <= n) len <<= 1;</pre>
46
        for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
47
48
        NTT(a, len, 1), NTT(b, len, 1);
49
        for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
50
        NTT(a, len, -1);
   }
51
53
   ll a1[N], a2[N], a3[N], b1[N], b2[N], b3[N];
54
55
   void solve() {
        int m, n; cin >> m >> n;
56
        string s, t; cin >> t >> s;
57
        for(int i = 0;i < m; i++) {</pre>
58
            if(t[i] == '*') continue
59
            int temp = t[i] - 'a' + 1;
60
            a1[i] = temp;
61
```

62

a2[i] = temp * temp;

```
a3[i] = temp * temp * temp;
63
64
        for(int i = 0; i < n; i++) {
65
            if(s[i] == '*') continue;
66
            int temp = s[i] - 'a' + 1;
67
           b1[i] = temp;
68
           b2[i] = temp * temp;
69
           b3[i] = temp * temp * temp;
70
       }
71
       reverse(a1, a1 + m);
72
73
       reverse(a2, a2 + m);
74
       reverse(a3, a3 + m);
       mul(a1, b3, n + m);
75
       mul(a2, b2, n + m);
76
       mul(a3, b1, n + m);
77
       vector<int> ans;
78
       for(int x = m - 1; x < n; x++) {
79
            ll res = a1[x] + a3[x] - a2[x] * 2;
80
            if(!res) ans.push_back(x - m + 2);
81
82
       cout << ans.size() << endl;</pre>
83
        for(int i = 0; i < ans.size(); i++) cout << ans[i] << (i == ans.size() - 1 ? endl :
84
85 }
   4.39 FFT 加速带有通配符字符串匹配 2
1
   void solve() {
2
       int o; cin >> o; while(o --) {
3
            int n, m; cin >> n >> m;
4
           vector<int> ans(n + m), res(n + m);
            string s, t; cin >> s >> t;
5
            reverse(t.begin(), t.end());
6
7
            Polynomial A(s.size() + 10), B(t.size() + 10), C;
8
9
            for(int cch = 0; cch <= 9; ++ cch) {
                char now = cch <= 9? char(cch + '0'): '*';</pre>
10
                for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] =
11
       0;
12
                for (int i = 0; i < s.size(); ++i) A[i] = s[i] == now;
                for (int i = 0; i < t.size(); ++i) B[i] = t[i] == now;
13
                C = A * B;
14
                for(int i = m - 1; i < n; ++ i) ans[i - m + 1] += C[i];
15
16
17
18 //
              for(int i = 0; i \le n; ++ i) A[i] = 0; for(int i = 0; i \le m; ++ i) B[i] = 0;
            for(int i = 0; i < s.size(); ++ i) A[i] = s[i] == '*';
19
20
            for(int i = 0; i < t.size(); ++ i) B[i] = 1;
            C = A * B;
21
            for(int i = m - 1; i < n; ++ i) ans[i - m + 1] += C[i];
22
23
24
25 //
              for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] = 0;
            for(int i = 0; i < s.size(); ++ i) A[i] = 1;</pre>
26
27
            for(int i = 0; i < t.size(); ++ i) B[i] = t[i] == '*';
28
            C = A * B;
            for(int i = m - 1; i < n; ++ i) ans[i - m + 1] += C[i];
29
```

```
30
       //
                                  for(int i = 0; i <= n; ++ i) A[i] = 0; for(int i = 0; i <= m; ++ i) B[i] = 0;
31
                             for(int i = 0; i < s.size(); ++ i) A[i] = s[i] == '*';
32
                             for(int i = 0; i < t.size(); ++ i) B[i] = t[i] == '*';
33
34
                             C = A * B;
                             for(int i = m - 1; i < n; ++ i) ans[i - m + 1] -= C[i];
35
36
                             for(int i = 0; i <= n - m; ++ i) res[m - ans[i]] ++;
37
                             for(int i = 1; i <= m; ++ i) res[i] += res[i - 1];</pre>
38
                             for(int i = 0; i <= m; ++ i) cout << res[i] << '\n';</pre>
39
40
41
                   }
42 }
                      FFT 加速朴素字符串匹配
        4.40
        #include <bits/stdc++.h>
        using namespace std;
 3
       const int N = 4e5 + 10;
 4
        // P[x] = \sum_{i=0}^{m-1} A[i] + \sum_{i=0}^{m-1} B[x - m + i + 1] - 2 * \sum_{i=0}^{m-1} B[x - m + i + 1] - 2 * \sum_{i=0}^{m-1} A[i] + \sum_{i=0}
                  -1A[i] * B[x - m + i + 1]
 6
       // reverse(a)
 8
 9
       // 当串中的字符集较少时,可以针对每个字符进行FFT,计算每个字符对整个串的贡献
10
11 ll qpow(ll a, ll b, ll mod);
12
13 const ll mod = 998244353;
14 const ll G = 3;
15 const ll invG = qpow(G, mod - 2, mod);
16 int tr[N];
17
        void NTT(ll *A, int len, int type) {
18
                   for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
19
20
                   for (int i = 2; i <= len; i <<= 1) {
21
                             int mid = i / 2;
22
                             ll Wn = qpow(type == 1 ? G : invG, (mod - 1) / i, mod);
23
                             for (int k = 0; k < len; k += i) {
                                       11 w = 1;
24
                                       for (int l = k; l < k + mid; l++) {
25
                                                 ll t = w * A[l + mid] % mod;
26
                                                 A[l + mid] = (A[l] - t + mod) \% mod;
27
                                                 A[1] = (A[1] + t) \% mod;
28
                                                 w = w * Wn % mod;
29
30
                                       }
                             }
31
32
                   if (type == -1) {
33
                             ll invn = qpow(len, mod - 2, mod);
34
                             for (int i = 0; i < len; i++)
35
                                       A[i] = A[i] * invn % mod;
36
37
                   }
38
        }
39
        void mul(ll *a, ll *b, int n) {
```

int len = 1; while (len <= n) len <<= 1;

41

```
for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
42
       NTT(a, len, 1), NTT(b, len, 1);
43
       for (int i = 0; i < len; i++) a[i] = a[i] * b[i] % mod;
44
45
       NTT(a, len, -1);
   }
46
47
  ll a[N], b[N];
48
49
   void solve() {
50
       string s, t; cin >> s >> t;
51
       int n = s.length(), m = t.length();
52
53
       for(int i = 0; i < n; i++) a[i] = s[i] - 'a' + 1;
       for(int i = 0; i < m; i++) b[i] = t[i] - 'a' + 1;
54
       reverse(b, b + m);
55
       mul(a, b, n + m - 2);
56
       double P = 0;
57
       for(int i = 0;i < m; i++) {
   P += (t[i] - 'a' + 1) * (t[i] - 'a' + 1);
58
59
60
       vector<int> f(n + 1);
61
       for(int i = 1;i < n; i++) {</pre>
62
            f[i] = f[i - 1] + (s[i] - 'a' + 1) * (s[i] - 'a' + 1);
63
64
        for(int x = m - 1; x < n; x++) {
65
66
            double res;
            if(x == m - 1) res = P + f[x] - a[x] * 2;
67
            else res = P + f[x] - f[x - m] - a[x] * 2;
68
            if(!res) cout << x - m + 2 << endl;
69
       }
70
71 }
   4.41 CDQ_FFT
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define gcd(a,b) __gcd(a,b)
4 #define lcm(a,b) (111 * a * b / gcd(a, b))
5 #define Polynomial vector<int>
6 #define Inv(x) quick_pow(x, mod - 2)
7 #define DEBUG(x, y) cout << x << ": " << y << '\n';</pre>
8 using ld = long double;
9 using ll = long long;
10 using ull = unsigned long long;
11 const ll mod = 100003;
12 const ld pi = acos(-1.0);
13 struct Complex {
       ld r, i;
14
15
       Complex(ld_r = 0, ld_i = 0) : r(_r), i(_i) {}
16 };
17
   Complex operator + (const Complex &a, const Complex &b) {
18
       return Complex(a.r + b.r, a.i + b.i);
19
20
21
   Complex operator - (const Complex &a, const Complex &b) {
       return Complex(a.r - b.r, a.i - b.i);
22
23
   Complex operator * (const Complex &a, const Complex &b) {
24
       return Complex(a.r * b.r - a.i * b.i, a.r * b.i + a.i * b.r);
25
```

```
26
   }
   Complex operator / (const Complex &a, const Complex &b) {
    return Complex((a.r * b.r + a.i * b.i) / (b.r * b.r + b.i * b.i), (a.i * b.r - a.r
27
28
        * b.i) / (b.r * b.r + b.i * b.i));
29
   }
30
   int R[int(1e6 + 10)];
31
   Complex x[int(1e6 + 10)], y[int(1e6 + 10)];
33
   void get_R(int lim) {
34
35
        for (int i = 0; i < lim; i++) {
36
            R[i] = (i \& 1) * (lim >> 1) + (R[i >> 1] >> 1);
37
   }
38
39
   void FFT(Complex *f, int lim, int rev) {
40
        for (int i = 0; i < lim; i++) {</pre>
41
42
            if (i < R[i]) swap(f[i], f[R[i]]);</pre>
43
        for (int mid = 1; mid < lim; mid <<= 1) {</pre>
44
            Complex wn = Complex(cos(pi / mid), rev * sin(pi / mid));
45
            for (int len = mid << 1, cur = 0; cur < lim; cur += len) {
46
                 Complex w = Complex(1, 0);
47
                 for (int k = 0; k < mid; k++, w = w * wn) {
48
49
                     Complex x = f[cur + k], y = w * f[cur + mid + k];
                     f[cur + k] = x + y, f[cur + mid + k] = x - y;
50
                 }
51
            }
52
53
        if (rev == -1) {
54
            for (int i = 0; i < lim; i++) {
55
                 f[i].r /= lim;
56
57
        }
58
   }
59
60
   int s[int(1e6 + 10)];
   vector<int> ans[int(1e6 + 10)];
62
63
   int quick_pow(int ans, int p, int res = 1) {
64
        for(; p; p >>= 1, ans = 1ll * ans * ans % mod)
65
            if(p & 1) res = 1ll * res * ans % mod;
66
        return res % mod;
67
68
   int fac[int(1e6 + 10)] = \{1\}, ifac[int(1e6 + 10)] = \{1\};
69
70
   void init(int n) {
71
        for(int i = 1; i <= n; ++ i)
72
            fac[i] = 111 * fac[i - 1] * i % mod,
                     ifac[i] = Inv(fac[i]);
73
74
75
   11 C(int n, int m) {
        if(n < 0 || m < 0 || m > n) return 0;
76
77
        if(m == 0 \mid l \mid m == n)
                                return 1;
        return 1ll * fac[n] * ifac[m] % mod * ifac[n - m] % mod;
78
79
80
   ll F(ll a, ll n, ll k) {
        return (111 * ans[1][k] - C(n, k) + mod) % mod * Inv(a - 1) % mod;
81
82
83 void CDQ_FFT(int rt, int l, int r) {
```

```
if (l == r) {
84
             ans[rt].push_back(1);
85
             ans[rt].push_back(s[l]);
86
87
             return ;
88
        int mid = l + r \gg 1;
89
        CDQ_FFT(rt << 1, l, mid);
90
        CDQ_FFT(rt \ll 1 \mid 1, mid + 1, r);
91
        int len1 = mid - l + 1, len2 = r - mid;
92
        for (int i = 0; i \le len1; i++) x[i] = Complex(ans[rt << 1][i], 0);
93
        for (int i = 0; i \le len2; i++) y[i] = Complex(ans[rt << 1 | 1][i], 0);
94
95
        int \lim = 1; while (\lim <= r - l + 1) \lim <<= 1;
96
        get_R(lim);
        FFT(x, lim, 1);
97
        FFT(y, lim, 1);
98
        for (int i = 0; i < \lim; i++) x[i] = x[i] * y[i];
99
        FFT(x, lim, -1);
100
        for (int i = 0; i \le r - l + 1; i++) ans[rt].push_back(l(x[i].r + 0.5) % mod);
101
        for (int i = 0; i < \lim_{i \to +\infty} x[i] = y[i] = Complex(0, 0);
102
103 }
104
    inline void solve() {
105
        init(int(1e6));
106
        int n, a, q; cin >> n >> a >> q;
107
108
        for(int i = 1; i <= n; ++ i) {</pre>
             cin >> s[i]; s[i] = quick_pow(a, s[i]);
109
110
        CDQ_FFT(1, 1, n);
111
        while(q --) {
112
             int k; cin >> k;
113
114
             cout << F(a, n, k) << '\n';
        }
115
116 }
    4.42 FFT
 1 #include <bits/stdc++.h>
 2 using namespace std;
   const double PI = acos(-1);
 4
    const int N = 4e5 + 10;
 5
    struct Complex {
 6
 7
        double a, b;
        Complex(double a = 0, double b = 0): a(a), b(b) {}
 8
 9
        Complex operator * (const Complex &rhs) { return Complex(a * rhs.a - b * rhs.b, a *
         rhs.b + b * rhs.a); }
        Complex operator + (const Complex &rhs) { return Complex(a + rhs.a, b + rhs.b); }
10
        Complex operator - (const Complex &rhs) { return Complex(a - rhs.a, b - rhs.b); }
11
    };
12
13
   int tr[N];
14
15
    void FFT(Complex *A, int len, int type) {
16
        for (int i = 0; i < len; i++) if (i < tr[i]) swap(A[i], A[tr[i]]);
17
        for (int i = 2; i <= len; i <<= 1) {//区间长度
18
             int mid = i / 2;
19
             Complex Wn(cos(2 * PI / i), type * sin(2 * PI / i));//单位根
20
             for (int k = 0; k < len; k += i) {//每个子问题的起始点
21
```

```
Complex w(1, 0);//omega
22
                for (int l = k; l < k + mid; l++) {
23
                    Complex t = w * A[l + mid];
24
                    A[l + mid] = A[l] - t;
25
26
                    A[l] = A[l] + t;
                    w = w * Wn;
27
                }
28
29
           }
30
       }
31
   }
32
33
   void mul(Complex *a, Complex *b, int n) {
       int len = 1; while (len <= n) len <<= 1;</pre>
34
       for (int i = 0; i < len; i++) tr[i] = (tr[i >> 1] >> 1) | (i & 1 ? len >> 1 : 0);
35
       FFT(a, len, 1), FFT(b, len, 1);
36
       for (int i = 0; i < len; i++) a[i] = a[i] * b[i];
37
38
       FFT(a, len, -1);
       for (int i = 0; i < len; i++) a[i].a /= len;
39
40 }
41
   Complex a[N], b[N];
42
43
   void solve() {
44
       int n, m;
45
       scanf("%d%d", &n, &m);
46
       for (int i = 0; i <= n; i++) scanf("%lf", &a[i].a);</pre>
47
       for (int i = 0; i <= m; i++) scanf("%lf", &b[i].a);</pre>
48
49
       mul(a, b, n + m);
50
51
       for (int i = 0; i <= n + m; i++)
            printf("%d'", (int)(a[i].a + 0.5));
52
53 }
   4.43 FWT
   inline void OR(11 * f, int n, int x = 1) {
2
       for (int o = 2; o <= n; o <<= 1) {
3
            for (int i = 0, k = 0 >> 1; i < n; i += 0) {
4
                for (int j = 0; j < k; ++j) {
                    f[i + j + k] = (f[i + j + k] + f[i + j] * x % mod + (x == 1 ? 0 : mod))
5
        % mod;
6
7
            }
8
9
   inline void AND(ll *f, int n, int x = 1) {
10
       for (int o = 2; o <= n; o <<= 1) {
11
12
            for (int i = 0, k = 0 >> 1; i < n; i += 0) {
13
                for (int j = 0; j < k; ++j) {
                    f[i + j] = (f[i + j] + f[i + j + k] * x + (x == 1 ? 0 : mod)) % mod;
14
15
            }
16
       }
17
18
   }
   inline void XOR(ll *f, int n, int x = 1) {
19
       for (int o = 2; o <= n; o <<= 1) {
20
            for (int i = 0, k = 0 >> 1; i < n; i += 0) {
21
                for (int j = 0; j < k; ++j) {
22
```

```
f[i + j] += f[i + j + k],

f[i + j + k] = f[i + j] - f[i + j + k] - f[i + j + k],
23
24
                    f[i + j] *= x, f[i + j + k] *= x;
25
                    f[i + j] \% = mod;
26
27
                    f[i + j + k] \% = mod;
                    while (f[i + j] < 0) f[i + j] += mod;
28
29
                    while (f[i + j + k] < 0) f[i + j + k] += mod;
30
                }
           }
31
       }
32
   }
33
34
   inline void OR(vector < ll > &f, int n, int x = 1) {
35
       for (int o = 2; o <= n; o <<= 1) {
            for (int i = 0, k = 0 >> 1; i < n; i += 0) {
36
                for (int j = 0; j < k; ++j) {
37
                    f[i + j + k] = (f[i + j + k] + f[i + j] * x % mod + (x == 1 ? 0 : mod))
38
        % mod;
39
                }
40
            }
       }
41
42
   43
       for (int o = 2; o <= n; o <<= 1) {
44
            for (int i = 0, k = 0 >> 1; i < n; i += 0) {
45
46
                for (int j = 0; j < k; ++j) {
                    f[i + j] = (f[i + j] + f[i + j + k] * x + (x == 1 ? 0 : mod)) % mod;
47
                }
48
49
           }
       }
50
51
   inline void XOR(vector<ll> &f, int n, int x = 1) {
52
       for (int o = 2; o <= n; o <<= 1) {
53
            for (int i = 0, k = 0 >> 1; i < n; i += 0) {
54
                for (int j = 0; j < k; ++ j) {
55
                    ll X = f[i + j], Y = f[i + j + k];
56
                    f[i + j] = (X + Y) \% mod;
57
                    f[i + j + k] = ((X - Y) \% \mod + \mod) \% \mod;
58
59
                    // mod is a prime
                    if(x != 1) {
60
                        f[i + j] = f[i + j] * inv2 % mod;
61
                        f[i + j + k] = f[i + j + k] * inv2 % mod;
62
                    }
63
64
                }
           }
65
66
   // mod is not a prime, let mod = mod * (1 << m), n = 1 << m;
67
68 //
              if(x != 1) for(int i = 0; i < n; ++ i) {
69 //
                   f[i] /= n;
70 //
               }
71 }
   4.44 NTT
1 Polynomial R;
   inline int Binary_Rounding(const int &n, int len = 1) { //¶0 %0 0 0 0 0 0 0 0
       £¬□ NTT±任□ ± ; f£
3
       while (len < n) len <<= 1;</pre>
       return len;
4
```

```
}
   //O ´ | O O RO O O 養O ± ± ±任£¬O O ÿ´O NTO O jO O O 0 ¼ÇμO O ô¬¬O O ¡£
   inline int Prepare_Transformation(int n) {
        int L = 0, len;
8
9
       for (len = 1; len < n; len <<= 1) L++;
       R.clear();
10
       R.resize(len);
11
       for (int i = 0; i < len; ++i)
12
            R[i] = (R[i >> 1] >> 1) | ((i & 1) << (L - 1));
13
14
       return len;
   }
15
16
   inline void NTT(Polynomial &a, int f) {
17
       int n = a.size();
       for (int i = 0; i < n; ++i)
18
            if (i < R[i])swap(a[i], a[R[i]]);</pre>
19
       for (int i = 1; i < n; i <<= 1)
20
            for (int j = 0, gn = quick_pow(mod_g, (mod - 1) / (i << 1)); <math>j < n; j += (i <<
21
       1))
                for (int k = 0, g = 1, x, y; k < i; ++k, g = 111 * g * gn % mod)
22
                    x = a[j + k], y = 111 * g * a[i + j + k] % mod,
23
                    a[j + k] = (x + y) \% \mod, a[i + j + k] = (x - y + mod) \% \mod;
24
       if (f == -1) {
25
            reverse(a.begin() + 1, a.end());
26
27
            int inv = Inv(n);
28
            for (int i = 0; i < n; ++i) a[i] = 1ll * a[i] * inv % mod;
29
30
   }
   inline Polynomial operator+(const Polynomial &a, const int &b) {
31
       int sizea = a.size();
32
       Polynomial ret = a;
33
34
        ret.resize(sizea);
35
        for (int i = 0; i < sizea; ++i)ret[i] = (1ll * a[i] + b + mod) % mod;</pre>
36
       return ret;
37
   inline Polynomial operator-(const Polynomial &a, const int &b) {
38
       int sizea = a.size();
39
       Polynomial ret = a;
40
41
       ret.resize(sizea);
       for (int i = 0; i < sizea; ++i)ret[i] = (1ll * a[i] - b + mod) % mod:
42
       return ret;
43
44
   inline Polynomial operator*(const Polynomial &a, const int &b) {
45
        int sizea = a.size();
46
47
       Polynomial ret = a;
       ret.resize(sizea);
48
       for (int i = 0; i < sizea; ++i) ret[i] = (1ll * a[i] * b % mod + mod) % mod;
49
50
       return ret;
51
   inline Polynomial operator+(const Polynomial &a, const Polynomial &b) {
52
        int sizea = a.size(), sizeb = b.size(), size = max(sizea, sizeb);
54
       Polynomial ret = a:
55
       ret.resize(size);
       for (int i = 0; i < sizeb; ++i) ret[i] = (1ll * ret[i] + b[i]) % mod;</pre>
56
       return ret;
57
58
   inline Polynomial operator-(const Polynomial &a, const Polynomial &b) {
        int sizea = a.size(), sizeb = b.size(), size = max(sizea, sizeb);
60
61
       Polynomial ret = a;
62
       ret.resize(size);
```

```
for (int i = 0; i < sizeb; ++i) ret[i] = (111 * ret[i] - b[i] + mod) % mod;
63
        return ret;
64
65
    inline Polynomial Inverse(const Polynomial &a) {
66
67
        Polynomial ret, inv_a;
        ret.resize(1);
68
        ret[0] = Inv(a[0]);
69
        int ed = a.size();
70
         for (int len = 2; len <= ed; len <<= 1) {
71
             int n = Prepare_Transformation(len << 1);</pre>
72
73
             inv_a = a;
74
             inv_a.resize(n);
75
             ret.resize(n);
             for (int i = len; i < n; ++i) inv_a[i] = 0;
76
             NTT(inv_a, 1);
77
             NTT(ret, 1);
 78
             for (int i = 0; i < n; ++i)
79
                 ret[i] = 111 * (211 - 111 * inv_a[i] * ret[i] % mod + mod) % mod * ret[i] %
80
         mod;
             NTT(ret, -1);
81
             for (int i = len; i < n; ++i) ret[i] = 0;</pre>
82
83
        ret.resize(ed);
84
85
         return ret;
86
    }
87
88
    inline Polynomial operator*(const Polynomial &a, const Polynomial &b) {
89
        Polynomial lsa = a, lsb = b, ret;
90
         int n = lsa.size(), m = lsb.size();
91
        n = Prepare_Transformation(n + m);
92
        lsa.resize(n);
93
        lsb.resize(n);
94
        ret.resize(n);
95
        NTT(lsa, 1);
96
        NTT(lsb, 1);
97
98
        for (int i = 0; i < n; ++i) ret[i] = 1ll * lsa[i] * lsb[i] % mod;</pre>
99
        NTT(ret, -1);
        return ret;
100
101
    inline Polynomial operator/(const Polynomial &a, const Polynomial &b) {
102
        Polynomial ret = a, ls = b;
103
        reverse(ret.begin(), ret.end());
104
105
        reverse(ls.begin(), ls.end());
        ls.resize(Binary_Rounding(a.size() + b.size()));
106
        ls = Inverse(ls);
107
108
        ls.resize(a.size() + b.size());
        ret = ret * ls;
109
        ret.resize(a.size() - b.size() + 1);
110
111
        reverse(ret.begin(), ret.end());
112
        return ret;
113
114
    inline Polynomial operator%(const Polynomial &a, const Polynomial &b) {
        Polynomial ret = a / b;
115
        ret = ret * b;
116
117
        ret.resize(a.size() + b.size());
        ret = a - ret;
118
119
        ret.resize(a.size() + b.size());
120
        return ret;
```

```
121 }
    inline Polynomial Derivation(const Polynomial &a) {
122
         int size = a.size();
123
        Polynomial ret;
124
125
        ret.resize(size);
        for (int i = 1; i < size; ++i) ret[i - 1] = 1ll * i * a[i] % mod;</pre>
126
127
        ret[size - 1] = 0;
        return ret;
128
129
    inline Polynomial Integral(const Polynomial &a) {
130
        int size = a.size();
131
132
        Polynomial ret;
133
        ret.resize(size);
         for (int i = 1; i < size; ++i) ret[i] = 1ll * Inv(i) * a[i - 1] % mod;
134
        ret[0] = 0;
135
        return ret;
136
137
    inline Polynomial Composition_Inverse(const Polynomial &a) {
138
139
        int n = a.size();
        Polynomial ret, Cinv = a, Pow;
140
        Cinv.resize(n);
141
        ret.resize(n);
142
        Pow.resize(n);
143
        Pow[0] = 1;
144
145
         for (int i = 0; i < n - 1; ++i) Cinv[i] = Cinv[i + 1];
        Cinv[n - 1] = 0;
146
        Cinv = Inverse(Cinv);
147
        for (int i = 1; i < n; ++i) {
148
             Pow = Pow * Cinv;
149
             Pow.resize(n);
150
             ret[i] = 1ll * Pow[i - 1] * Inv(i) % mod;
151
152
        return ret;
153
154
    inline Polynomial Logarithmic(const Polynomial &a) {
155
        Polynomial ln_a = Derivation(a) * Inverse(a);
156
157
        ln_a.resize(a.size());
158
        return Integral(ln_a);
159
    }
    inline Polynomial Exponential(const Polynomial &a, int Constant = 1) {
160
161
        Polynomial ret, D;
        int ed = a.size();
162
        ret.resize(1);
163
164
         ret[0] = Constant;
         for (int len = 2; len <= ed; len <<= 1) {</pre>
165
             D = Logarithmic(ret);
166
             D.resize(len);
167
             D[0] = (111 * a[0] + 111 - D[0] + mod) \% mod;
168
             for (int i = 1; i < len; ++i) D[i] = (1ll * a[i] - D[i] + mod) % mod;
169
             int n = Prepare_Transformation(len << 1);</pre>
170
171
             ret.resize(n);
172
             D.resize(n);
             NTT(ret, 1);
173
             NTT(D, 1);
174
             for (int i = 0; i < n; ++i) ret[i] = 1ll * ret[i] * D[i] % mod;</pre>
175
             NTT(ret, -1);
176
177
             for (int i = len; i < (len << 1); ++i) ret[i] = D[i] = 0;
178
179
        ret.resize(ed);
```

```
180
        return ret;
181 }
    4.45
          高斯消元
 1 #include <bits/stdc++.h>
 2 using namespace std;
 3 const int N=210;
 4 int a[N][N];//增广矩阵
 5 int x[N];//解集
   int freeX[N];//自由变元
   // equ:方程个数 var:变量个数
 7
    int Gauss(int equ, int var){//返回自由变元个数
 8
 9
        /*初始化*/
        for(int i = 0; i \le var; i++){
10
11
           x[i] = 0;
           freeX[i] = 0;
12
        }
13
14
        /*转换为阶梯阵*/
15
        int col = 0;//当前处理的列
16
        int num = 0; // 自由变元的序号
17
        int k;//当前处理的行
18
        for(k = 0; k < equ \&\& col < var; k++, col++) {//枚举当前处理的行
19
            int maxr = k; // 当前列绝对值最大的行
20
21
            for(int i = k + 1; i < equ; i++){//寻找当前列绝对值最大的行
22
               if(a[i][col] > a[maxr][col]){
23
                   maxr = i;
24
                   swap(a[k], a[maxr]);//与第k行交换
25
                   break;
               }
26
27
28
            if(a[k][col] == 0){//col}列第k行以下全是0,处理当前行的下一列
29
               freeX[num++] = col;//记录自由变元
30
               k--;
               continue;
31
           }
32
33
34
           for(int i = k + 1; i < equ; i++){}
35
               if(a[i][col] != 0){
36
                   for(int j = col; j < var + 1; j++){//对于下面出现该列中有1的行,需要把1消掉
                       a[i][j] ^= a[k][j];
37
38
39
               }
40
           }
        }
41
42
        /*求解*/
43
44
        //无解: 化简的增广阵中存在(0,0,\ldots,a)这样的行,且a!=0
45
        for(int i = k;i < equ; i++)
            if(a[i][col] != 0)
46
               return -1;
47
48
49
        //无穷解: 在var*(var+1)的增广阵中出现(0,0,\ldots,0)这样的行
50
        if(k < var)//返回自由变元数
           return var - k;//自由变元有var-k个
51
52
        //唯一解: 在var*(var+1)的增广阵中形成严格的上三角阵
53
```

```
for(int i = var - 1;i >= 0; i--){//计算解集
54
55
           x[i] = a[i][var];
           for(int j = i + 1; j < var; j++)
56
57
               x[i] ^= (a[i][j] \&\& x[j]);
58
59
       return 0;
60
  }
   4.46
          高斯消元 2
1 #include<bits/stdc++.h>
2 using namespace std;
3 #define MAX_SIZE 1048
4 int Matrix[MAX_SIZE][MAX_SIZE];
5 int Free_x[MAX_SIZE]; //0 0 0 m0 0
6 int X_Ans[MAX_SIZE]; //½__
  int Free_num=0; //0 0 0 m0 0 0 0
7
   int gcd(int a,int b) { return b==0? a : gcd(b,a%b); }
   int lcm(int a,int b) { return a/gcd(a,b)*b; }
9
10
   int Guass(int Row,int Column) {//0 0 0 ¾0 0 0 0 0 9-0 0
11
       int row=0,col=0,max_r;
12
13
       for(row=0;row<Row&&col<Column;row++,col++) {</pre>
14
           max_r=row;
           for(int i=row+1;i<Row;i++)</pre>
                                         //O xO jO eO O O O
15
               if(abs(Matrix[i][col])>abs(Matrix[max_r][col]))
16
17
                   max_r=i;
           if(Matrix[max_r][col]==0) { //0 0 0 0 0£-\u00f40 0 0 0 0 0 £-\u00e40 \u00e40
18
19
               Free_x[++Free_num]=col+1;
20
               continue;
21
22
           if(max_r!=row)
                           //½«O O O O »»μ½μ±jO O
23
               for(int i=col;i<Column+1;i++)</pre>
24
25
                   swap(Matrix[row][i],Matrix[max_r][i]);
26
           for(int i=row+1;i<Row;i++) { //[ [ ] [</pre>
               if(Matrix[i][col]!=0) {
27
                   int LCM=lcm(abs(Matrix[i][col]),abs(Matrix[row][col]));
28
                   int ta=LCM/abs(Matrix[i][col]);
29
                   int tb=LCM/abs(Matrix[row][col]);
30
                   31
32
                       tb=-tb;
                   for(int j=col;j<Column+1;j++)</pre>
33
                       Matrix[i][j]=Matrix[i][j]*ta-Matrix[row][j]*tb;
34
35
               }
           }
36
37
       //row0 0 30 h±0 30 0 0 0 0 0 0 0 0
38
39
       40
           if(Matrix[i][Column]!=0)
41
               return -1;
42
43
44
       if(row<Column)</pre>
                       //O O O O O O ·µ»O O O O mO O O
45
           return Column-row;
46
47
       for(int i=Column-1;i>=0;i--) { //Ψμ½□
48
```

```
int temp=Matrix[i][Column];
49
            for(int j=i+1; j<Column; j++)</pre>
50
                 if(Matrix[i][j]!=0)
51
                     temp-=Matrix[i][j]*X_Ans[j];
52
            X_Ans[i]=temp/Matrix[i][i];
53
54
55
        return 0;
   }
56
   4.47
           高斯消元异或
   #include<bits/stdc++.h>
   using namespace std;
   #define MAX_SIZE 350
3
   #define ll long long
   11 Matrix[MAX_SIZE][MAX_SIZE];
   ll Free_x[MAX_SIZE];
                            //O O O mO O
   11 X_Ans[MAX_SIZE];
                         //%工
7
   ll Free_num=0; //0 0 0 m0 0 0
8
9
   ll Guass(ll Row,ll Column) { //□ □ □ ¾□ □ □ □ κ□ □ □
10
        11 row=0,col=0,max_r;
11
        for(row=0;row<Row&&col<Column;row++,col++) {</pre>
12
            max_r=row;
13
            for(ll i=row+1;i<Row;i++) //0 x0 j0 0 0 0 0
14
                 if(abs(Matrix[i][col])>abs(Matrix[max_r][col]))
15
                     max_r=i;
16
            if(Matrix[max_r][col]==0) { //¼0 ¼0 0 0 m0 0
17
18
19
                Free_x[Free_num++]=col+1;
                continue;
20
21
            if(max_r!=row) //½»»»
22
23
                 for(ll i=col;i<Column+1;i++)</pre>
                     swap(Matrix[row][i],Matrix[max_r][i]);
24
25
            for(ll i=row+1;i<Row;i++) { //[ [ ] [</pre>
26
                if(Matrix[i][col]!=0) {
27
                     for(ll j=col;j<Column+1;j++)</pre>
28
                         Matrix[i][j]^=Matrix[row][j];
29
                }
30
            }
31
        for(ll i=row;i<Row;i++)</pre>
32
33
            if(Matrix[i][Column]!=0)
34
                 return -1;
35
        if(row<Column)</pre>
                          //0 0 0 0 0 0
36
            return Column-row;
37
38
        //Yh½0
39
40
        for(ll i=Column-1;i>=0;i--) {
            X_Ans[i]=Matrix[i][Column];
41
            for(ll j=i+1; j<Column; j++)</pre>
42
               X_Ans[i]^=(Matrix[i][j]&&X_Ans[j]);
43
44
45
        return 0;
46
   }
```

4.48 矩阵快速幂

```
/***¾0 0 0 0 0 0 0 ***/
1
2
3
   struct Matrix {
4
       static const int M = 2;
5
       double mx[M][M];
6
       Matrix() { memset(mx, 0, sizeof mx); }
7
8
       void Out() {
9
           for (auto &i : mx) {
10
                for (auto j : i)
11
                    cout << j << ' ';
12
               cout << endl;</pre>
13
14
           }
       }
15
16
17
       void Tranfer_E() {
18
           memset(mx, 0, sizeof mx);
           for (int i = 0; i < M; ++i) mx[i][i] = 1;
19
20
21
       Matrix operator*(const struct Matrix a) const {
22
           Matrix x;
23
           memset(x.mx, 0, sizeof x.mx);
24
           for (int i = 0; i < M; ++i)
25
                for (int j = 0; j < M; ++j)
26
27
                    for (int k = 0; k < M; ++k)
28
                        x.mx[i][j] = (x.mx[i][j] + mx[i][k] * a.mx[k][j]);
29
           return x;
       }
30
31
   };
   Matrix mat_pow(Matrix mx, int p) {
32
33
       Matrix res;
       res.Tranfer_E();
34
       for (; p; p >>= 1, mx = mx * mx)
35
           if (p \& 1) res = res * mx;
36
37
       return res;
38 }
          矩阵求逆
   4.49
   /***¾0 0 0 0 0 0 0 ***/
   void Gauss_jordan() {
       3
       for(int i = 0, r; i < n; ++ i) {
                                            //O O O !O O O O iO O
4
5
           r = i;
           for(int j = i + 1; j < n; ++j)
6
                if(fabs(a[j][i]) > fabs(a[r][i])) r = j;
7
8
           if(fabs(a[r][i]) < eps) {
               puts("No Solution");
9
10
               return;
11
           if(i!=r) swap(a[i],a[r]);
12
13
           for(int k = 0; k < n; ++ k) {
14
               //ÿh[ x¼´¦[ [
15
```

```
if(k==i) continue;
16
                double p=a[k][i]/a[i][i];
17
                for(re int j=i; j<=n; ++j) a[k][j]-=p*a[i][j];</pre>
18
19
           }
       }
20
21
       //O O O O O O O O do ¶O ØO O O ,´O C³O O O O
22
       for(int i = 0; i < n; ++i) printf("%.2lf\n",a[i][n+1]/a[i][i]);</pre>
23
   }
24
   4.50 线件基
   // 每个异或值都相同的个数都为2^n-r,所以不同的异或值有2^r个.
   #include <bits/stdc++.h>
3 using namespace std;
4
  typedef long long ll;
5
   const int maxl = 60;
7
   ll quick_pow(ll a, ll b);
8
   struct LinearBasis {
       ll a[maxl + 10];
9
10
       int n, size; // 每个相同异或值有2^{n-size}个
       vector<ll> v;
11
12
13
       LinearBasis() {
14
           memset(a, 0, sizeof(a));
15
            size = n = 0;
16
           v.clear();
17
       }
18
19
       void insert(ll t) {
20
            n++;
21
            for (int i = maxl; i >= 0; --i) {
22
                if (!(t >> i & 1)) continue;
                if (a[i]) t ^= a[i];
23
                else {
24
25
                    ++size;
                    for (int j = i - 1; j >= 0; j--) if (t >> j \& 1) t ^- a[j];
26
27
                    for (int j = i + 1; j \le maxl; ++j) if (a[j] >> i & 1) a[j] ^= t;
28
                    a[i] = t;
29
                    return;
30
                }
31
           }
       }
32
33
       void basis() {
34
            for (int i = 0; i \le maxl; ++i) if (a[i]) v.push_back(i);
35
       }
36
37
       // 查询能否xor出x这个数
38
39
       bool find(ll x) {
            for(int i = maxl;i >= 0; i--) {
40
                if(x >> i \& 1) {
41
                    if(!a[i]) return 0;
42
                    x ^= a[i];
43
44
                }
            }
45
            return 1;
46
```

```
}
47
48
        // 查询异或最大值
49
        11 askmax() {
50
51
            11 \text{ ans} = 0;
            for(int i = maxl; i \ge 0; i--) ans = max(ans, ans ^ a[i]);
52
            return ans;
53
        }
54
55
        // 查询异或最小值
56
57
        11 askmin() {
58
            for(int i = 0;i <= maxl; i++) if(a[i]) return a[i];</pre>
59
            return 0;
        }
60
61
        // 查询异或第k大
62
        ll askmaxk(ll x) {
63
64
        }
65
66
        // 查询异或第k小
67
        ll askmink(ll x) {
68
            if(v.size() != n) x--;
69
70
            if(!x) return 0;
71
            if(x >= (111 << v.size())) return -1;
            ll ans = 0;
72
            for(int i = 0;i < v.size(); i++) {</pre>
73
                if(x >> i & 1) ans \wedge= a[v[i]];
74
75
76
            return ans;
        }
77
78
        ll rank(ll x) {
79
80
            ll ret = 0;
            for (int i = 0; i < v.size(); ++i) if (x >> v[i] & 1) ret += 1LL << i;
81
            return ret;
82
83
        }
84
   };
85
86
   void solve() {
87
        int n, x, q;
88
        scanf("%d", &n);
89
90
        LinearBasis lb;
        for (int i = 0; i < n; ++i) scanf("%d", &x), lb.insert(x);
91
        lb.basis();
92
93
        scanf("%d", &q);
        11 num = quick_pow(2, n - lb.size);
94
        printf("%lld\n", (lb.rank(q) * num + 1));
95
96 }
   4.51 线件基
   struct LinerBase {
        static const int MAX_BIT = 61;
2
        bool isZero; ll num[N], tmp[N];
3
 4
        //判断线性基中是否存在0
        LinerBase() {
```

```
memset(num, 0, sizeof num);
6
7
            memset(tmp, 0, sizeof tmp);
8
        void insert(ll x) {
9
            for (int i = MAX_BIT; i >= 0; i--) {
10
                 if (!(x & (1ll << i))) continue;</pre>
11
                if (num[i]) {
12
                     x \sim num[i];
13
                } else {
14
15
                     num[i] = x;
                     return;
16
17
                }
18
            isZero = true;
19
20
        bool find(ll x) {
21
            for (int i = MAX_BIT; i >= 0; i--) {
22
23
                 if (x & (1ll << i)) {
                     if (!num[i]) return false;
24
25
                     x \sim num[i];
                }
26
27
            }
            return true;
28
29
30
        int countNumber() {
31
            int cnt = 0;
            for (int i = 0; i <= MAX_BIT; i++) if (num[i] & (1ll << i)) cnt++;</pre>
32
            return cnt;
33
34
        void rebuild() {
35
36
            int cnt = 0;
            for (int i = MAX_BIT; i >= 0; i--)
37
                 for (int j = i-1; j >= 0; j--)
38
                     if (num[i] & (1ll << j)) num[i] ^= num[j];</pre>
39
40
            for (int i = 0; i <= MAX_BIT; i++) {</pre>
41
42
                if (num[i]) tmp[cnt++] = num[i];
43
44
        ll qryKth(ll k) {
45
            if (isZero) k--;
46
            if (!k) return 0;
47
            int cnt = countNumber(); ll ans = 0;
48
49
            if (k >= (1ll << cnt)) return -1;
            for (int i = 0; i < cnt; i++) {</pre>
50
                if (k & (1ll << i)) ans ^= tmp[i];</pre>
51
52
            return ans;
53
        }
54
  }lb;
   4.52
           康托展开
1 #include <iostream>
2 #include <vector>
   #include <algorithm>
3
5 using namespace std;
```

```
6
   typedef long long ll;
   const int mod = 1e9 + 7;
8
9 const int N = 1e5 + 10;
10
12 int a[N]; // 排列, 康托展开求解
13 int n;
14 ll x; // 逆康托展开求解
15
   void Get_F() {
16
17
       fac[0] = 1;
       for(int i = 1; i < N; i++)
18
            fac[i] = fac[i - 1] * i % mod;
19
   }
20
21
   11 CanTor() {
22
       ll ans = 0;
23
       for(int i = 1;i <= n; i++) {</pre>
24
            ll smaller = 0;
25
            for(int j = i + 1; j \le n; j++) {
26
                if(a[j] < a[i])
27
                    smaller++;
28
29
30
            ans = (ans + fac[n - i] * smaller % mod) % mod;
31
32
       return ans + 1;
   }
33
34
   void DeCantor() {
35
       vector<int> v; // 存放当前可选数
36
37
       vector<int> a; // 所求的排列组合序
38
       for(int i = 1;i <= n; i++) {</pre>
            v.push_back(i);
39
40
        for(int i = n;i >= 1; i--) {
41
42
            int r = x \% fac[i - 1];
43
            int t = x / fac[i - 1];
44
            x = r;
            sort(v.begin(), v.end());
45
           a.push_back(v[t]);
46
           v.erase(v.begin() + t);
47
48
49
        for(int i = 0; i < a.size(); i++)
            cout << a[i] << " ";
50
       cout << endl;</pre>
51
52 }
53
54 // 线段树优化
56 const int N = 1000010;
57
58 ll fac[N];
   int a[N]; // 排列, 康托展开求解
59
   int n;
60
61
62
   struct SegmentTree {
63
       int ls, rs;
       int sum;
64
```

```
65 }t[N << 2];
66
    int cnt, root;
67
68
    void push_up(int u) {
69
         t[u].sum = (t[lc].sum + t[rc].sum) % mod;
70
71
72
    void build(int &u, int l, int r) {
73
74
         if(!u) u = ++cnt;
75
         if(l == r) {
76
             t[u].sum = 1;
77
             return ;
         }
78
         build(lc, l, m);
79
         build(rc, m + 1, r);
80
81
         push_up(u);
82
    }
83
    void update(int &u, int l, int r, int k) {
84
         if(!u) u = ++cnt;
85
         if(l == r) {
86
             t[u].sum = 0;
87
88
             return ;
89
         if(k <= m) update(lc, l, m, k);</pre>
90
         else update(rc, m + 1, r, k);
91
         push_up(u);
92
    }
93
94
    ll query(int u, int l, int r, int ql, int qr) {
95
         if(ql > qr) return 0;
96
97
         if(ql == l \&\& qr == r) {
             return t[u].sum;
98
99
         if(qr <= m) return query(lc, l, m, ql, qr) % mod;</pre>
100
101
         else if(ql > m) return query(rc, m + 1, r, ql, qr) % mod;
102
         else return (query(lc, l, m, ql, m) + query(rc, m + 1, r, m + 1, qr)) % mod;
103
    }
104
    void Get_F() {
105
         fac[0] = 1;
106
         for(int i = 1;i < N; i++)</pre>
107
             fac[i] = fac[i - 1] * i % mod;
108
109
    }
110
111 void solve()
112 {
         Get_F();
113
114
         cin >> n;
115
         build(root, 1, n);
116
         11 \text{ ans} = 0;
         for(int i = 1;i <= n; i++) {</pre>
117
             cin >> a[i];
118
             update(root, 1, n, a[i]);
119
             ans = (ans + query(root, 1, n, 1, a[i] - 1) * fac[n - i]) % mod;
120
121
122
         cout << (ans + 1) % mod << endl;
123 }
```

4.53 模数非质数的组合

```
1 // 模数非质数情况下的组合问题
  // one way, use CRT merge ans
   // https://ac.nowcoder.com/discuss/655940?type=101&order=0&pos=2&page=1&channel=-1&
       source_id=discuss_tag_nctrack
4 // another way
5 // https://ac.nowcoder.com/acm/contest/view-submission?submissionId=47754622
   #include <bits/stdc++.h>
7
8
9 using namespace std;
10 typedef long long ll;
   const int N = 1e6 + 10;
11
12
   ll qpow(ll a, ll b, ll mod) {
13
       ll res = 1;
14
15
       while (b) {
            if (b & 1) res = res * a % mod;
16
            a = a * a \% mod;
17
           b >>= 1;
18
19
20
       return res;
   }
21
22
   ll exgcd(ll a, ll b, ll &x, ll &y) {
23
24
       if (!b) {
25
            x = 1, y = 0;
26
            return a;
27
       ll res = exgcd(b, a \% b, x, y);
28
29
       11 t = y;
30
       y = x - a / b * y;
31
       x = t;
32
       return res;
33 }
34
35 ll inv(ll a, ll b) {
36
       11 x = 0, y = 0;
       exgcd(a, b, x, y);
37
38
       return x = (x \% b + b) \% b;
39 }
40
   //r□为余数,m为模数,其中模数互质
42 //M = pi(mi), Mi = M / mi, invMi = Mi % mi
43 //ni满足是除了mi之外的倍数,且模mi为ri
44 //利用逆元性质,即ri * Mi * invMi = ri (mod mi)
45 //res = (sigma(ri * Mi * invMi)) % M
46
   ll china(ll r[], ll m[], int n) {
47
       ll M = 1, res = 0;
48
       for (int i = 1; i <= n; i++) M *= m[i];
49
       for (int i = 1; i <= n; i++) {
50
            ll Mi = M / m[i], invMi = inv(Mi, m[i]);
51
            res = (res + r[i] * Mi % M * invMi % M) % M;
52
            //res = (res + mul(mul(r[i], Mi, M), invMi, M)) % M;按位乘
53
54
       return (res % M + M) % M;
55
  }
56
```

```
57
   int f[N], g[N], F[N], G[N], invF[N];
59
    int calc(int n, int p, int k) {
60
61
        11 mod = qpow(p, k, LONG_LONG_MAX);
        F[0] = 1, G[0] = 0;
62
        for (int i = 1; i <= n; i++) {
63
            g[i] = 0, f[i] = i;
64
            while (f[i] \% p == 0) f[i] /= p, g[i] ++;
65
            F[i] = 111 * F[i - 1] * f[i] % mod;
66
            G[i] = G[i - 1] + g[i];
67
68
        }
        invF[n] = inv(F[n], mod);
69
        for (int i = n; i >= 1; i--) invF[i - 1] = 1ll * invF[i] * f[i] % mod;
70
        int ans = 0;
71
        for (int i = 0; i <= n / 2; i++) {
72
            73
74
            ans = (ans + 111 * t) % mod;
75
76
77
        return ans;
78 }
79
80 ll r[20], m[20];
81
   int main() {
82
    #ifdef ACM_LOCAL
83
        freopen("input.in", "r", stdin);
84
        freopen("output.out", "w", stdout);
85
    #endif
86
        int n, p;
scanf("%d%d", &n, &p);
87
88
        int num = 0;
89
        for (int i = 2; i * i <= p; i++)
90
            if (p \% i == 0) {
91
                int k = 0;
92
93
                m[++num] = 1;
94
                while (p % i == 0) p /= i, k++, m[num] *= i;
                r[num] = calc(n, i, k);
95
            }
96
        if (p > 1) {
97
            m[++num] = p;
98
99
            r[num] = calc(n, p, 1);
100
        printf("%lld\n", china(r, m, num));
101
102
103
        return 0;
104 }
          普通型母函数
    4.54
 1 // 普诵型母函数: (1+x^1+x^2+...) (1+x^2+x^4)(1+x^3+x^6..)(...)(...)... 类似整数拆分
 3 // a_n=1,1,1,1... = \frac{1}{1-x}
   // a_n=1,0,1,0... = \frac{1}{1-x^2}
 5 // a_n=1,2,3,4... = \frac{1}{(1-x)^2}
 6 // a_n=C(m,n)
                     = (1+x)^m
 7 // a_n=C(m+n,n) = \frac{1}{(1-x)^{m+1}}
```

```
8
   #include <bits/stdc++.h>
9
   using namespace std;
10
11
  typedef long long ll;
12
13
14 // 求解硬币等普通问题
15
  const int N = 1e5 + 10;
16
17
  int a[N]; // 权重为i的组合数, a[P]为答案
18
  int b[N]; // 辅助数组
20 int P; // 需要被分解的数
21 int k; // 物品个数
  int v[N]; // 每个物品的权重
  int n1[N]; // 对于每种物品起始的因子(所需要的每个物品最小个数), 最小为0
  int n2[N]; // 对于每种物品最终的因子(所需要的每个物品最大个数),最大为INF
25
26 // 模板一(标准)
27
28 void Calc1() {
29
       memset(a, 0, sizeof(a));
       a[0] = 1;
30
31
32
       for(int i = 1;i <= k; i++) { // 枚举每个物品因子
          memset(b, 0, sizeof(b));
33
          for(int j = n1[i]; j <= n2[i] && j * v[i] <= P; j++) { // 每个物品从最小因子到最大因
34
      子循环,如果n2是无穷的,则j<=n2[i]可以删去
              for(int m = 0;m + j * v[i] <= P; m++) { // 循环a的每个项
35
                  b[m + j * v[i]] += a[m]; // 把结果加到对应项里,有点dp的味道
36
37
38
39
          memcpy(a, b, sizeof(b));
       }
40
   }
41
42
   // 模板二(数据量大的时候可以用, 快速)
43
44
   void Calc2() {
45
       memset(a, 0, sizeof(a));
46
       a[0] = 1;
47
       int last = 0;
48
       for(int i = 1; i <= k; i++) {
49
           int last2 = min(last + n2[i] * v[i], P);//计算下一次的last
50
          memset(b, 0, sizeof(int) * (last2 + 1));//只清空b[0..last2]
51
          for(int j = n1[i]; j <= n2[i] && j * v[i] <= last2; j++) //last2</pre>
52
              for(int m = 0; m <= last && m + j * v[i] <= last2; m++) //一个是last, 一个是
53
      last2
                  b[m + j * v[i]] += a[m];
54
          memcpy(a, b, sizeof(int) * (last2 + 1));//b赋值给a. 只赋值0..last2
55
56
          last = last2;//更新last
57
       }
58 }
   4.55
         牛成函数
2 // O(n*sqrt(n))
```

```
#include <bits/stdc++.h>
5
 6
   using namespace std;
7
   typedef long long 11;
8
9
  const int N = 1e5 + 5;
10
   const ll mod = 1e9 + 7;
   int f1[270], f2[270];
13
14 ll ans[N];
15
16
   void init() {
        for (int i = 1; ; i++) {
    f1[i] = (3 * i * i - i) >> 1;
17
18
            if (f1[i] > 100000) break;
19
20
            f2[i] = (3 * i * i + i) >> 1;
            if (f2[i] > 100000) break;
21
22
        ans[0] = 1;
23
        for (int i = 1; i <= 100000; i++) {
24
            for (int j = 1; j++) {
25
26
                 if (f1[j] \le i) ans[i] += j & 1 ? ans[i-f1[j]] : -ans[i-f1[j]];
27
                 else break;
                 if (f2[j] \le i) ans[i] += j & 1 ? ans[i-f2[j]] : -ans[i-f2[j]];
28
                 else break;
29
30
            ans[i] = (ans[i] \% mod + mod) \% mod;
31
32
        }
33 }
           斯特林数
   4.56
   1
2 1
3 1 1
 4 2 3 1
5 6 11 6 1
   24 50 35 10 1
   120 274 225 85 15 1
7
   720 1764 1624 735 175 21 1
8
   namespace STIRLING {
9
        typedef long long 11;
10
11
        const int N = 21;
        const int mod = 1e9 + 7;
12
        ll Stirling1[N][N], fac[N] = \{1\};
13
        void get_stirling1() {
14
             for(int i = 1; i < N; i++)</pre>
15
                 fac[i] = 111 * fac[i - 1] * i % mod;
16
17
            Stirling1\lceil 0 \rceil \lceil 0 \rceil = 0;
            Stirling1\lceil 1 \rceil \lceil 1 \rceil = 1;
18
            for(int i = 2; i < N; i++) {
19
20
                 for(int j = 1; j <= i; j++) {
                     Stirling1[i][j] = (Stirling1[i - 1][j - 1] + (i - 1) * Stirling1[i -
21
        1][j] % mod) % mod;
22
                 }
23
            }
```

```
}
24
25
   1
   1 1
26
   1 3 1
27
   1761
   1 15 25 10 1
29
   1 31 90 65 15 1
   1 63 301 350 140 21 1
   1 127 966 1701 1050 266 28 1
32
       11 Stirling2[N][N];
33
34
       void get_stirling2() {
35
           Stirling2[0][0] = 0;
           Stirling2[1][1] = 1;
36
           for(int i = 2; i < N; i++) {
37
               for(int j = 1; j <= i; j++) {</pre>
38
                   Stirling2[i][j] = (Stirling2[i - 1][j - 1] + j * Stirling2[i - 1][j] %
39
       mod) % mod;
40
               }
41
           }
42
       }
  }
43
          五边形数
   4.57
   /***O O O O O O ***/
1
2
   namespace WUBIANXING {
3
   /*1,5,12,22,35..n^2+n(n-1)/2*/
       const int N = 2e5 + 10;
4
       const int mod = 1e5 + 10;
5
       int f[N], p[N];
6
7
       void init() {
           for (int i = 1; i < N; i++) {
8
               f[i + i - 1] = (111 * i * (i + i + i - 1) >> 1) % mod;
9
               f[i + i] = (111 * i * (i + i + i + 1) >> 1) \% mod;
10
11
           for (int i = p[0] = 1; i < N; i++) {
12
               p[i] = 0;
13
               for (int j = 1, k = -1; i >= f[j]; j++) {
14
15
                   if (j \& 1) k *= -1;
                   p[i] = (p[i] + 111 * k * p[i - f[j]] % mod) % mod;
16
17
               p[i] = (p[i] + mod) \% mod;
18
           }
19
20
       }
   }
21
          指数型母函数
   4.58
1
   // 需要借助e^x的泰勒展开,一般求解多重排列数,即有 种物品,已知每种物品的数量为 k1,k2,...,kn 个,求
       从中选出m件物品的排列数。
3
   // 对n个元素全排列,方案数为n!/(n1!n2!...nk!),对n个中的r个元素进行全排列,这里就用到了指数型母函
       数, 即G(x)=(1+x/1!+x^2/2!+...+x^k1/k1!)(1+x/1!+x^2/2!+...+x^k2/k2!)...(1+x/1!+x
       ^{2/2!+...+x^{kn/kn!}}
  // 化简得G(x)=a0 + a1*x+a2*x^2/2!+...+ap*x^p/p!
                                                  (p = k1+k2+k3+...) ai为洗出i个物品的排列
       方案数
```

```
7
   // 若题目有规定条件,比如需要物品i出现非0的偶数次,即原式为(x^2/2!+x^4/4!+...+x^ki/ki!)
8
9
   #include <bits/stdc++.h>
10
   using namespace std;
11
12
   typedef long double ld;
13
14
   double num[15]; // 每种物品的数量, 第i个物品有num[i]个
15
16
   double a[15], b[15];
17
18
   double f[120]; // 阶乘
19
20
   void fac()
21
22
23
       f[0] = 1;
       for(int i = 1;i <= 105; i++)
24
           f[i] = f[i - 1] * i;
25
   }
26
27
   void Calc() {
28
29
       int n, m;
30
       cin >> n >> m;
31
       for(int i = 1;i <= n; i++)</pre>
            cin >> num[i];
32
33
       memset(a, 0, sizeof(a));
34
       memset(b, 0, sizeof(b));
35
36
       for(int i = 0;i <= num[1]; i++) {</pre>
37
38
           a[i] = 1.0 / f[i];
39
       }
40
       for(int i = 2;i <= n; i++) {</pre>
41
            for(int j = 0; j <= m; j++) {
42
43
                for(int k = 0; k \le num[i] \&\& j + k \le m; k++) {
44
                    b[j + k] += a[j] / f[k];
                }
45
           }
46
47
            for(int j = 0; j <= m; j++) {</pre>
48
                a[j] = b[j];
49
50
                b[j] = 0;
            }
51
52
       }
53
       cout << a[m] * f[m] << endl;</pre>
54
  }
55
          组合数学基础
   4.59
   1
2
   namespace COMB {
       typedef long long ll;
3
       const int mod = 1e9 + 7;
4
       const int N = 5e5 + 10;
5
6
       int fac[N];
```

```
void get_fac(int n, int i = 1) {
7
            for (fac[0] = 1; i \le n; i++) fac[i] = 111 * fac[i - 1] * i % mod;
8
9
       Il quick_pow(ll ans, ll p, ll res = 1) {
10
           ans %= mod, p %= mod - 1;
11
           for (; p; p >>= 1, ans = 1ll * ans * ans % mod) {
12
               if (p & 1) res = 1ll * res * ans % mod;
13
14
           return res % mod;
15
16
       ll inv(ll ans) {
17
18
           return quick_pow(ans, mod - 2);
19
       ll C(ll n, ll m) {
20
           if (m == 0 || n == m) return 1;
21
           if (n < m \mid l \mid n < 0 \mid l \mid m < 0) return 0;
22
           return 1ll * fac[n] % mod * inv(1ll * fac[m] * fac[n - m] % mod) % mod;
23
24
       ll Lucas(ll n, ll m) {
25
           if (m == 0) return 1;
26
           return 1ll * (C(n % mod, m % mod) * Lucas(n / mod, m / mod)) % mod;
27
28
       }
29
  }
   4.60 组合数
   ll f[N];
2
   ll ksm(ll a, ll b) {
3
       ll res = 1, base = a;
       while (b) {
4
           if (b & 1) res = res * base % MOD;
5
           base = base * base % MOD;
6
7
           b >>= 1;
8
9
       return res;
10 }
11
   void init() {
12
13
       for (ll i = 1; i < N; i++) f[i] = f[i-1] * i % MOD;
14
   ll C(ll a, ll b) {
15
       if (a < 0 \mid | b < 0 \mid | b > a) return 0;
16
       return f[a] * ksm(f[a-b], MOD-2) % MOD * ksm(f[b], MOD-2) % MOD;
17
18
19
             -----//卢卡斯定理
   11 f[N];
20
   ll ksm(ll a, ll b, ll mm) {
21
       ll res = 1, base = a;
22
       while (b) {
23
           if (b & 1) res = res * base % mm;
24
25
           base = base * base % mm;
26
           b >>= 1:
       }
27
28
       return res;
29
   }
   void init(ll mm) {
30
31
       for (ll i = 1; i < N; i++) f[i] = f[i-1] * i % mm;
32
```

```
33 }
   il C(ll a, ll b, ll mm) {
       if (a' < 0 | | b < 0 | | b > a) return 0;
35
       return f[a] * ksm(f[a-b], mm-2, mm) % mm * ksm(f[b], mm-2, mm) % mm;
36
   }
37
38
   ll lucas(ll a,ll b,ll mm) {
39
       if(b == 0) return 1;
40
       return C(a % mm,b % mm,mm) * lucas(a / mm,b / mm,mm) % mm;
41
42 }
   4.61 卡特兰数
1 namespace KART {
   /*1,1,2,5,14,42,132,429,1430,4862,16796...*/
   /*h(n) = h(n-1)(4n+2)/(n+1) = C(2n,n)/(n+1) = C(2n,n)-C(2n,n-1) (n=0,1,2,...)*/
3
       typedef long long ll;
4
       const int N = 1e5 + 10;
5
       const int mod = 1e9 + 7;
6
       int cart[N], inv[N];
7
       void get_inv() {
8
            inv[0] = inv[1] = 1;
9
            for(int i = 2; i < N; ++i)</pre>
10
                inv[i] = 1ll * (mod - mod / i) * inv[mod % i] % mod;
11
12
       void get_cart() {
13
            cart[0] = 1;
14
            for(int i = 1; i < N; ++ i)
15
                cart[i] = 111 * cart[i - 1] * ((4 * i % mod - 2 + mod) % mod) % mod * inv[i
16
        + 1] % mod;
17
18 }
```

5 随机化算法

```
int n, m, r;
2
   circle c[20];
   Point p[1010], ansp;
   double ans, now, nxt;
5
   double Rand() { return (double)rand() / RAND_MAX; }
6
   double calc(Point P) {
7
        double minn = r;
8
        rep(i, 1, n) minn = min(minn, P.distance(c[i].p) - c[i].r);
9
10
        double res = 0;
        rep(i, 1, m) if(P.distance(p[i]) <= minn) res ++;</pre>
11
        return res;
12
13 }
14
   void SA() {
15
16
        double x = ansp.x, y = ansp.y;
        double t = 3722;
17
        while(t > 9e-16) {
18
            double X = x + ((rand() * 2) - RAND_MAX) * t;
19
            double Y = y + ((rand() * 2) - RAND_MAX) * t;
20
            now = calc(Point(X, Y));
21
            double Delta = now - ans;
22
23
            if(Delta > 0) {
                ansp = Point(X, Y);
24
25
                x = X, y = Y;
26
                ans=now;
27
            } else if(exp(-Delta/t) * RAND_MAX < rand()) x = X, y = Y;
28
            t *= 0.997577;
        }
29
30
31
   inline void solve() {
32
        srand(time(0));
        cin >> n >> m >> r;
33
        rep(i, 1, n) c[i].input();
34
        rep(i, 1, m) {
35
36
            p[i].input();
37
            ansp = ansp + p[i];
38
        }
39
        ansp = ansp / m;
40
        ans = calc(ansp);
        while ((double)clock()/CLOCKS_PER_SEC < 0.5) SA();</pre>
41
42
        cout << ans << '\n';
43 }
   5.1 SA2
1 circle c[4];
  Point p;
   double dx[4]=\{0, 1, 0, -1\},\
4
           dy[4]=\{1, 0, -1, 0\};
5
6
   double calc(Point a) {
7
        double dif = 0, d[3];
8
        for(int i = 0; i < 3; ++ i)
9
            d[i] = a.distance(c[i].p) / c[i].r;
10
```

```
for(int i = 0; i < 3; ++ i)
11
             for(int j = i + 1; j < 3; ++ j)
    dif += (d[i] - d[j]) * (d[i] - d[j]);</pre>
12
13
        return dif;
14
15 }
   void solve() {
16
        double dis = 1.0, now, t;
17
18
        for(int i = 0; i < 3; ++ i) {
19
             c[i].input();
20
             p = p + c[i].p;
        }
21
22
        p = p / 3.0;
23
        while(dis > eps) {
            now = calc(p);
24
             int best = -1;
25
             for(int i = 0; i < 4; ++ i) {
26
                 t = calc(p + Point(dis * dx[i], dis * dy[i]));
27
                 if(now > t) {
28
29
                      now = t;
                      best = i;
30
31
                 }
32
33
             if(best == -1) dis = 0.7 * dis;
34
             else p = p + Point(dis * dx[best], dis * dy[best]);
35
36
37
        if(calc(p) < eps) p.output();</pre>
38 }
```

6 图论

```
1
2
         x=w/l \rightarrow w-l*x =0
3
         f(x) = w-1*x;
4
5
         将边权更改为w-l*x来求生成树
6
7
        因为f(x)是个单调递减函数,随着x的增大而减少
8
        对于任意一个生成树如果
9
         f(x)>0
                        l需要增大
         f(x)<0
                    否则 1需要减小
10
11
        若要满足f(x)==0恒成立
12
        1. 若要x取最大值,则不能存在任意一个生成树f(x)>0,否则x还能继续增大,即任意生成树f(x)<=0
13
          若存在一个生成树f(x)>0,则那个生成树的比率一定大于当前x,w/l > x -> w-l*x > 0
14
        2. 若要x取最小值,则不能存在任意一个生成树f(x)<0,否则x还能继续减小,即任意生成树f(x)>=0
15
          若存在一个生成树f(x)<0,则那个生成树的比率一定小于当前x,w/1 < x -> w-1*x < 0
16
17
18
        若要满足f(x)>0恒成立,则最小生成树>0
        若要满足f(x)<0恒成立,则最大生成树<0
19
20
         此题目求解最小的x值,也就是检查是否所有的生成树f(x)>=0,即最小生成树>=0
21
22
23
         如果最小生成树大于0,所有的生成树都满足f(x)>0,尝试增加x得到f(x)=0
         否则,有生成树不满足这个条件,那么x一定要减少来使所有f(x)>=0
24
25
26
  */
```

6.1 DAG 图上建支配树

支配树概念:比如说我们要到a节点必须经过b节点,那么b节点就是a的支配点,在一棵支配树下,所有节点都是子树节点的支配点。
 vector<int> G1[N], G2[N];
 int in[N], n, m, q[N], idx;

```
int f[N][25], dep[N];
4
5
   void tupo() {
6
        idx = 0; queue<int> que;
7
        for (int i = 1; i <= n; i++) {
8
            if (!in[i]) que.push(i);
9
            if (!G1[i].size()) G1[i].push_back(0), in[0]++;
10
11
        while (que.size()) {
12
            int now = que.front();
13
            que.pop();
            q[++idx] = now;
14
            for (auto v : G1[now]) {
15
16
                in[v]--
                if (!in[v]) que.push(v);
17
            }
18
        }
19
20
   }
   int lca(int x, int y) {
21
        if (dep[x] > dep[y]) swap(x, y);
22
        for (int i = 20; i >= 0; i--) if (dep[f[y][i]] >= dep[x]) y = f[y][i];
23
24
25
        if (x == y) return x;
        for (int i = 20; i >= 0; i--) {
26
```

```
if (f[y][i] != f[x][i])
27
28
                y = f[y][i], x = f[x][i];
29
       return f[x][0];
30
31 }
32
   void solve() {
33
       int T; cin >> T; while (T--) {
34
35
            cin >> n >> m;
            for (int i = 0; i <= n; i++) G1[i].clear(), G2[i].clear(), dep[i] = 0;</pre>
36
37
            for (int i = 1; i \le m; i++) {
38
                int u, v; cin >> u >> v;
39
                G1[u].push_back(v);
                in[v]++;
40
            }
41
            tupo();
42
            for (int i = n; i >= 1; i--) {
43
                int now = q[i];
44
                if (!G1[now].size()) continue;
45
                int _lca = G1[now][0];
46
                for (int j = 1; j < G1[now].size(); j++) _lca = lca(_lca, G1[now][j]);</pre>
47
                f[now][0] = _lca;
48
                dep[now] = dep[\_lca] + 1;
49
50
                for (int j = 1; j \le 20; j++) f[now][j] = f[f[now][j-1]][j-1];
51
            int qry; cin >> qry; while (qry--) {
52
                int x, y; cin >> x >> y;
53
                cout << dep[x] + dep[y] - dep[lca(x, y)] << endl;
54
            }
55
56
       }
57 }
   6.2 spfa 判负环(01 分数规划)
1
2 //bfs版
3 #include <bits/stdc++.h>
4 using namespace std;
5 double eps = 1e-4;
6 const int N = 1e3 + 5;
7 const int M = 5e3 + 5;
   struct edge {
8
9
       int to, next;
       double vi;
10
11 }e[M * 2];
12
int vis[N], n, cnt, m, h[N], times[N];
14 double dis[N];
15 int ve[N];
16
   void add(int u, int v, double w) {
17
       e[cnt].to = v;
18
19
       e[cnt].vi = w;
20
       e[cnt].next = h[u];
21
       h[u] = cnt++;
22
   bool spfa(double mid) {
23
24
       queue<int> q;
```

```
for (int i = 1; i <= n; i++) {
25
26
            q.push(i);
            vis[i] = 1, dis[i] = 0, times[i] = 0;
27
28
29
        while (q.size()) {
            int u = q.front();
30
            q.pop();
31
            vis[u] = 0;
32
            for (int i = h[u]; \sim i; i = e[i].next) {
33
                int v = e[i].to;
34
                double va = mid * e[i].vi - (double)ve[u];
35
36
                if (dis[v] > dis[u] + va) {
                     dis[v] = dis[u] + va;
37
                     times[v] = times[u] + 1;
38
                     if (times[v] >= n) return true;
39
                     if (!vis[v]) {
40
41
                         q.push(v);
42
                         vis[v] = 1;
                     }
43
                }
44
            }
45
46
        return false;
47
48 }
49
50 //dfs版
51 #include <bits/stdc++.h>
52 using namespace std;
53 double eps = 1e-9;
54 const int N = 3e3 + 5;
   const int M = 1e4 + 5;
56
   struct edge {
57
        int to, next;
        double vi;
58
   }e[M << 1];
59
60
   int vis[N], n, cnt, m, h[N], times[N], flag;
62
   double dis[N];
63
   void add(int u, int v, double w) {
64
65
        e[cnt].to = v;
        e[cnt].vi = w;
66
        e[cnt].next = h[u];
67
68
        h[u] = cnt++;
69
   }
70
   void spfa(int u, double mid) {
71
        vis[u] = 1;
72
        for (int i = h[u]; ~i; i = e[i].next) {
            int v = e[i].to;
73
74
            double tmp = dis[u] + e[i].vi - mid;
75
            if (dis[v] > tmp) {
76
                dis[v] = tmp;
77
                if (vis[v]) {
78
                     flag = 1;
79
                     return;
80
81
                spfa(v, mid);
82
            }
        }
83
```

```
vis[u] = 0;
84
    }
85
86
    bool check(double mid) {
87
        memset(vis, 0, sizeof vis);
88
        memset(times, 0, sizeof times);
89
        for (int i = 1; i <= n; i++) dis[i] = 0;</pre>
90
        for (int i = 1; i <= n; i++) {
91
             flag = 0;
92
93
             spfa(i, mid);
             if (flag) return true;
94
95
96
        return false;
    }
97
98
    int main() {
99
        scanf("%d %d", &n, &m);
100
        memset(h, -1, sizeof h);
101
102
        for (int i = 1; i <= m; i++) {
103
             int x, y;
104
             double z;
105
             scanf("%d %d %lf", &x, &y, &z);
106
             add(x, y, z);
107
108
        }
109
        double l = -1e5, r = 1e5, mid;
110
        while (r - l > eps) {
111
             mid = (l + r) * 0.5;
112
             if (check(mid)) r = mid;
113
114
             else l = mid;
115
116
        printf("%.8lf", mid);
117 }
         边双连通分量 +e-dcc 缩点
    const int N = 5e5 + 10, M = 5e5 + 10, INF = 0x3f3f3f3f;
    const int MOD = 1e9 + 7;
 3
    int in[N];
 4 int ans;
    struct E_DCC {
 5
         int dfn[N], low[N], bridge[M], h1[N], h2[N], belong[N];
 6
 7
        int cnt1, cnt2, idx, scc, num;
 8
        struct Edge {
 9
             int to, next;
        } e1[N << 1], e2[N << 1];</pre>
10
11
        void init(int n, int m) {
12
             idx = cnt1 = cnt2 = 0;
13
             for (int i = 0; i <= n; i++) h1[i] = h2[i] = -1;
14
             for (int i = 0; i <= n; i++) belong[i] = low[i] = dfn[i] = 0;
15
16
             for (int i = 0; i <= 2*m; i++) bridge[i] = 0;</pre>
17
        }
18
        void add(int u, int v) {
19
             e1[cnt1].to = v;
20
             e1[cnt1].next = h1[u];
21
```

```
h1[u] = cnt1++;
22
23
       }
24
       void add_cc(int u, int v) {
25
26
            e2[cnt2].to = v;
27
            e2[cnt2].next = h2[u];
28
           h2[u] = cnt2++;
29
       }
30
       void tarjan(int u, int in_edge) {
31
32
            dfn[u] = low[u] = ++idx;
33
            for (int i = h1[u]; ~i; i = e1[i].next) {
                int v = e1[i].to;
34
                if (!dfn[v]) {
35
                    tarjan(v, i);
36
                    low[u] = min(low[u], low[v]);
37
                    if (low[v] > dfn[u]) num++, bridge[i] = bridge[i ^ 1] = 1;
38
                } else if (i != (in_edge ^ 1))
39
                    low[u] = min(low[u], dfn[v]);
40
           }
41
       }
42
43
       void rebuild(int x) {
44
45
           belong[x] = scc;
46
            for (int i = h1[x]; \sim i; i = e1[i].next) {
                int v = e1[i].to;
47
                if (belong[v] || bridge[i]) continue;
48
                rebuild(v);
49
            }
50
51
52
   }cc;
   void solve() {
53
       cc.init(n, m);
54
       for (int i = 1; i <= m; i++) {
55
            int u, v; cin >> u >> v;
56
            cc.add(u, v), cc.add(v, u);
57
58
       }
59
       for (int i = 1; i \le n; i++) if (!cc.dfn[i]) cc.tarjan(i, 0);
       for (int i = 1; i <= n; i++) {
60
            if (!cc.belong[i]) {
61
62
                ++cc.scc; cc.rebuild(i);
63
64
       for (int i = 0; i < cc.cnt1; i+=2) {
65
            int u = cc.e1[i].to, v = cc.e1[i^1].to;
66
67
            if (cc.belong[u] == cc.belong[v]) continue;
            cc.add_cc(cc.belong[u], cc.belong[v]), cc.add_cc(cc.belong[v], cc.belong[u]);
68
            in[cc.belong[u]]++, in[cc.belong[v]]++;
69
       }
70
71
  }
   6.4 差分约束系统
1
   1.求s[n] - s[1]的最小值时, 转换为1到n的最长路模型
2
3
   2. 求s[n] - s[1]的最大值时, 转换为1到n的最短路模型
4
5
```

```
ep:
   求最短路时
   1. xi + T >= xj --> i 向 j 建一条长度为T的边
9 2. xi + T > xj --> xi + T-1 >= xj i 向 j 建一条长度为T-1的边 10 3. xi + T == xj --> x + T >= xj && x + T <= xj (同(1))
11
12
   求最长路时,将 >= 换成 <= 即可
13
         点双连通分量 +v-dcc 缩点
   6.5
1 #include <bits/stdc++.h>
  using namespace std;
   const int N = 100010;
   int dfn[N], low[N], idx, cut[N], cnt, stk[N], tp;
   vector<int> g[N], dcc[N];
   int n, m;
   void tarjan(int u, int root) {
7
        dfn[u] = low[u] = ++idx;
8
9
        stk[++tp] = u;
        if (u == root && g[u].empty()) {
10
            dcc[++cnt].push_back(u);
11
            return;
12
        }
13
        int child = 0;
14
        for (auto v : g[u]) {
15
16
            if (!dfn[v]) {
17
                 tarjan(v, root);
18
                 low[u] = min(low[u], low[v]);
                 if (low[v] >= dfn[u]) {
19
20
                     child++;
21
                     if (child > 1 || u != root) cut[u] = 1;
22
                     cnt++;
23
                     int y;
                     while (y = stk[tp--]) {
24
                         dcc[cnt].push_back(y);
25
26
                         if (y == v) break;
27
28
                     dcc[cnt].push_back(u);
29
                 }
            }
30
31
            else
32
                 low[u] = min(low[u], dfn[v]);
33
        }
34
   }
35
   int main() {
36
37
        cin >> n >> m;
        for (int i = 0; i < m; i++) {
38
39
            int x, y;
40
            cin >> x >> y;
            g[x].push_back(y);
41
42
            g[y].push_back(x);
43
        }
        for (int i = 1; i <= n; i++) if (!dfn[i]) tarjan(i, i);</pre>
44
        for (int i = 1; i <= n; i++) {
45
            for (auto x : dcc[i]) cout << x << " ";</pre>
46
            cout << endl;</pre>
47
```

```
}
48
   }
49
50
51
   9 11
52
53 1 2
  1 5
54
55
  2 3
56
  2 5
   3 4
57
   4 5
   1 6
60
  6 7
61 9 8
62 6 9
63 6 8
64 */
```

6.6 二分图 HK 算法

```
1 #include <iostream>
2 #include <cstring>
3 #include <cstdio>
4 #include <cmath>
5 #include <algorithm>
6 #include <queue>
7 #include <stack>
8 #include <string>
9 #include <set>
10 #include <map>
11 #include <bitset>
12 using namespace std;
13 #define ACM_LOCAL
14
15 const int N = 3e3 + 5;
16 const int INF = 0x3f3f3f3f;
17 typedef pair<int, int> PII;
18 typedef long long ll;
19 typedef unsigned long long ull;
20 typedef long double ld;
21
22 int h[N], cnt, n; //存图
  int dx[N], dy[N], dis;
                                        //左边部分距离,右边部分距离,记录右边部分没有被匹配过的
      点的最大距离
   int machx[N], machy[N];
                                    //左边部分点匹配右边点,左边部分的点个数,右边部分点匹配左边的
   bool vist[N];
25
26
27
  struct edge {
28
       int to, next;
29 }e[N * N];
30
31
  void add(int u, int v) {
       e[cnt].to = v;
32
       e[cnt].next = h[u];
33
       h[u] = cnt++;
34
   }
35
36
```

```
bool searchpath() {//找有没有增广路
37
38
        queue<int>q;
        dis = INF;
39
        memset(dx, -1, sizeof(dx));
40
        memset(dy, -1, sizeof(dy));
41
42
        for (int i = 1; i <= n; i++)
43
            if (machx[i] == -1)
44
                q.push(i), dx[i] = 0;
45
46
        while (!q.empty()) {
47
48
            int u = q.front(); q.pop();
            if (dx[u] > dis)
49
50
                break;
            for (int i = h[u]; i != -1; i = e[i].next) {
51
52
                int v = e[i].to;
                if (dy[v] == -1) {
53
                     dy[v] = dx[u] + 1;
54
                     if (machy[v] == -1)
55
                         dis = dy[v];
56
                     else {
57
                         dx[machy[v]] = dy[v] + 1;
58
                         q.push(machy[v]);
59
60
                     }
61
                }
            }
62
63
        return dis != INF;
64
65
   bool findroad(int u) {
66
        for (int i = h[u]; i != -1; i = e[i].next) {
67
            int v = e[i].to;
68
            if (!vist[v] && dy[v] == dx[u] + 1) {
69
70
                vist[v] = 1;
                if (machy[v] != -1 \&\& dy[v] == dis)
71
72
                     continue;
73
                if (machy[v] == -1 || findroad(machy[v])) {
74
                    machy[v] = u; machx[u] = v; return true;
                }
75
76
            }
77
        }
        return false;
78
   }
79
80
   int MaxMatch() {
        int ans = 0;
81
82
        memset(machx, -1, sizeof(machx));
83
        memset(machy, -1, sizeof(machy));
        while (searchpath()) {
84
            memset(vist, 0, sizeof(vist));
85
86
            for (int i = 1; i <= n; i++)
87
                if (machx[i] == -1)
88
                     ans += findroad(i);
89
90
        return ans;
   }
91
```

6.7 二分图 KM 算法

```
1 #include <iostream>
2 #include <cstring>
3 #include <cstdio>
 4 #include <cmath>
5 #include <algorithm>
6 #include <queue>
7 #include <stack>
8 #include <string>
9 #include <set>
10 #include <map>
11 #include <bitset>
12 using namespace std;
13 #define ACM_LOCAL
14
15 const int INF = 0x3f3f3f3f;
typedef pair<int, int> PII;
typedef long long ll;
18 typedef unsigned long long ull;
19 typedef long double ld;
20 const int N = 310;
int n, m, match[N], pre[N];
22 bool vis[N];
23 ll g[N][N];
24 ll val1[N], val2[N], slack[N];
25
   void bfs(int p){
26
        memset(pre, 0, sizeof pre);
27
        memset(slack, 0x3f, sizeof slack);
28
        match[0] = p;
29
        int x = 0, nex = 0;
30
        do{
31
32
            vis[x] = true;
33
            int y = match[x];
            11 d = INF;
34
            for (int i = 1; i <= m; i++)</pre>
35
                if (!vis[i]){
36
                     if (slack[i] > val1[y] + val2[i] - g[y][i])
37
38
                         slack[i] = val1[y] + val2[i] - g[y][i], pre[i] = x;
                     if (slack[i] < d)</pre>
39
40
                         d = slack[i], nex = i;
                }
41
            for (int i = 0; i <= m; i++){
42
43
                if (vis[i])
44
                     val1[match[i]] -= d, val2[i] += d;
                else
45
                     slack[i] -= d;
46
47
            }
            x = nex;
48
        } while (match[x]);
49
50
        while (x){
51
            match[x] = match[pre[x]];
52
            x = pre[x];
53
        }
   }
54
55
   11 KM(){
56
        memset(match, 0, sizeof match);
57
58
        memset(val1, 0, sizeof val1);
        memset(val2, 0, sizeof val2);
59
```

```
for (int i = 1; i <= n; i++){
60
            memset(vis, false, sizeof vis);
61
            bfs(i);
62
63
        11 \text{ res} = 0;
64
        for (int i = 1; i <= m; i++) res += g[match[i]][i];</pre>
65
66
        return res;
67 }
   6.8 二分图匈牙利算法
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 10010;
 4 int match[N], vis[N];
5 int n, m, ans;
   vector<int> g[N];
7
8
   bool dfs(int x) {
        for (auto v : g[x]) {
9
            if (!vis[v]) {
10
11
                vis[v] = 1;
                if (!match[v] || dfs(match[v])) {
12
13
                    match[v] = x;
14
                    return true;
15
                }
16
            }
17
        return false;
18
19
   }
   6.9 割边
1 #include<bits/stdc++.h>
2 using namespace std;
3 \quad const int N = 100010;
4 int dfn[N], low[N], idx;
5 int n, m, ans;
6
   vector<int> g[N];
   void tarjan(int u, int fa) {
7
        dfn[u] = low[u] = ++idx;
8
        for (auto v : g[u]) {
9
            if(v == fa) continue;
10
            if (!dfn[v]) {
11
                tarjan(v, u);
12
                low[u] = min(low[u], low[v]);
13
                if (low[v] > dfn[u]) ans++;
14
            }
15
            else
16
                low[u] = min(low[u], dfn[v]);
17
        }
18
   }
19
20
   int main() {
21
22
        cin >> n >> m;
        for (int i = 0; i < m; i++) {
23
24
            int x, y;
```

```
25
            cin >> x >> y;
            g[x].push_back(y);
26
27
            g[y].push_back(x);
       }
28
29
30
       tarjan(1, 0);
31
       cout << m - ans;</pre>
32 }
   6.10 割点
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 100010;
 4 int dfn[N], low[N], idx, cut[N], siz[N], n, m;
5 vector<int> g[N];
6 int n, m;
   void tarjan(int u, int root) {
7
       dfn[u] = low[u] = ++idx;
8
9
       int child = 0;
       for (auto v : g[u]) {
10
            if (!dfn[v]) {
11
12
                tarjan(v, root);
13
                siz[u] += siz[v];
                low[u] = min(low[u], low[v]);
14
                if (low[v] >= dfn[u]) {
15
                    child++;
16
                    if (child > 1 || u != root) cut[u] = 1;
17
                }
18
19
            }
20
            else
21
                low[u] = min(low[u], dfn[v]);
22
       }
   }
23
   6.11
          克鲁斯卡尔重构树
1 #include <bits/stdc++.h>
2 using namespace std;
3 \quad const int N = 1e5 + 10;
4
   const int M = 2e5 + 10;
5
   struct Edge {
       int u, v, w;
6
7
       bool operator < (const Edge &rhs) const {</pre>
8
            return w > rhs.w;
9
            // > 最大边中的最小值
10
            // < 最小边中的最大值
11
   } e[M];
  int n, m, son[N], dep[N], pre[N], siz[N], top[N], tot, q, rnk[N], dfn[N], fat[N];
14 int cnt, val[N], fa[N];
   int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }
15
16
17
   vector<int> g[N];
18
   void dfs1(int u, int fa) {
19
       son[u] = -1; siz[u] = 1; dep[u] = dep[fa] + 1; fat[u] = fa;
20
```

```
for (auto v : g[u]) {
21
22
            if (v == fa) continue;
23
            dfs1(v, u);
            siz[u] += siz[v];
24
            if (son[u] == -1 \mid | siz[v] > siz[son[u]]) son[u] = v;
25
26
        }
27
   }
28
29
   void dfs2(int u, int t) {
        rnk[dfn[u] = ++tot] = u; top[u] = t;
30
31
        if (son[u] == -1) return;
32
        dfs2(son[u], t);
33
        for (auto v : g[u]) {
            if (v != son[u] && v != fat[u]) dfs2(v, v);
34
35
   }
36
37
   int lca(int u, int v) {
38
        while (top[u] != top[v]) {
39
40
            if (dep[top[u]] > dep[top[v]])
                u = pre[top[u]];
41
42
            else
                v = pre[top[v]];
43
44
45
        return dep[u] > dep[v] ? v : u;
   }
46
47
   void exKruskal() {
48
        cnt = n; for (int i = 1; i < (n << 1); i++) fa[i] = i;
49
        sort(e + 1, e + 1 + m);
50
        for (int i = 1; i <= m; i++) {
51
            int u = find(e[i].u), v = find(e[i].v);
52
53
            if (u == v) continue;
            val[++cnt] = e[i].w;
54
            fa[u] = fa[v] = cnt;
55
            g[u].push_back(cnt), g[cnt].push_back(u);
56
57
            g[v].push_back(cnt), g[cnt].push_back(v);
58
            if (cnt == (n << 1) - 1) break;//最多2N-1个点
59
        for (int i = 1; i <= cnt; i++)</pre>
60
            if (!siz[i]) {//未访问过
61
                int rt = find(i);//下树剖lca
62
                dfs1(rt, 0); dfs2(rt, rt);
63
64
            }
65
   }
```

6.12 克鲁斯卡尔重构树上建主席树

```
1 #include <bits/stdc++.h>
2 #define ACM_LOCAL
3 using namespace std;
4 typedef long long ll;
5 const int INF = 0x3f3f3f3f;
6 const int N = 2e5 + 10;
7 const int M = 5e5 + 10;
8 const int MOD = 1e9 + 7;
9
10 struct Hash {
```

```
int b[N], tot;
11
        void init() { tot = 0; }
void insert(int x) { b[++tot] = x; }
12
13
        void build() {
14
            sort(b + 1, b + 1 + tot);
15
16
            tot = unique(b + 1, b + 1 + tot) - (b + 1);
17
        int pos(int x) { return lower_bound(b + 1, b + 1 + tot, x) - b; }
18
   } ha;
19
   struct Edge {
        int u, v, w;
21
22
        bool operator < (const Edge &rhs) const {</pre>
23
            return w < rhs.w;</pre>
24
            // > 最大边中的最小值
            // < 最小边中的最大值
25
26
   } e[M];
27
28
29 int n, m, d[N], dfn[N], t, f[N][20], q, idx, rnk[N], in[N], out[N];
30 int cnt, val[N], fa[N], a[N];
31 int rt[N], NodeNum;
32 int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }
33
34 vector<int> g[N];
35
36
   void dfs(int x, int fa) {
        rnk[dfn[x] = ++idx] = x, f[x][0] = fa, d[x] = d[fa] + 1;
37
        for (int i = 1; i \leftarrow 19; i++) f[x][i] = f[f[x][i-1]][i-1];
38
        in[x] = idx;
39
40
        for (auto &y : g[x])
41
            if (y != fa) dfs(y, x);
        out[x] = idx;
42
   }
43
44
   void exKruskal() {
45
        cnt = n; for (int i = 1; i < (n << 1); i++) fa[i] = i;
46
47
        sort(e + 1, e + 1 + m);
        for (int i = 1; i <= m; i++) {</pre>
48
            int u = find(e[i].u), v = find(e[i].v);
49
            if (u == v) continue;
50
            val[++cnt] = e[i].w;
51
            fa[u] = fa[v] = cnt;
52
            g[u].push_back(cnt), g[cnt].push_back(u);
53
54
            g[v].push_back(cnt), g[cnt].push_back(v);
            if (cnt == (n << 1) - 1) break;//最多2N-1个点
55
56
57
        for (int i = 1; i <= cnt; i++)
            if (!d[i]) {//未访问过
58
                int rt = find(i);//下树剖lca
59
60
                dfs(rt, 0);
61
            }
62
63
   int FindPoint (int x, int p) {
        for (int i = 20; i >= 0; i--)
64
            if (d[x] > (1 << i) && val[f[x][i]] <= p) x = f[x][i];
65
66
        return x;
67
   }
68
69 struct {
```

```
int t[N << 5], lc[N << 5], rc[N << 5];</pre>
70
        int update(int pre, int 1, int r, int x) {
71
72
             int num = ++NodeNum;
             lc[num] = lc[pre], rc[num] = rc[pre], t[num] = t[pre] + 1;
73
74
             if (l != r) {
                 int mid = (l + r) \gg 1;
75
76
                 if (x <= mid) lc[num] = update(lc[pre], l, mid, x);</pre>
                 else rc[num] = update(rc[pre], mid + 1, r, x);
77
             }
78
79
             return num;
80
81
        int query(int pre, int now, int l, int r, int k) {
             if (t[now] - t[pre] < k) return -1;//如果之间的数少于k个, 返回-1
82
             k = t[now] - t[pre] - k + 1; // 这里主席树写的是第<math>k小,转换一下变成第k大
83
             while (l < r) {
84
                 int sum = t[lc[now]] - t[lc[pre]];
85
86
                 if (k \le sum) {
87
                     now = lc[now], pre = lc[pre];
                     r = 1 + r >> 1;
88
                 }
89
                 else {
90
                     now = rc[now], pre = rc[pre];
91
                     l = (l + r >> 1) + 1;
92
93
                     k -= sum;
94
                 }
             }
95
96
             return ha.b[l];
97
    }hjt;
98
99
    int ask(int x, int p, int k) {
100
         int pos = FindPoint(x, p);
101
        return hjt.query(rt[in[pos]-1], rt[out[pos]], 1, ha.tot, k);
102
    }
103
104
    void solve() {
105
106
        cin >> n >> m >> q;
107
         for (int i = 1; i <= n; i++) cin >> a[i], ha.insert(a[i]);
        ha.build();
108
109
        for (int i = 1; i <= m; i++)
             cin >> e[i].u >> e[i].v >> e[i].w;
110
        exKruskal();
111
        for (int i = 1; i <= idx; i++) {
112
113
             rt[i] = rt[i-1];
             if (rnk[i] \ll n)
114
                 rt[i] = hjt.update(rt[i-1], 1, ha.tot, ha.pos(a[rnk[i]]));
115
        }
116
117
        while (q--) {
118
119
             int v, x, k;
120
             cin >> v >> x >> k;
121
             printf("%d\n", ask(v, x, k));
122
        }
123
    }
124
125
    int main() {
126
        ios_base::sync_with_stdio(false);
        cin.tie(0);
127
128
        cout.tie(0);
```

```
129 #ifdef ACM_LOCAL
         freopen("input", "r", stdin);
freopen("output", "w", stdout);
130
131
    #endif
132
133
         solve();
134 }
    6.13
            欧拉图
 1 #include <bits/stdc++.h>
 2 using namespace std;
 3 const int N = 1e5 + 10;
 4 \text{ const int } M = 2e5 + 10;
 5
    struct edge {
 6
 7
         int to, next;
    }e[M<<1];
 8
 9
   int n, m, t, stk[N], tp, cnt, h[N], vis[M<<1], ans[N];</pre>
10
11
    void add(int u, int v) {
12
         e[cnt].to = v;
13
         e[cnt].next = h[u];
14
         h[u] = cnt++;
15
    }
16
17
    void euler() {
18
19
         stk[++tp] = 1;
20
         while (tp > 0) {
             int x = stk[tp], i = h[x];
21
22
             while (i && vis[i]) i = e[i].next;
23
             if (i) {
24
                  stk[++tp] = e[i].to;
25
                  vis[i] = vis[i ^ 1] = 1;
                  h[x] = e[i].next;
26
27
             }
             else {
28
29
                  tp--:
30
                  ans[++t] = x;
31
             }
32
         }
    }
33
34
    int main() {
35
36
         cin >> n >> m;
37
         for (int i = 0; i < m; i++) {
38
             int x, y;
             cin >> x >> y;
39
40
             add(x, y), add(y, x);
         }
41
         euler();
42
         for (int i = t; i >= 1; i--) cout << ans[i] << " ";
43
    }
44
45
46 /*
47 7 9
48 1 2
49 1 3
```

```
50 1 4
51 1 5
52 2 3
53 4 5
54 5 6
55 6 7
56 5 7
57 */
```

6.14 强联通分量

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 10010;
4 int h[N], cnt, n, m, scc;
   int dfn[N], low[N], idx, tp, in_stk[N], vis[N], sd[N];
   set<int> sc[N];
   struct edge{
7
8
        int to, next;
9
   }e[N*2];
10
   void add(int u, int v) {
11
12
        e[cnt].to = v;
        e[cnt].next = h[u];
13
        h[u] = cnt++;
14
15
   }
16
17
   void tarjan(int x) {
        low[x] = dfn[x] = ++idx;
18
        vis[x] = 1;
19
        in\_stk[++tp] = x;
20
21
        for (int i = h[x]; \sim i; i = e[i].next) {
22
            int v = e[i].to;
            if (!dfn[v]) {
23
24
                tarjan(v);
25
                low[x] = min(low[x], low[v]);
26
            else if (vis[v]) {
27
28
                low[x] = min(low[x], dfn[v]);
29
30
        if (low[x] == dfn[x]) {
31
32
            int y;
33
            ++SCC;
            while(y = in_stk[tp--]) {
34
35
                sd[y] = scc;
36
                vis[y] = 0;
                sc[scc].insert(y);
37
                if (x == y) break;
38
            }
39
        }
40
   }
41
42
43
   int main() {
44
        cin >> n >> m;
        memset(h, -1, sizeof h);
45
        for (int i = 0; i < m; i ++) {
46
            int x, y;
47
```

```
48
            cin >> x >> y;
49
           add(x, y);
50
       for (int i = 1; i <= n; i ++) if (!dfn[i]) tarjan(i);</pre>
51
52
       for (int i = 1; i <= scc; i ++) {
53
            cout << "#" << i << ":";
54
            for (auto x : sc[i]) {
55
                cout << x << "
56
            }
57
            cout << endl;</pre>
58
59
       }
  }
60
          限制流量费用流 (Dijkstra)
   6.15
   struct Edge {
3
       int to;
       int cap, dis; //容量、费用
4
       int rev;
                      //(u,v)的反向弧中,v在u的位置
5
6
       Edge() {}
       Edge(int to, int cap, int dis, int rev): to(to), cap(cap), dis(dis), rev(rev) {}
7
   };
8
   vector<Edge> G[N];
   struct Pre { //记录前驱
10
       int node; //前驱结点
11
12
       int edge; //对应的边
13
   } pre[N];
   int h[N];
                //势能函数
   ll dis[N]; //费用
16
   void addEdge(int x, int y, int cap, int cost) {
17
       G[x].push_back(Edge(y, cap, cost, (int)G[y].size()));
                                                                    //正向边
18
       G[y].push_back(Edge(x, 0, -cost, (int)(G[x].size() - 1))); //反向边
19
   bool Dijkstra(int S, int T) {
20
21
       memset(dis, 0x3f3f3f3f, sizeof dis);
22
       dis[S] = 0;
23
24
       priority_queue<PII, vector<PII>, greater<PII>> Q;
       0.push(PII(dis[S], S));
25
26
       while (!Q.empty()) {
27
            PII now = Q.top();
28
           Q.pop();
29
            int u = now.second;
30
            if (dis[u] < now.first)</pre>
31
32
                continue;
33
            for (int i = 0; i < G[u].size(); i++) {</pre>
34
35
                int v = G[u][i].to;
                int cap = G[u][i].cap;
36
                int w = G[u][i].dis;
37
38
                if (cap \&\& dis[v] > w + dis[u] + h[u] - h[v]) {
                    dis[v] = w + dis[u] + h[u] - h[v]; //进行松弛
39
                    pre[v].node = u;
40
                                                         //记录前驱点
41
                    pre[v].edge = i;
                                                        //记录前驱边
                    Q.push(PII(dis[v], v));
42
```

```
43
               }
           }
44
45
       if (dis[T] == INF)
46
           return false;
47
48
       else {
           for (int i = 0; i <= T + 1; i++) //对于势能函数, 每次加上当前轮的dis
49
               h[i] += dis[i];
50
           return true;
51
       }
52
   }
53
   void maxFlow(int S, int T, int &flow, ll &cost) {//flow 代表最大总流量
54
       memset(h, 0, sizeof(h));
55
       memset(pre, 0, sizeof(pre));
56
57
       int newFlow = 0;
                                         //增广流量
58
       while (flow && Dijkstra(S, T)) { //当无法增广时,即找到答案
59
           int minn = INF;
60
           for (int i = T; i != S; i = pre[i].node) {
61
               int node = pre[i].node;
62
               int edge = pre[i].edge;
63
               minn = min(minn, G[node][edge].cap);
64
           }
65
66
67
           flow -= minn;
                                 //原流量
           newFlow += minn;
68
                                 //增广流量
           cost += 1ll * h[T] * minn; //增广流花销
69
70
           for (int i = T; i != S; i = pre[i].node) {
71
                int node = pre[i].node;
72
               int edge = pre[i].edge;
73
               int rev = G[node][edge].rev;
74
75
               G[node][edge].cap -= minn;
               G[i][rev].cap += minn;
76
77
           }
       }
78
79
  }
   6.16 一般图最大匹配(带花树)
   struct Edge {
2
       int to, next;
3
   } e[M];
   int cnt, n, m, nn, h[N], fa[N], match[N], pre[N], tic[N], tim, ty[N];
   queue<int> que;
5
   void add(int u, int v) {
6
7
       e[cnt].to = v;
       e[cnt].next = h[u];
8
9
       h[u] = cnt++;
   }
10
11
   int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }
12
13
14
   int lca(int x, int y) {
       for (tim++;; swap(x, y))
15
           if (x) {
16
               x = find(x);
17
               if (tic[x] == tim) return x;
18
```

```
tic[x] = tim;
19
20
                x = pre[match[x]];
            }
21
   }
22
23
   void shrink(int x, int y, int p) {
24
25
        while (find(x) != p) {
26
            pre[x] = y;
27
            y = match[x];
            if (ty[y] == 2) ty[y] = 1, que.push(y);
28
29
            if (find(x) == x) fa[x] = p;
30
            if (find(y) == y) fa[y] = p;
31
            x = pre[y];
32
        }
   }
33
34
   bool aug(int s) {
35
        for (int i = 1; i \le n; i++) fa[i] = i, ty[i] = pre[i] = 0;
36
        while (!que.empty()) que.pop();
37
38
        que.push(s); ty[s] = 1;
        while (!que.empty()) {
39
            int x = que.front();
40
            que.pop();
41
42
            for (int i = h[x], y = e[i].to; \sim i; i = e[i].next, y = e[i].to) {
43
                if (find(x) == find(y) | | ty[y] == 2) continue;
                if (!ty[y]) {
44
                     ty[y] = 2;
45
                     pre[y] = x;
46
                     if (!match[y]) {
47
                         for (int tmp; y; y = tmp, x = pre[y])
48
                             tmp = match[x], match[x] = y, match[y] = x;
49
                         return 1;
50
                     } else ty[match[y]] = 1, que.push(match[y]);
51
                } else if (ty[y] == 1) {
52
                     int p = lca(x, y);
53
                     shrink(x, y, p);
54
55
                     shrink(y, x, p);
56
                }
            }
57
        }
58
59
        return 0;
   }
60
   int calc() {
61
62
        int ans = 0;
        for (int i = 1; i <= n; i++) {
63
64
            if (!match[i] && aug(i)) ans++;
65
        return ans;
66
  }
67
   void init() {
69
        for (int i = 1; i <= n; i++) {
70
            match[i] = tic[i] = 0;
71
            h[i] = -1;
72
73
        cnt = tim = 0;
74 }
```

6.17 最大独立集 (一般图)

```
1 const int N = 200 + 5;
2 int G[N][N];
3 int ans[N];
4 int vis[N];
5 int res, n;
   void dfs(int x, int cnt) {
6
7
       if (x > n) {
            if (cnt > res) {
8
                res = cnt;
9
                for (int i = 1; i <= n; i++)
10
                    ans[i] = vis[i];
11
12
            }
13
            return;
14
       if (cnt + n - x + 1 < res) return;
15
       int pd = 0;
16
17
       for (int i = 1; i < x; i++)
18
            if (vis[i] && G[i][x]) {
                pd = 1;
19
                break;
20
21
       if (!pd) {
22
            vis[x] = 1;
23
            dfs(x+1, cnt+1);
24
25
           vis[x] = 0;
26
27
       dfs(x+1, cnt);
28 }
   6.18 最大流 (Dinic)
1 typedef long long ll;
2 typedef pair<int, int> PII;
3 const int N = 1e3 + 10, M = 2e5 + 10, INF = 0x3f3f3f3f;
4 const int MOD = 1e9 + 7;
5 int state[N][11], n, p, cap[N], st, ed;
6
   struct Maxflow {
7
       int h[N], cnt, maxflow, deep[N], cur[N];
8
       struct Edge {
            int to, next;
9
10
            ll cap;
       } e[M<<1];</pre>
11
12
       void init() {
13
            memset(h, -1, sizeof h);
14
15
            cnt = maxflow = 0;
16
       void add(int u, int v, int cap) {
17
18
            e[cnt].to = v;
            e[cnt].cap = cap;
19
20
            e[cnt].next = h[u];
21
            h[u] = cnt++;
22
            e[cnt].to = u;
23
            e[cnt].cap = 0;
24
25
            e[cnt].next = h[v];
```

```
h[v] = cnt++;
26
27
        }
28
29
        bool bfs() {
30
            for (int i = 0; i \le ed; i++) deep[i] = -1, cur[i] = h[i];
31
32
            queue<int> q; q.push(st); deep[st] = 0;
            while (q.size()) {
33
                int u = q.front();
34
35
                q.pop();
36
                for (int i = h[u]; \sim i; i = e[i].next) {
37
                    int v = e[i].to;
                    if (e[i].cap \&\& deep[v] == -1) {
38
                        deep[v] = deep[u] + 1;
39
40
                        q.push(v);
                    }
41
                }
42
43
            }
            if (deep[ed] >= 0) return true;
44
            else return false;
45
        }
46
47
        ll dfs(int u, ll mx) {
48
49
            11 a;
50
            if (u == ed) return mx;
            for (int i = cur[u]; ~i; i = e[i].next) {
51
                cur[u] = i;//优化
52
                int v = e[i].to;
53
                if (e[i].cap \&\& deep[v] == deep[u] + 1 \&\& (a = dfs(v, min(e[i].cap, mx))))
54
       {
55
                    e[i].cap -= a;
                    e[i ^ 1].cap += a;
56
57
                    return a;
                }
58
            }
59
            return 0;
60
61
        }
62
        void dinic() {
63
            ll res;
64
            while (bfs()) {
65
                while (1) {
66
                    res = dfs(st, INF);
67
68
                    if (!res) break;
69
                    maxflow += res;
70
                }
71
            }
72
73
   }mf;
          最大团 (一般图)
   6.19
1 const int N = 200 + 5;
2 int n;
3 int G[N][N];
4 int cnt[N];//cnt[i]为>=i的最大团点数
5 int group[N];//最大团的点
6 int vis[N];//记录点的位置
```

```
int res;//最大团的数目
   bool dfs(int pos, int num) {//num为已取的点数
8
       for (int i = pos + 1; i \le n; i++) {
9
           if (cnt[i] + num <= res)//剪枝, 若取i但cnt[i]+已经取了的点数仍<ans
10
11
               return false;
12
           if (G[pos][i]) {//与当前团中元素比较,取Non-N(i)
13
               int j;
14
               for (j = 0; j < num; j++)
15
                   if (!G[i][vis[j]])
16
                       break;
17
18
               if (j == num) {//若为空,则皆与i相邻,则此时将i加入到最大团中
                   vis[num] = i;
19
                   if (dfs(i, num + 1))
20
21
                       return true;
               }
22
           }
23
       }
24
25
26
       if (num > res) {//每添加一个点最多使最大团数+1,后面的搜索就没有意义了
27
           res = num;//最大团中点的数目
28
           for (int i = 1; i <= num; i++)//最大团的元素
29
               group[i] = vis[i - 1];
30
           return true;
31
       return false;
32
33 }
34
   void maxClique() {
35
       res = -1;
36
       for (int i = n; i >= 1; i--) {//枚举所有点
37
38
           vis[0] = i;
           dfs(i, 1);
39
           cnt[i] = res;
40
       }
41
42 }
   6.20 最短路
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1e5 + 10;
   const int M = 5e5 + 10;
   typedef long long ll;
5
6
   struct Node {
7
       struct Edge {
8
           int to, next, w;
       }e[M<<1];
9
10
       int h[N], cnt, vis[N], count[N];
11
       ll dis[N];
12
       void init() {
13
           memset(h, -1, sizeof h);
14
15
           cnt = 0;
16
       void add(int u, int v, int w) {
17
           e[cnt].to = v;
18
19
           e[cnt].w = w;
```

```
e[cnt].next = h[u];
20
21
            h[u] = cnt++;
       }
22
23
24
       struct node {
            int now; ll d;
25
            bool operator < (const node &rhs) const {</pre>
26
27
                return d > rhs.d;
28
            }
       };
29
30
31
       void dij(int st) {
            memset(dis, 0x3f, sizeof dis);
32
            memset(vis, 0, sizeof vis);
33
            dis[st] = 0; priority_queue<node> q;
34
            q.push({st, dis[st]});
35
36
            while (q.size()) {
37
                int u = q.top().now;
38
                q.pop();
                if (vis[u]) continue;
39
                vis[u] = 1;
40
                for (int i = h[u]; ~i; i = e[i].next) {
41
                    int v = e[i].to;
42
43
                    if (dis[v] > dis[u] + e[i].w) {
44
                        dis[v] = dis[u] + e[i].w;
                        if (!vis[v]) {
45
                             q.push({v, dis[v]});
46
                        }
47
48
                    else if (dis[v] == dis[u] + e[i].w) {
49
50
                        count[v]++;
51
52
                }
53
           }
54
   }Dij;
55
   6.21
          最小费用最大流 (SPFA)
   typedef long long ll;
   typedef pair<int, int> PII;
3 const int N = 1e4 + 10, M = 2e5 + 10, INF = 0x3f3f3f3f;
   const int MOD = 1e9 + 7;
   int st, ed;
5
6
   struct node {
       int maxflow, mincost, cnt;
7
       int vis[N], dis[N], pre[N], last[N], h[N], flow[N];
8
       struct edge {
9
            int to, next, cap, cos;
10
       e[M << 1];
11
12
       void add(int u, int v, int cap, int cos) {
13
            e[cnt].to = v;
14
            e[cnt].cap = cap;
15
            e[cnt].cos = cos;
16
            e[cnt].next = h[u];
17
18
            h[u] = cnt++;
19
```

```
e[cnt].to = u;
20
            e[cnt].cap = 0;
21
            e[cnt].cos = -cos;
22
23
            e[cnt].next = h[v];
24
            h[v] = cnt++;
        }
25
26
        void init() {
27
            memset(h, -1, sizeof h);
28
29
            cnt = 0;
30
            mincost = maxflow = 0;
31
        bool spfa() {
32
            queue<int> q;
33
            for (int i = 0; i <= ed; i++) dis[i] = INF, vis[i] = 0;
34
            vis[st] = 1, dis[st] = 0, flow[st] = INF;
35
36
            q.push(st);
37
            while (q.size()) {
                int u = q.front();
38
                q.pop(); vis[u] = 0;
39
                for (int i = h[u]; ~i; i = e[i].next) {
40
                    int v = e[i].to;
41
                    if (e[i].cap && dis[v] > dis[u] + e[i].cos) {
42
43
                         dis[v] = e[i].cos + dis[u];
44
                         flow[v] = min(flow[u], e[i].cap);
                        pre[v] = u;
45
                        last[v] = i;
46
                        if (!vis[v]) {
47
                             vis[v] = 1;
48
49
                             q.push(v);
50
                        }
                    }
51
                }
52
53
            if (dis[ed] != INF) return true;
54
            else return false;
55
56
        }
57
        void MCMF() {
58
            while (spfa()) {
59
                int now = ed;
60
                maxflow += flow[ed];
61
                mincost += flow[ed] * dis[ed];
62
63
                while (st != now) {
64
                    e[last[now]].cap -= flow[ed];
65
                    e[last[now] ^ 1].cap += flow[ed];
66
                    now = pre[now];
67
                }
68
69
            }
70
   }mcmf;
   6.22
          最小路径生成树
1 #include <bits/stdc++.h>
  using namespace std;
```

 $3 \quad const int N = 100010;$

```
typedef long long ll;
   typedef pair<int, int> PII;
6
   struct Edge {
7
8
        int to, next, w;
   }e[N<<1];
9
10
11 int h[N], cnt;
   void add(int u, int v, int w) {
12
13
        e[cnt].to = v;
14
        e[cnt].w = w;
15
        e[cnt].next = h[u];
        h[u] = cnt++;
16
   }
17
   int n, m, k;
18
   struct Node {
19
        vector<PII> g[N];
20
21
        int vis[N];
        11 dis[N];
22
23
        void init() {
            for (int i = 1; i <= n; i++) g[i].clear();</pre>
24
25
26
27
        struct node {
28
            int now; ll d;
            bool operator < (const node &rhs) const {</pre>
29
                 return d > rhs.d;
30
            }
31
        };
32
33
        void dij(int st) {
34
35
            memset(dis, 0x3f, sizeof dis);
            memset(vis, 0, sizeof vis);
36
            dis[st] = 0; priority_queue<node> q;
37
            q.push({st, dis[st]});
38
            while (q.size()) {
39
40
                int u = q.top().now;
41
                q.pop();
                if (vis[u]) continue;
42
                vis[u] = 1;
43
                for (auto item : g[u]) {
44
                     int v = item.first;
45
                     int w = item.second;
46
                     if (dis[v] > dis[u] + w) {
47
                         dis[v] = dis[u] + w;
48
49
                         if (!vis[v]) {
50
                              q.push({v, dis[v]});
                         }
51
52
                     }
53
                }
54
            }
55
        }
   }Dij;
56
   int tap[N];
57
   void build_tree(int x) {
58
        tap[x] = 1;
59
        for (auto item : Dij.g[x]) {
60
61
            int v = item.first;
            int w = item.second;
62
```

```
if (tap[v]) continue;
63
            if (Dij.dis[v] == Dij.dis[x] + w) {
64
                 add(x, v, w), add(v, x, w);
65
66
                 build_tree(v);
            }
67
        }
68
   }
69
70
   int main() {
71
        scanf("%d %d %d", &n, &m, &k);
72
73
        Dij.init();
74
        for (int i = 1; i \le m; i++) {
            int u, v, w; scanf("%d %d %d", &u, &v, &w);
75
            Dij.g[u].push_back(PII{v, w});
76
            Dij.g[v].push_back(PII{u, w});
77
78
        Dij.dij(1);
79
80
        for (int i = 1; i <= n; i++)
            sort(Dij.g[i].begin(), Dij.g[i].end());
81
82
        memset(h, -1, sizeof h);
        build_tree(1);
83
84 }
           最小生成树 boruvka
1
   11 boruvka() {
2
        for (int i = 1; i <= n; i++) fa[i] = i;</pre>
        11 ans = 0, num = 0;
3
4
        while (num < n-1) {
            int tmp = 0;
5
            for (int i = 1; i <= n; i++) E[i] = PII{INF, INF};
for (int i = 1; i <= m; i++) {</pre>
6
7
8
                 int fx = find(e[i].u);
                 int fy = find(e[i].v);
9
                 if (fx == fy) continue;
10
                 tmp++;
11
12
                 E[fx] = min(E[fx], PII\{e[i].w, i\});
                 E[fy] = min(E[fy], PII\{e[i].w, i\});
13
14
15
            if (tmp == 0) break;
            for (int i = 1; i \le m; i++) {
16
                 int fx = find(e[i].u);
17
                 int fy = find(e[i].v);
18
                 if (fx == fy) continue;
19
                 if (E[fx] == PII\{e[i].w, i\} \mid \mid E[fy] == PII\{e[i].w, i\}) 
20
                     ans += e[i].w;
21
                     num++;
22
                     fa[fx] = fy;
23
                 }
24
            }
25
26
27
        if (num < n-1) return -1;
28
        else return ans;
29
   }
```

6.24 最小生成树 prim

```
int prim() {
        memset(vis, 0, sizeof vis);
2
        memset(dis, 0x3f, sizeof dis);
3
        int ans = 0;
4
        for (int i = h[1]; ~i; i = e[i].next) {
5
6
            dis[e[i].to] = min(dis[e[i].to], e[i].vi);
7
8
        dis[1] = 0, vis[1] = 1;
9
        for (int i = 2; i <= n; i++) {
            int minn = INF;
10
            int k;
11
            for (int j = 1; j \le n; j++) {
12
                if (!vis[j] && dis[j] < minn) {</pre>
13
                    minn = dis[j];
14
                    k = j;
15
                }
16
17
            if (minn == INF) return -1;
18
19
            ans += minn;
            vis[k] = 1;
20
21
            for (int j = h[k]; \sim j; j = e[j].next) {
22
                dis[e[j].to] = min(dis[e[j].to], e[j].vi);
23
24
25
        return ans;
26 }
```

7 杂项

```
vector<int> TreeToPrufer(int n) {// 1~n-2个数
 2
                vector<int> pru(n+1);
 3
                vector<int> in(n+1);
                vector<int> fa(n+1);
 4
                for (int i = 1; i < n; i++) read(fa[i]), ++in[fa[i]];</pre>
 5
                for (int i = 1, j = 1; i \le n - 2; i++, j++) {
 6
 7
                         while (in[j]) ++j; pru[i] = fa[j];
                         while (i \le n - 2 \& !--in[pru[i]] \& pru[i] < j) pru[i+1] = fa[pru[i]], ++i;
 8
 9
10
                return pru;
11 }
12
       vector<int> PruferToTree(int n) {// 1~n-1个数
13
                vector<int> fa(n+1);
14
                vector<int> in(n+1);
15
16
                vector<int> pru(n+1);
                for (int i = 1; i \le n - 2; i++) read(pru[i]), ++in[pru[i]]; pru[n-1] = n;
17
18
                for (int i = 1, j = 1; i < n; i++, j++) {
                          while (in[j]) ++j; fa[j] = pru[i];
19
                         while (i < n && !--in[pru[i]] && pru[i] < j) fa[pru[i]] = pru[i+1], ++i;</pre>
20
21
22
                return fa;
23 }
       7.1 DEBUG
       #define debug(a...) cout << "(" << (#a) << ")" << " = ("; DEBUG(a)
       template<typename T> void DEBUG(T value) {
                cout << value << ")" << endl;
 3
 4 }
       template<typename T1, typename... T2>
       void DEBUG(T1 now, T2... other) {
                cout << now << ", ", DEBUG(other...);</pre>
 7
 8
       }
       7.2 MODINT
       namespace MODINT {
 2
                 template<unsigned M_> struct ModInt {
 3
                          static constexpr unsigned M = M_;
 4
                          unsigned x;
                          constexpr ModInt() : x(0U) {}
 5
                          constexpr ModInt(unsigned x_{-}) : x(x_{-} % M) {}
 6
                          constexpr ModInt(unsigned long long x_-) : x(x_- % M) {}
 7
                          constexpr ModInt(int x_) : x(((x_ \%= static_cast < int > (M)) < \emptyset) ? (x_ + x_c < (M)) < (M)) ? (x_ + x_c <
 8
                static_cast<int>(M)) : x_) {}
                          constexpr ModInt(long long x_) : x(
 9
                                       ((x_ \%= static_cast< long long>(M)) < 0) ? (x_ + static_cast< long long>(M))
10
                ): x_) {}
                         ModInt &operator+=(const ModInt &a) {
11
                                  x = ((x += a.x) >= M) ? (x - M) : x;
12
                                  return *this;
13
14
                         ModInt &operator-=(const ModInt &a) {
15
                                  x = ((x -= a.x) >= M) ? (x + M) : x;
16
```

```
return *this;
17
18
            ModInt &operator*=(const ModInt &a) {
19
                x = (static_cast<unsigned long long>(x) * a.x) % M;
20
                return *this;
21
22
            ModInt &operator/=(const ModInt &a) { return (*this *= a.inv()); }
23
            ModInt quick_pow(long long e) const {
24
                if (e < 0) return inv().quick_pow(-e);</pre>
25
                ModInt a = *this, b = 1U;
26
27
                for (; e; e >>= 1) { if (e & 1) b *= a; a *= a; }
28
                return b;
29
            ModInt inv() const {
30
                unsigned a = M, b = x;
31
                int y = 0, z = 1;
32
                for (; b;) {
33
                    const unsigned q = a / b;
34
35
                     const unsigned c = a - q * b;
                    a = b; b = c;
36
                     const int w = y - static_cast<int>(q) * z;
37
38
                    y = Z; Z = W;
39
                assert(a == 10);
40
                return ModInt(y);
41
42
            ModInt operator+() const { return *this; }
43
            ModInt operator-() const { ModInt a; a.x = x ? (M - x) : 0U; return a; }
44
            ModInt operator+(const ModInt &a) const { return (ModInt(*this) += a); }
45
            ModInt operator-(const ModInt &a) const { return (ModInt(*this) -= a); ModInt operator*(const ModInt &a) const { return (ModInt(*this) *= a);
46
47
            ModInt operator/(const ModInt &a) const { return (ModInt(*this) /= a); }
48
            template<class T> friend ModInt operator+(T a, const ModInt &b) { return (
49
       ModInt(a) += b); }
            template<class T> friend ModInt operator-(T a, const ModInt &b) { return (
50
       ModInt(a) -= b); }
            template<class T> friend ModInt operator*(T a, const ModInt &b) { return (
51
       ModInt(a) *= b); }
            template<class T> friend ModInt operator/(T a, const ModInt &b) { return (
52
       ModInt(a) /= b); }
            explicit operator bool() const { return x; }
53
            bool operator==(const ModInt &a) const { return (x == a.x); }
54
            bool operator!=(const ModInt &a) const { return (x != a.x);
55
            friend std::ostream &operator<<(std::ostream &os, const ModInt &a) { return os
56
       << a.x; }
57
        };
        constexpr unsigned MO = 1000000007;
58
  //
          constexpr unsigned MO = 998244353;
59
        using Mint = ModInt<MO>;
60
61 }
   7.3
        大数模板
1 constexpr int base = 1000000000;
   constexpr int base_digits = 9;
3
  struct bigint {
       // value == 0 is represented by empty z
```

```
vector<int> z; // digits
6
7
        // sign == 1 <==> value >= 0
8
        // sign == -1 <==> value < 0
9
10
        int sign;
11
12
        bigint() : sign(1) {}
13
        bigint(ll v) { *this = v; }
14
15
        bigint &operator=(ll v) {
16
17
            sign = v < 0 ? -1 : 1;
            v *= sign;
18
            z.clear();
19
            for (; v > 0; v = v / base) z.push_back((int) (v % base));
20
            return *this;
21
        }
22
23
        bigint(const string &s) { read(s); }
24
25
        bigint &operator+=(const bigint &other) {
26
27
            if (sign == other.sign) {
                for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
28
29
                    if (i == z.size())
30
                         z.push_back(0);
                    z[i] += carry + (i < other.z.size() ? other.z[i] : 0);
31
                    carry = z[i] >= base;
32
                    if (carry)
33
                        z[i] -= base;
34
35
            } else if (other != 0 /* prevent infinite loop */) {
36
                *this -= -other;
37
38
            }
            return *this;
39
        }
40
41
42
        friend bigint operator+(bigint a, const bigint &b) { return a += b; }
43
        bigint &operator-=(const bigint &other) {
44
            if (sign == other.sign) {
45
                if (sign == 1 && *this >= other || sign == -1 && *this <= other) {
46
                    for (int i = 0, carry = 0; i < other.z.size() || carry; ++i) {
47
                        z[i] = carry + (i < other.z.size() ? other.z[i] : 0);
48
                         carry = z[i] < 0;
49
                        if (carry)
50
                             z[i] += base;
51
52
                    trim();
53
                } else {
54
                    *this = other - *this;
55
56
                    this->sign = -this->sign;
57
                }
58
            } else {
                *this += -other;
59
60
61
            return *this;
62
63
        friend bigint operator-(bigint a, const bigint &b) {
64
```

```
65
             return a -= b;
         }
66
67
         bigint &operator*=(int v) {
68
             if (v < 0) sign = -sign, v = -v;
69
             for (int i = 0, carry = 0; i < z.size() || carry; ++i) {
70
                 if (i == z.size()) z.push_back(0);
71
                 ll cur = (ll) z[i] * v + carry;
72
                 carry = (int) (cur / base);
73
74
                 z[i] = (int) (cur \% base);
75
             }
             trim();
76
             return *this;
77
         }
78
79
         bigint operator*(int v) const { return bigint(*this) *= v; }
80
81
         friend pair<br/>digint, bigint> divmod(const bigint &a1, const bigint &b1) {
82
             int norm = base / (b1.z.back() + 1);
83
             bigint a = a1.abs() * norm;
84
             bigint b = b1.abs() * norm;
85
             bigint q, r;
86
             q.z.resize(a.z.size());
87
88
89
             for (int i = (int) \ a.z.size() - 1; i >= 0; i--) {
90
                 r *= base;
                 r += a.z[i];
91
                 int s1 = b.z.size() < r.z.size() ? r.z[b.z.size()] : 0;</pre>
92
                 int s2 = b.z.size() - 1 < r.z.size() ? r.z[b.z.size() - 1] : 0;</pre>
93
                 int d = (int) (((ll) s1 * base + s2) / b.z.back());
94
                 r -= b * d;
95
                 while (r < 0) r += b, --d;
96
97
                 q.z[i] = d;
             }
98
99
             q.sign = a1.sign * b1.sign;
100
             r.sign = a1.sign;
101
102
             q.trim();
103
             r.trim();
             return {q, r / norm};
104
105
         }
106
         friend bigint sqrt(const bigint &a1) {
107
108
             bigint a = a1;
             while (a.z.empty() \parallel a.z.size() \% 2 == 1) a.z.push_back(0);
109
110
             int n = a.z.size();
111
112
             int firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
113
             int norm = base / (firstDigit + 1);
114
115
             a *= norm;
116
             a *= norm;
             while (a.z.empty() \mid | a.z.size() \% 2 == 1) a.z.push_back(0);
117
118
             bigint r = (ll) a.z[n - 1] * base + a.z[n - 2];
119
             firstDigit = (int) ::sqrt((double) a.z[n - 1] * base + a.z[n - 2]);
120
121
             int q = firstDigit;
122
             bigint res;
123
```

```
for (int j = n / 2 - 1; j >= 0; j--) {
124
                 for (;; --q) {
125
                      bigint r1 = (r - (res * 2 * base + q) * q) * base * base +
126
                                  (j > 0 ? (ll) a.z[2 * j - 1] * base + a.z[2 * j - 2] : 0);
127
128
                      if (r1 >= 0) {
129
                          r = r1;
                          break;
130
                      }
131
132
                 (res *= base) += q;
133
134
135
                 if (j > 0) {
                      int d1 = res.z.size() + 2 < r.z.size() ? r.z[res.z.size() + 2] : 0;</pre>
136
                      int d2 = res.z.size() + 1 < r.z.size() ? r.z[res.z.size() + 1] : 0;</pre>
137
                      int d3 = res.z.size() < r.z.size() ? r.z[res.z.size()] : 0;</pre>
138
                      q = (int) (((ll) d1 * base * base + (ll) d2 * base + d3) / (firstDigit)
139
        * 2));
                 }
140
141
             }
142
             res.trim();
143
             return res / norm;
144
         }
145
146
147
         bigint operator/(const bigint &v) const {
             return divmod(*this, v).first;
148
         }
149
150
         bigint operator%(const bigint &v) const {
151
             return divmod(*this, v).second;
152
         }
153
154
         bigint &operator/=(int v) {
155
             if (v < 0) sign = -sign, v = -v;
156
             for (int i = (int) z.size() - 1, rem = 0; i >= 0; --i) {
157
                 ll cur = z[i] + rem * (ll) base;
158
159
                 z[i] = (int) (cur / v);
160
                 rem = (int) (cur % v);
             }
161
             trim();
162
             return *this;
163
         }
164
165
166
         bigint operator/(int v) const {
             return bigint(*this) /= v;
167
         }
168
169
         int operator%(int v) const {
170
             if (v < 0) v = -v;
171
172
             int m = 0;
173
             for (int i = (int) z.size() - 1; i >= 0; --i)
174
                 m = (int) ((z[i] + m * (ll) base) % v);
175
             return m * sign;
176
         }
177
178
         bigint &operator*=(const bigint &v) {
             return *this = *this * v;;
179
         }
180
181
```

```
bigint &operator/=(const bigint &v) {
182
             return *this = *this / v;
183
184
185
         bool operator<(const bigint &v) const {</pre>
186
             if (sign != v.sign)
187
                  return sign < v.sign;</pre>
188
             if (z.size() != v.z.size())
189
                  return z.size() * sign < v.z.size() * v.sign;</pre>
190
             for (int i = (int) z.size() - 1; i >= 0; i--)
191
192
                  if (z[i] != v.z[i])
193
                      return z[i] * sign < v.z[i] * sign;</pre>
194
             return false;
         }
195
196
         bool operator>(const bigint &v) const { return v < *this; }</pre>
197
198
         bool operator<=(const bigint &v) const { return !(v < *this); }</pre>
199
200
         bool operator>=(const bigint &v) const { return !(*this < v); }</pre>
201
202
         bool operator==(const bigint &v) const { return !(*this < v) && !(v < *this); }</pre>
203
204
205
         bool operator!=(const bigint &v) const { return *this < v | | v < *this; }</pre>
206
         void trim() {
207
             while (!z.empty() && z.back() == 0) z.pop_back();
208
             if (z.empty()) sign = 1;
209
         }
210
211
212
         bool isZero() const {
213
             return z.empty();
214
         }
215
         friend bigint operator-(bigint v) {
216
217
             if (!v.z.empty()) v.sign = -v.sign;
218
             return v;
219
         }
220
221
         bigint abs() const {
222
             return sign == 1 ? *this : -*this;
         }
223
224
225
         11 longValue() const {
             ll res = 0;
226
227
             for (int i = (int) z.size() - 1; i >= 0; i--)
228
                 res = res * base + z[i];
             return res * sign;
229
         }
230
231
232
         friend bigint gcd(const bigint &a, const bigint &b) {
233
             return b.isZero() ? a : gcd(b, a % b);
234
         }
235
236
         friend bigint lcm(const bigint &a, const bigint &b) {
             return a / gcd(a, b) * b;
237
238
239
         void read(const string &s) {
240
```

```
241
             sign = 1;
             z.clear();
242
             int pos = 0;
243
             while (pos < s.size() && (s[pos] == '-' || s[pos] == '+')) {</pre>
244
                 if (s[pos] == '-') sign = -sign;
245
246
                 ++pos;
247
             for (int i = (int) s.size() - 1; i >= pos; i -= base_digits) {
248
                 int x = 0;
249
                 for (int j = max(pos, i - base\_digits + 1); j <= i; j++)
250
251
                      x = x * 10 + s[j] - '0';
252
                 z.push_back(x);
253
             }
254
             trim();
         }
255
256
         friend istream &operator>>(istream &stream, bigint &v) {
257
             string s;
258
259
             stream >> s;
             v.read(s);
260
             return stream;
261
         }
262
263
264
         friend ostream &operator<<(ostream &stream, const bigint &v) {</pre>
265
             if (v.sign == -1)
                 stream << '-';
266
             stream << (v.z.empty() ? 0 : v.z.back());</pre>
267
             for (int i = (int) v.z.size() - 2; i >= 0; --i)
268
                 stream << setw(base_digits) << setfill('0') << v.z[i];</pre>
269
270
             return stream;
         }
271
272
273
         static vector<int> convert_base(const vector<int> &a, int old_digits, int
        new_digits) {
             vector<ll> p(max(old_digits, new_digits) + 1);
274
             p[0] = 1;
275
276
             for (int i = 1; i < p.size(); i++)</pre>
277
                 p[i] = p[i - 1] * 10;
278
             vector<int> res;
             ll cur = 0;
279
             int cur_digits = 0;
280
             for (int v : a) {
281
                 cur += v * p[cur_digits];
282
283
                 cur_digits += old_digits;
                 while (cur_digits >= new_digits) {
284
                      res.push_back(int(cur % p[new_digits]));
285
                      cur /= p[new_digits];
286
                      cur_digits -= new_digits;
287
                 }
288
             }
289
290
             res.push_back((int) cur);
291
             while (!res.empty() && res.back() == 0) res.pop_back();
292
             return res;
293
         }
294
         typedef vector<ll> vll;
295
296
297
         static vll karatsubaMultiply(const vll &a, const vll &b) {
298
             int n = a.size();
```

```
vll res(n + n);
299
             if (n <= 32) {
300
                 for (int i = 0; i < n; i++)
301
                     for (int j = 0; j < n; j++)
302
303
                         res[i + j] += a[i] * b[j];
304
                 return res;
             }
305
306
             int k = n \gg 1;
307
             vll a1(a.begin(), a.begin() + k);
308
             vll a2(a.begin() + k, a.end());
309
310
             vll b1(b.begin(), b.begin() + k);
311
             vll b2(b.begin() + k, b.end());
312
             vll a1b1 = karatsubaMultiply(a1, b1);
313
             vll a2b2 = karatsubaMultiply(a2, b2);
314
315
             for (int i = 0; i < k; i++) a2[i] += a1[i];
316
             for (int i = 0; i < k; i++) b2[i] += b1[i];
317
318
             vll r = karatsubaMultiply(a2, b2);
319
             for (int i = 0; i < a1b1.size(); i++) r[i] -= a1b1[i];
320
             for (int i = 0; i < a2b2.size(); i++) r[i] -= a2b2[i];</pre>
321
322
323
             for (int i = 0; i < r.size(); i++) res[i + k] += r[i];
             for (int i = 0; i < a1b1.size(); i++) res[i] += a1b1[i];</pre>
324
             for (int i = 0; i < a2b2.size(); i++) res[i + n] += a2b2[i];
325
326
             return res;
        }
327
328
        bigint operator*(const bigint &v) const {
329
             vector<int> a6 = convert_base(this->z, base_digits, 6);
330
             vector<int> b6 = convert_base(v.z, base_digits, 6);
331
             vll a(a6.begin(), a6.end());
332
             vll b(b6.begin(), b6.end());
333
             while (a.size() < b.size()) a.push_back(0);</pre>
334
335
             while (b.size() < a.size()) b.push_back(0);</pre>
336
             while (a.size() & (a.size() - 1)) a.push_back(0), b.push_back(0);
             vll c = karatsubaMultiply(a, b);
337
338
             bigint res;
             res.sign = sign * v.sign;
339
             for (int i = 0, carry = 0; i < c.size(); i++) {
340
                 ll cur = c[i] + carry;
341
                 res.z.push_back((int) (cur % 1000000));
342
                 carry = (int) (cur / 1000000);
343
344
             res.z = convert_base(res.z, 6, base_digits);
345
             res.trim();
346
             return res;
347
348
        }
349 };
    7.4 待修莫队
 1 #include <bits/stdc++.h>
 2 using namespace std;
 3 #define ACM_LOCAL
 4 const int N = 2e6 + 10;
```

```
5 int n, m, k, a[N], cnt[N], qnum, cnum, pos[N];
   int Ans, ans[N], x, y;
6
   char op;
7
   inline int read(){
8
9
      int s=0, w=1;
      char ch=getchar();
10
      while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
11
      while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();</pre>
12
      return s*w;
13
   }
14
15
16 struct node {
17
       int l, r, pre, id;
18
   }q[N];
19
20 struct node2 {
       int val, pos;
21
22
   }c[N];
23
24 bool cmp(node x, node y) {
       if (x.l != y.l) return pos[x.l] < pos[y.l];</pre>
25
       if (x.r != y.r) return pos[x.r] < pos[y.r];</pre>
26
27
       return x.pre < y.pre;</pre>
28 }
29
  inline void add(int x) {if (++cnt[a[x]] == 1) ++Ans;}
30
31
   inline void sub(int x) {if (--cnt[a[x]] == 0) --Ans;}
32
33
34
   inline void work(int now, int i) {
35
36
       if (c[now].pos >= q[i].l && c[now].pos <= q[i].r) {</pre>
            if (--cnt[a[c[now].pos]] == 0) --Ans;
37
            if (++cnt[c[now].val] == 1) ++Ans;
38
39
       swap(c[now].val,a[c[now].pos]);// 浜ゆ崲鏀瑰彉鐨勫∈煎拰鍘熸潵鐨勫∈硷紝鏂逛究鍥為∈€鐨勬椂鍊
40
41
   }
42
   void solve() {
43
       n = read(), m = read();
44
       int siz = pow(n, 2.0 / 3.0);
45
       for (int i = 1; i \le n; ++i) a[i] = read(), pos[i] = (i - 1) / siz + 1;
46
47
       while (m--) {
48
49
            op = getchar();
50
            if (op == 'Q') {
                q[++qnum].l = read();
51
                q[qnum].r = read();
52
53
                a[qnum].id = qnum;
54
                q[qnum].pre = cnum;
55
            }
56
            else {
                c[++cnum].pos = read();
57
                c[cnum].val = read();
58
59
            }
60
       }
61
62
       sort(q+1, q+qnum+1, cmp);
```

```
63
       int l = 1, r = 0, now = 0;
       for (int i = 1; i \le qnum; ++i) {
64
            while (q[i].l < l) add(--l);
65
66
            while (q[i].r > r) add(++r);
            while (q[i].l > l) sub(l++);
67
            while (q[i].r < r) sub(r--);
68
            while (now < q[i].pre) work(++now, i);</pre>
69
            while (now > q[i].pre) work(now--, i);
70
            ans[q[i].id] = Ans;
71
       }
72
73
74
       for (int i = 1; i \leftarrow qnum; ++i) printf("%d\n", ans[i]);
75
76
   signed main() {
77
       //ios_base::sync_with_stdio(false);
78
79
       //cin.tie(0);
80
       //cout.tie(0);
   #ifdef ACM_LOCAL
81
       freopen("in.txt", "r", stdin);
freopen("out.txt", "w", stdout);
82
83
   //#else
84
85
       solve();
86
   #endif
87
        return 0;
   }
88
   7.5 对拍
1 /***¶□ □ □ ***/
2 //D D D D LD f°test.cpp biaoda.cpp data.cpp input.txt duipai.bat:
3 duipai.bat:
4 :again
5 data > input.txt
6 biaoda < input.txt > biaoda_out.txt
7 test < input.txt > test_out.txt
8 fc biaoda_out.txt test_out.txt
9 if not errorlevel 1 goto again
10 pause
   7.6 二讲制压缩枚举子集
   for (int sub = S; sub; sub = (sub - 1) & S) {
2
       // sub 为 S 的子集
3
  a \wedge b = c \longrightarrow c \wedge b = a
   7.7 二维差分
   void Insert(int x1, int y1, int x2, int y2, int v) {
1
       c[x1][y1] += v;
2
3
       c[x1][y2+1] -= v;
       c[x2+1][y1] -= v;
4
       c[x2+1][y2+1] += v;
5
6 }
```

7.8 分块

```
int block = sqrt(n);
   int num = n / block + (n % block ? 1 : 0);
3
   for (int i = 1; i <= num; i++) {
        l[i] = (i - 1) * block + 1;
4
        r[i] = i * block;
5
   }
6
7
   r[num] = n;
   for (int i = 1; i \le n; i++) belong[i] = (i - 1) / block + 1;
    7.9
         高精度
   #include<cstdio>
   #include <cassert>
3 #include<cstring>
4 #include<algorithm>
5 using namespace std;
6 const int MOD=10000;
7 const int B=10000;
8 const int SIZEN=505;
9 const int L=505;
10 struct Mat{
        int num[40][40];
11
12
        void init(int n){
13
             for(int i=0;i<n;i++)</pre>
14
                 for(int j=0;j<n;j++)</pre>
15
                      num[i][j]=i*n+j;
16
        void change(int n){
17
18
             int t_num[40][40];
19
             for(int i=0;i<n;i++)</pre>
20
                 for(int j=0;j<n;j++) t_num[j][n-i-1]=num[i][j];</pre>
21
             for(int i=0;i<n;i++)</pre>
22
                 for(int j=0;j<n;j++) num[i][j]=t_num[i][j];</pre>
23
24
        void change1(int n){
25
             int t_num[40][40];
26
             for(int i=0;i<n;i++)</pre>
27
                 for(int j=0;j<n;j++) t_num[i][n-j-1]=num[i][j];</pre>
28
             for(int i=0;i<n;i++)</pre>
                 for(int j=0;j<n;j++) num[i][j]=t_num[i][j];</pre>
29
30
        void change2(int n){
31
32
             int t_num[40][40];
             for(int i=0;i<n;i++)</pre>
33
                 for(int j=0;j<n;j++) t_num[n-i-1][j]=num[i][j];</pre>
34
             for(int i=0;i<n;i++)</pre>
35
36
                 for(int j=0;j<n;j++) num[i][j]=t_num[i][j];</pre>
37
        void change3(int n){
38
             int t_num[40][40];
39
40
             for(int i=0;i<n;i++)</pre>
41
                 for(int j=0;j<n;j++) t_num[j][i]=num[i][j];</pre>
             for(int i=0;i<n;i++)</pre>
42
                 for(int j=0;j<n;j++) num[i][j]=t_num[i][j];</pre>
43
44
        void change4(int n){
45
```

```
int t_num[40][40];
46
              for(int i=0;i<n;i++)</pre>
47
                  for(int j=0;j<n;j++) t_num[n-1-j][n-1-i]=num[i][j];
48
             for(int i=0;i<n;i++)</pre>
49
50
                  for(int j=0;j<n;j++) num[i][j]=t_num[i][j];</pre>
51
52
         void output(int n){
              for(int i=0;i<n;i++){</pre>
53
                  for(int j=0;j<n;j++) printf("%d ",num[i][j]);</pre>
54
                  printf("\n");
55
56
             }
57
         }
    };
58
    struct BigInteger {
59
         BigInteger(int number = 0) : length(!!number) {
60
             assert(0 <= number && number < B);</pre>
61
             memset(digit, 0, sizeof(digit));
62
63
             digit[0] = number;
         }
64
65
         BigInteger normalize() {
66
             while (length && !digit[length - 1]) {
67
                  length --;
68
69
             }
70
             return *this;
         }
71
72
73
         int operator[](int index) const {
74
              return digit[index];
75
 76
77
         int& operator[](int index) {
78
             return digit[index];
         }
79
80
         void output(){
81
82
             printf("%d",digit[length-1]);
83
             for(int i=length-2;i>=0;i--) printf("%04d",digit[i]);
             printf("\n");
84
85
         }
86
         int length, digit[L];
87
    };
88
89
    bool operator < (const BigInteger &a, const BigInteger &b)</pre>
90
    {
91
         if (a.length != b.length) {
92
             return a.length < b.length;</pre>
93
94
95
         for (int i = 0; i < a.length; ++ i) {
96
             if (a[i] != b[i]) {
97
                  return a[i] < b[i];</pre>
98
             }
99
         return false;
100
101
102
    BigInteger operator + (const BigInteger &a, const BigInteger &b)
103
104
    {
```

```
105
        BigInteger c;
        c.length = std::max(a.length, b.length) + 1;
106
        for (int i = 0, delta = 0; i < c.length; ++ i) {
107
             delta += a[i] + b[i];
108
109
             c[i] = delta \% B;
             delta /= B;
110
111
112
        return c.normalize();
113
    }
114
    BigInteger operator - (const BigInteger &a, int b)
115
116
        assert(0 \le b \&\& b < B);
117
        BigInteger c;
118
        c.length = a.length;
119
        for (int i = 0, delta = -b; i < a.length; ++ i) {
120
121
             delta += a[i];
122
             c[i] = delta;
             delta = 0;
123
             if (c[i] < 0) {
124
                 c[i] += B;
125
                 delta = -1;
126
             }
127
128
129
        return c.normalize();
130
    }
131
    BigInteger operator * (const BigInteger &a, const BigInteger &b)
132
133
    {
        BigInteger c;
134
        c.length = a.length + b.length;
135
        for (int i = 0; i < a.length; ++ i) {
136
             for (int j = 0, delta = 0; j \le b.length; ++ j) {
137
                 delta += a[i] * b[j] + c[i + j];
138
                 c[i + j] = delta \% B;
139
                 delta /= B;
140
141
             }
142
143
        return c.normalize();
    }
144
145
    BigInteger operator / (const BigInteger &a, int b)
146
147
148
        assert(0 \le b \&\& b < B);
        BigInteger c;
149
        c.length = a.length;
150
        for (int i = c.length - 1, delta = 0; i >= 0; --i) {
151
             delta = delta * B + a[i];
152
             c[i] = delta / b;
153
154
             delta %= b;
155
156
        return c.normalize();
157
    BigInteger operator ^(const BigInteger &a,int b){
158
        BigInteger ret,ta;
159
160
        ret=1;ta=a;
161
        while(b){
162
             if(b&1) ret=ret*ta;
             ta=ta*ta;
163
```

```
164
            b>>=1;
        }
165
166
        return ret;
167
168
    Mat mat;
    BigInteger ret,tmp;
169
    void solve(int n,int c){
170
        ret=0;
171
        ret.normalize();
172
173
        tmp=c;
174
        tmp.normalize();
175
        if(n\%2==0){
            ret=ret+(tmp^(n*n));
176
            ret=ret+(tmp^(n*n/4));
177
            ret=ret+(tmp^{n*n/2});
178
            ret=ret+(tmp^{n*n/4});
179
            ret=ret+(tmp^(n*n/2))*2;
180
            ret=ret+(tmp^{(n*n-n)/2+n)}*2;
181
        }
182
        else{
183
            ret=ret+(tmp^(n*n));
184
            ret=ret+(tmp^(n*n-1)/4+1);
185
            ret=ret+(tmp^(n*n-1)/2+1);
186
187
            ret=ret+(tmp^(n*n-1)/4+1);
188
            ret=ret+(tmp((n*n-n)/2+n))*2;
            ret=ret+(tmp^{(n*n-n)/2+n)}*2;
189
        }
190
        ret=ret/8;
191
        ret.output();
192
193
    int main()
194
195
    {
196
        int n,c;
        while(scanf("%d%d",&n,&c)!=EOF)
197
            solve(n,c);
198
    }
199
    7.10
           环形均分
    // Created by SANZONG on 2021/7/8.
   //
 3
    #include "bits/stdc++.h"
 4
 5
    #define int long long
 6
    using namespace std;
 7
   const int maxn = 3000000;
 9 int arr[maxn];
10 int sum[maxn];
    signed main() {
          freopen("in.txt","r",stdin);
12
        //前缀和为0说明前一段内部可以自己解决,其内部的前缀和绝对值相加即为前一段的子段转移花费
13
14
        int n;
15
        cin >> n;
        int ave = 0;
16
        for (int i = 1; i \le n; ++i) {
17
            cin >> arr[i];
18
19
            ave += arr[i];
```

```
}
20
       ave /= n;
21
       for (int i = 1; i <= n; ++i) {
22
23
            arr[i] -= ave;
24
            sum[i] = sum[i-1] + arr[i];
25
26
       sort(sum+1,sum+1+n);
       int mid = (n+1)/2;
27
       int ans = 0;
28
       for (int i = 1; i <= n; ++i) {
29
30
            ans += abs(sum[i] - sum[mid]);
                                              //做前缀和相加模拟出了传递时的花费
31
32
       cout << ans << endl;</pre>
   }
33
   7.11
          回滚莫队
1 #include <bits/stdc++.h>
2 using namespace std;
3 //#define ACM_LOCAL
4 typedef long long ll;
5 const int N = 2e5 + 10;
6 int n, m, k, a[N], pos[N], x1[N], xr[N], b[N], _cnt[N], st[N], ed[N], clear[N];
7 int Max, temp, ans[N], num;
8
9
   inline int read(){
10
       int s=0, w=1;
11
       char ch=getchar();
       while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
12
       while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();</pre>
13
14
       return s*w;
15
   }
16
17
   struct node {
       int l, r, id;
18
       bool operator < (node xx) const {</pre>
19
20
            if(pos[l] == pos[xx.l]) return r < xx.r;</pre>
            else return pos[l] < pos[xx.l];</pre>
21
22
23
   }q[N];
24
   void solve() {
25
26
       n = read();
       for (int i = 1; i \le n; i++) a[i] = read(), b[i] = a[i];
27
28
       sort(b+1, b+n+1);
29
       int tot = unique(b+1, b+n+1) - b-1;
30
       for (int i = 1; i \le n; i++) a[i] = lower_bound(b+1, b+tot+1, a[i]) - b;
31
32
33
       for (int i = 1; i \ll m; i++) q[i].l = read(), q[i].r = read(), q[i].id = i;
34
35
36
       int siz = sqrt(n);
37
       for (int i = 1; i <= n; i++){
            pos[i] = i / siz;
38
            xl[pos[i]] = (xl[pos[i]] == 0 || xl[pos[i]] > i) ? i : xl[pos[i]];
39
            xr[pos[i]] = (xr[pos[i]] < i) ? i : xr[pos[i]];
40
       }
41
```

```
42
        sort(q+1, q+m+1);
43
44
        int l = 1, r = 0, lastblock = -1;
45
46
        for (int i = 1; i <= m; i++) {
47
             if (pos[q[i].l] == pos[q[i].r]) {
48
                 int temp = 0;
49
                 for (int j = q[i].l; j \leftarrow q[i].r; j++) {
50
                      if (!_cnt[a[j]]) _cnt[a[j]] = j;
51
                      else temp = max(temp, j - _cnt[a[j]]);
52
53
                 for (int j = q[i].l; j \ll q[i].r; j++) _cnt[a[j]] = 0;
54
                 ans[q[i].id] = temp;
55
            }
56
            else {
57
                    (lastblock != pos[q[i].l]) {
58
                      l = xr[pos[q[i].l]] + 1;
59
                     r = 1 - 1;
60
                      for (int j = 1; j <= num; j++) st[clear[j]] = ed[clear[j]] = 0;</pre>
61
                     num = 0;
62
                     Max = 0, lastblock = pos[q[i].l];
63
64
                 while (r < q[i].r) {
65
66
                      r++;
                      ed[a[r]] = r;
67
68
                     clear[++num] = a[r];
                     if (!st[a[r]]) st[a[r]] = r;
69
                     Max = max(Max, r - st[a[r]]);
70
71
72
                 temp = Max;
                 while (l > q[i].l) {
73
                     1--;
74
                      if (ed[a[1]]) temp = max(temp, ed[a[1]] - 1);
75
                      else ed[a[1]] = 1;
76
77
78
                 while (l < xr[pos[q[i].l]] + 1) {</pre>
79
                      if (ed[a[1]] == 1) ed[a[1]] = st[a[1]] = 0;
80
                     1++;
81
82
                 ans[q[i].id] = temp;
            }
83
        }
84
85
        for (int i = 1; i \le m; i++) printf("%d\n", ans[i]);
86
87
   }
88
   signed main() {
89
        //ios_base::sync_with_stdio(false);
90
91
        //cin.tie(0);
92
        //cout.tie(0);
93
   #ifdef ACM_LOCAL
        freopen("in.txt", "r", stdin);
freopen("out.txt", "w", stdout);
94
95
    #endif
96
97
        solve();
98
        return 0;
99
   }
```

7.12 离散化

```
struct Hash {
1
2
       int b[N], tot;
3
       void init() {tot = 0;}
4
       void insert(int x) {b[++tot] = x;}
5
       void build() {
            sort(b+1, b+1+tot);
6
7
            tot = unique(b+1, b+tot+1) - (b+1);
8
       int pos(int x) {return lower_bound(b+1, b+tot+1, x) - b;}
9
   }ha;
10
   7.13 莫队 + 树状数组
1 #include <bits/stdc++.h>
   using namespace std;
3 #define ACM_LOCAL
4 \quad const int N = 1e5 + 10;
5 int n, m, k, a[N], pos[N], b[N], upa[N], prea[N], sum[N], Ans, ans[N];
6
   inline int read(){
7
      int s=0, w=1;
8
9
      char ch=getchar();
      while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}
10
      while(ch>='0'&&ch<='9') s=s*10+ch-'0',ch=getchar();</pre>
11
12
      return s*w;
13
   }
14
15
  struct Q{
       int 1, r, id;
16
17
   }p[N];
18
19
   bool cmp(Q x, Q y)  {
20
       if (pos[x.l] == pos[y.l]) return x.r < y.r;
       else return pos[x.l] < pos[y.l];</pre>
21
22 }
23
24 inline int lowbit(int x) {return x & -x;}
25
26 void add(int p, int v) {for (int i = p; i \le 3*n+10; i += lowbit(i)) sum[i] += v;}
27
28
   int query(int p) {
29
        int res = 0;
30
        for (int i = p; i > 0; i -= lowbit(i)) res += sum[i];
31
       return res;
32
  int query_(int l, int r) {return query(r) - query(l-1);}
33
34
   void solve() {
35
36
       n = read(), m = read(), k = read();
       int tot = 0;
37
       int siz = sqrt(n);
38
39
       for (int i = 1; i \le n; i++) a[i] = read();
40
       for (int i = 1; i <= n; i++) {
41
            upa[i] = a[i] + k, prea[i] = a[i] - k;
42
           b[++tot] = a[i], b[++tot] = upa[i], b[++tot] = prea[i];
43
```

```
44
        sort(b + 1, b + tot + 1);
45
        int cnt = unique(b + 1, b + tot + 1) - b - 1;
46
        for (int i = 1; i <= n; i++) {
47
             a[i] = lower\_bound(b+1, b+cnt+1, a[i]) - b;
48
             upa[i] = lower_bound(b+1, b+cnt+1, upa[i]) - b;
49
50
             prea[i] = lower_bound(b+1, b+cnt+1, prea[i]) - b;
        }
51
52
        for (int i = 1; i <= m; i++) {
53
            p[i].l = read(), p[i].r = read();
54
55
            p[i].id = i;
        }
56
        sort(p + 1, p + m + 1, cmp);
57
58
        int l = 1, r = 0;
        for (int i = 1; i <= m; i++) {
    while (p[i].l < l) {</pre>
59
60
61
                 Ans += query_(prea[l], upa[l]);
62
                 add(a[l], 1);
63
             }
64
             while (p[i].l > l) {
65
66
                 add(a[1], -1);
67
                 Ans -= query_(prea[l], upa[l]);
68
                 1++;
69
            while (p[i].r < r) {
70
                 add(a[r], -1);
71
                 Ans -= query_(prea[r], upa[r]);
72
73
                 r--;
74
75
            while (p[i].r > r) {
76
                 r++;
                 Ans += query_(prea[r], upa[r]);
77
                 add(a[r], 1);
78
79
             }
80
            ans[p[i].id] = Ans;
81
        for (int i = 1; i \le m; i++) printf("%d\n", ans[i]);
82
   }
83
84
   signed main() {
85
        ios_base::sync_with_stdio(false);
86
87
        cin.tie(0);
        cout.tie(0);
88
89
   #ifdef ACM_LOCAL
        freopen("in.txt", "r", stdin);
freopen("out.txt", "w", stdout);
90
91
   //#else
92
93
        solve();
94
   #endif
95
        return 0;
96
   }
    7.14
           普诵莫队
```

1 #include <bits/stdc++.h>
2 //#define ACM_LOCAL

```
3 using namespace std;
   const int N = 5e4 + 10;
5
   inline int read() {
7
       int s = 0, w = 1;
       char ch = getchar();
8
       while (ch < '0' || ch>'9') { if (ch == '-')w = -1; ch = getchar(); }
9
       while (ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch = getchar();
10
       return s * w;
11
12 }
13
14 int pos[N], a[N], vis[2 * N];
15 int n, q, num, k;
16 long long res, ans[N];
17
  struct Q {
18
       int l, r, id;
19
20 }p[N];
21
22 bool cmp(Q x, Q y) {
       if (pos[x.l] == pos[y.l]) return x.r < y.r;
23
24
       else return pos[x.l] < pos[y.l];</pre>
25 }
27 void add(int x) {
28
29 }
30
31 void sub(int x) {
32
33 }
34
35 void solve() {
       n = read(), q = read(), k = read();
36
37
       for (int i = 1; i <= n; i++) a[i] = read();</pre>
38
39
       int siz = sqrt(n);
40
       for (int i = 1; i <= n; i++) pos[i] = i / siz;</pre>
41
42
       for (int i = 1; i \le q; i++) {
           p[i].l = read(), p[i].r = read();
43
           p[i].id = i;
44
45
46
       sort(p + 1, p + q + 1, cmp);
47
48
       int l = 1, r = 0;
49
       for (int i = 1; i <= q; i++) {
            while (p[i].l < l) add(--l);
50
            while (p[i].r > r) add(++r);
51
            while (p[i].l > l) sub(l++);
53
            while (p[i].r < r) sub(r--);
54
            ans[p[i].id] = res;
55
       for (int i = 1; i \le q; i++) {
56
            printf("%lld\n", ans[i]);
57
       }
58
59 }
```

7.15 三维偏序问题 (CDQ 分治)

```
#include <bits/stdc++.h>
2 using namespace std;
3 #define fi first
4 #define se second
5 #define re register
6 typedef long long ll;
7 typedef pair<int, int> PII;
8 typedef unsigned long long ull;
9 const int N = 5e5 + 10, M = 1e6 + 5, INF = 1e9;
10 const int MOD = 1e9 + 7;
int n, k, cnt[N], ans[N], f[N];
12 struct node {
13
        int a, b, c, cnt, id;
        bool operator < (const node &rhs) {</pre>
14
15
            if (a == rhs.a) {
                if (b == rhs.b) return c < rhs.c;</pre>
16
17
                return b < rhs.b;</pre>
18
19
            return a < rhs.a;</pre>
20
        }
21
   };
22 node p[N], q[N], tmp[N];
23 int lowbit(int x) {return x&-x;}
24
   void add(int x, int c) {
25
        for (int i = x; i <= k; i += lowbit(i)) cnt[i] += c;</pre>
26
27
   int query(int x) {
28
        int res = 0;
        for (int i = x; i > 0; i -= lowbit(i)) res += cnt[i];
29
30
        return res;
31
   }
32
   void clear(int x) {
33
        for (int i = x; i <= k; i += lowbit(i)) {</pre>
34
            if (cnt[i]) cnt[i] = 0;
35
            else break;
36
        }
37
   }
   void CDQ(int 1, int r) {
39
        if (l == r) return;
40
        int mid = (l + r) >> 1;
41
        CDQ(l, mid);
        CDQ(mid+1, r)
42
        int t1 = 1, t^2 = mid+1;
43
44
        for (int i = l; i <= r; i++) {
            if ((t1 <= mid && q[t1].b <= q[t2].b) || t2 > r) {
45
                add(q[t1].c, q[t1].cnt);
46
                tmp[i] = q[t1++];
47
48
            } else {
                ans[q[t2].id] += query(q[t2].c);
49
50
                tmp[i] = q[t2++];
            }
51
52
53
        for (int i = l; i <= r; i++) q[i] = tmp[i], clear(q[i].c);
54
   }
55
   void solve() {
56
        cin >> n >> k;
        for (int i = 1; i <= n; i++) cin >> p[i].a >> p[i].b >> p[i].c;
57
```

```
sort(p+1, p+n+1);
58
        int tot = 0;
59
        q[++tot] = \{p[1].a, p[1].b, p[1].c, 1, 1\};
60
        for (int i = 2; i <= n; i++) {
61
            if (p[i].a == p[i-1].a \& p[i].b == p[i-1].b \& p[i].c == p[i-1].c) {
62
63
                q[tot].cnt++;
            } else {
64
                q[++tot] = \{p[i].a, p[i].b, p[i].c, 1, tot\};
65
66
67
68
        CDQ(1, tot);
69
        for (int i = 1; i <= tot; i++) {
            f[ans[q[i].id] + q[i].cnt - 1] += q[i].cnt;
70
71
        for (int i = 0; i < n; i++) printf("%d\n", f[i]);
72
73
74
75
   signed main() {
        ios_base::sync_with_stdio(false);
76
        cin.tie(0);
77
        cout.tie(0);
78
   #ifdef ACM_LOCAL
79
        freopen("input", "r", stdin);
freopen("output", "w", stdout);
80
81
82
   #endif
83
        solve();
        return 0;
84
   }
85
   7.16
          树上莫队
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 1e5 + 5;
4 inline int read() {
        int s = 0, w = 1;
5
6
        char ch = getchar();
        while (ch < '0' | | ch > '9')  { if (ch == '-')w = -1; ch = getchar(); }
7
8
        while (ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch = getchar();
9
        return s * w;
10 }
11
   int n, m;
12
   struct edge {
13
        int to, next;
   }e[N<<1];
14
15
   struct Q {
16
        int l, r, id, lca;
17
18
   }p[N];
19
   int h[N], cnt, tot, in[N], out[N], a[N], b[N], pos[N], son[N], rnk[N], siz[N], top[N],
       d[N], fa[N], res, used[N], ans[N], vis[N];
21
22
   void add(int u, int v) {
        e[cnt].to = v;
23
        e[cnt].next = h[u];
24
25
        h[u] = cnt++;
26 }
```

```
27
   28
        if (pos[x.l] == pos[y.l]) return x.r < y.r;</pre>
29
        else return pos[x.l] < pos[y.l];</pre>
30
   }
31
32
   void discrete() {
33
        sort(b + 1, b + n + 1);
34
        int num = unique(b + 1, b + n + 1) - b - 1;
35
        for (int i = 1; i \le n; i++) a[i] = lower_bound(b + 1, b + num + 1, a[i]) - b;
36
37
   }
38
   void dfs1(int u) {
39
        son[u] = -1;
40
        siz[u] = 1;
41
        in[u] = ++tot, rnk[tot] = u;
42
        for (int i = h[u]; \sim i; i = e[i].next) {
43
            int v = e[i].to;
44
            if (!d[v]) {
45
                d[v] = d[u] + 1;
46
                fa[v] = u;
47
                dfs1(v);
48
                siz[u] += siz[v];
49
50
                if (son[u] == -1 \mid | siz[v] > siz[son[u]]) son[u] = v;
51
            }
52
        out[u] = ++tot, rnk[tot] = u;
53
   }
54
55
   void dfs2(int u, int t) {
56
        top[u] = t;
57
        if (son[u] == -1) return;
58
59
        dfs2(son[u], t);
        for (int i = h[u]; \sim i; i = e[i].next) {
60
            int v = e[i].to;
61
            if (v != son[u] && v != fa[u]) dfs2(v, v);
62
63
        }
64
   }
65
   int lca(int u, int v) {
66
        while (top[u] != top[v]) {
67
            if (d[top[u]] > d[top[v]])
68
69
                u = fa[top[u]];
70
            else
                v = fa[top[v]];
71
72
73
        return d[u] > d[v] ? v : u;
74 }
75
   void Add(int x) {
77
        if (++vis[a[x]] == 1) ++res;
78
   }
79
   void Sub(int x) {
80
        if (--vis[a[x]] == 0) --res;
81
   }
82
83
   void ADD(int x) {
84
        used[x] ? Sub(x) : Add(x);
85
```

```
used[x] ^= 1;
86
    }
87
88
    int main() {
89
90
         n = read(), m = read();
         memset(h, -1, sizeof h);
91
92
         int siz = sqrt(n);
         for (int i = 1; i <= n; i++) a[i] = read(), b[i] = a[i];
93
         for (int i = 1; i <= 2 * n; i++) pos[i] = i / siz;
94
         discrete();
95
96
         for (int i = 1; i <= n - 1; i++) {
97
             int x, y;
             x = read(), y = read();
98
             add(x, y), add(y, x);
99
100
         d[1] = 1;
101
         dfs1(1);
102
         dfs2(1, 1);
103
         for (int i = 1; i <= m; i++) {
104
             int x, y;
105
             x = read(), y = read();
106
             if (in[x] > in[y]) swap(x, y);
107
             int lca_= lca(x, y);
108
109
             p[i].id = i;
110
             if (lca_{=} = x) p[i].l = in[x], p[i].r = in[y];
             else p[i].l = out[x], p[i].r = in[y], p[i].lca = lca_;
111
112
         sort(p + 1, p + m + 1, cmp);
113
114
         int l = 1, r = 0;
115
         for (int i = 1; i <= m; i++) {
    while (p[i].l < 1) ADD(rnk[--l]);</pre>
116
117
             while (p[i].r > r) ADD(rnk[++r]);
118
             while (p[i].l > l) ADD(rnk[l++]);
119
             while (p[i].r < r) ADD(rnk[r--]);</pre>
120
             if (p[i].lca) ADD(p[i].lca);
121
122
             ans[p[i].id] = res;
123
             if (p[i].lca) ADD(p[i].lca);
124
125
         for (int i = 1; i \le m; i++) printf("%d\n", ans[i]);
126 }
```

8 字符串

```
//
1
  // Created by acer on 2021/2/16.
3
  //
4 //判断子串,不同子串个数,所有子串字典序第i大,最长公共子串
5
6 #include <cstring>
  #include <iostream>
7
  #include "string"
8
9
10
#define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 1e6 + 10;
14 int id[MAXN<<1];</pre>
15 int visit[MAXN<<1];</pre>
16 int endpos[MAXN << 1];</pre>
17 int siz[MAXN << 1];</pre>
18 int len[MAXN << 1];</pre>
19 int ch[MAXN << 1][27];</pre>
20 int fa[MAXN << 1];</pre>
21 int last = 1;
22 int tot = 1;
23 int p;
24 int len1;
25
   void add(int c) {
26
       p = last;
27
       last = ++tot;
       int np = last;
28
       siz[np] = 1;
29
30
       len[np] = len[p] + 1;
31
       for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
32
       if (!p) fa[np] = 1;
       else {
33
            int q = ch[p][c];
34
            if (len[q] == len[p] + 1) fa[np] = q;
35
36
            else {
37
                int nq = ++tot;
38
                memcpy(ch[nq],ch[q],sizeof(ch[nq]));
39
                fa[nq] = fa[q];
                len[nq] = len[p] + 1;
40
                fa[np] = fa[q] = nq;
41
                for (; p && ch[p][c] == q; p = fa[p]) {
42
43
                    ch[p][c] = nq;
44
            }
45
46
47
       }
48
49
   }
   void getTuopu()
51
   {
        for (int k = 1; k \le tot; ++k) {
52
                                                //给每个点赋权值
53
            endpos[len[k]] ++;
54
55
       for (int m = 1; m <= tot; ++m) {</pre>
                                              //获得长度大于等于m的点的总个数,所以必然是不会相同的,
       故可以用来标记。
            endpos[m] += endpos[m-1];
56
```

```
57
        for (int n = 1; n <= tot; ++n) { //根据点出现顺序获得拓扑序
58
            id[endpos[len[n]]--] = n;
59
60
    }
61
    char s[MAXN], s1[MAXN];
62
63
    void doit()
64
    {
65
        for (int i = tot; i >= 1; --i) {
66
67
           int p = id[i];
68
            siz[fa[p]] += siz[p];
        }
69
    }
70
71
    int solve(int w) {
72
73
        int p = 1;
74
        int tmp = 0;
        int sum = 0;
75
76
        for (int i = 1; i <= len1+len1; ++i) {</pre>
            int v = s1[i]-'a';
77
            if (ch[p][v])
78
79
                p = ch[p][v], tmp++;
80
81
                while (p \&\& !ch[p][v]) p = fa[p];
82
                if (!p) p = 1, tmp = 0;
83
                else
                    tmp = len[p]+1, p = ch[p][v];
84
85
    ///siz[p]是[i-l+1,i]的出现次数,我们需要求的是[i-l+l+1,i]的
86
    ///直到parent树上的祖先是孩子的后缀,我们一直往上fa
    ///直到父亲的longest小于ItI
89
   ///fa即删前缀字符, ch加后缀字符, 因为在同一个endpos, 所以跳完之后即使长度不等, endpos所包含的个数也是
90
        一样的
91
            if(tmp >= len1)
92
            {
93
                while (len[fa[p]]>=len1) p = fa[p];
                                                           //fa的longest严格小于len1, 保证可以加
                if( visit[p]!=w )
94
                    sum+=siz[p], visit[p] = w;
95
96
                tmp = len1;
97
            }
98
99
100
        return sum;
101
    }
102
    int main() {
103
        scanf("%s", s + 1);
104
105
        int l = strlen(s + 1);
106
        for (int i = 1; i <= l; ++i) {
107
            add(s[i] - 'a');
108
109
        getTuopu();
        doit();
110
111
        int n;
112
        cin >> n;
113
        for (int j = 1; j \le n; ++j)
114
        {
```

```
115
            scanf("%s",s1+1);
116
            len1 = strlen(s1+1);
117
            for (int i = 1; i <= len1; ++i) {</pre>
118
                s1[i+len1] = s1[i];
119
            }
120
121
            cout << solve(j) << endl;</pre>
122
123
        }
124 }
    8.1 (SAM)k 长子串最大出现次数
   //
   // Created by acer on 2021/2/16.
 3 //
   //判断子串,不同子串个数,所有子串字典序第i大,最长公共子串
 5
   #include <cstring>
 6
 7
    #include <iostream>
    #include "string"
 8
 9
10
#define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 3e5 + 10;
14
15
   int len[MAXN << 1];int ch[MAXN << 1][27];int fa[MAXN << 1];int weig[MAXN << 1];
16
   int last = 1;int tot = 1;int p;int size[MAXN<<1];</pre>
    void add(int c) {
17
18
        p = last;
19
        last = ++tot;
20
        int np = last;
        size[np] = 1;
21
                               //np节点表示的后缀出现过几次
22
        len[np] = len[p] + 1;
23
        for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
        if (!p) fa[np] = 1;
24
        else {
25
26
            int q = ch[p][c];
27
            if (len[q] == len[p] + 1) fa[np] = q;
28
            else {
                int nq = ++tot;
29
                memcpy(ch[nq],ch[q],sizeof(ch[nq]));
30
31
                fa[nq] = fa[q];
32
                len[nq] = len[p] + 1;
                fa[np] = fa[q] = nq;
33
                for (; p && ch[p][c] == q; p = fa[p]) {
34
35
                     ch[p][c] = nq;
                }
36
            }
37
38
39
        }
    }
40
41
    char s[MAXN<<1];int id[MAXN<<1];int ans[MAXN<<1];</pre>
    void getTuopu()
42
43
    {
        for (int k = 1; k \le tot; ++k) {
44
                                                //给每个点赋权值
            weig[len[k]] ++;
45
```

```
46
       for (int m = 1; m <= tot; ++m) {</pre>
47
                                              //获得长度小于等于m的点的总个数, 所以必然是不会相同的,
       故可以用来标记。
           weig[m] += weig[m-1];
48
49
       for (int n = 1; n <= tot; ++n) { //根据点出现顺序获得拓扑序
50
            id[weig[len[n]]--] = n;
51
       }
52
   }
53
   int main() {
54
55
       scanf("%s", s + 1);
56
       int l = strlen(s+1);
       for (int i = 1; i <= l; ++i) {
57
           add(s[i]-'a');
58
59
       getTuopu();
60
       for (int i = tot; i >= 1 ; --i) {
    size[fa[id[i]]] += size[id[i]];
61
62
            ans[len[id[i]]] = max(ans[len[id[i]]], size[id[i]]);
63
64
       for(int i = tot; i >= 1; --i) ans[i] = max(ans[i], ans[i + 1]);
65
          for(int i = 1; i <= l; ++i) printf("%d\n", ans[i]);
66
   //
67
   }
        (SAM) 第 k 小子串
1
  // Created by acer on 2021/2/16.
2
3 //
4 //判断子串,不同子串个数,所有子串字典序第i大,最长公共子串
5
6 #include <cstring>
7 #include <iostream>
8 #include "string"
9
10
#define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 3e5 + 10;
14
15 int len[MAXN << 1];</pre>
16 int ch[MAXN << 1][27];</pre>
  int fa[MAXN << 1];</pre>
18 int last = 1;
19 int tot = 1;
20 int p;
21
   void add(int c) {
22
       p = last;
23
24
       last = ++tot;
       int np = last;
25
       len[np] = len[p] + 1;
26
27
       for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
28
       if (!p) fa[np] = 1;
       else {
29
            int q = ch[p][c];
30
            if (len[q] == len[p] + 1) fa[np] = q;
31
            else {
32
```

```
33
                int nq = ++tot;
                memcpy(ch[nq], ch[q], sizeof(ch[nq]));
34
                fa[nq] = fa[q];
35
                len[nq] = len[p] + 1;
36
                fa[np] = fa[q] = nq;
37
                for (; p && ch[p][c] == q; p = fa[p]) {
38
                    ch[p][c] = nq;
39
40
            }
41
42
43
        }
44
   }
45
   char s[MAXN];
46
   int count[MAXN<<1];</pre>
47
   void dfs(int k)
49
   {
        if (count[k]) return;
50
        count[k] = 1;
51
        for (int i = 0; i <= 26; ++i) {
52
                                                  //此处是实现算法的关键,通过从a遍历到z来计算字典序
            if (!ch[k][i]) continue;
53
            dfs(ch[k][i]);
54
            count[k]+=count[ch[k][i]];
                                              //很明显, count的意义就是字典序为k的子串的首字母
55
56
        }
57
   }
   int main() {
58
        scanf("\%s", s + 1);
59
        int l = strlen(s + 1);
60
        for (int i = 1; i <= l; ++i) {
   add(s[i] - 'a');</pre>
61
62
        }
63
        int T;
64
        dfs(1);
65
        scanf("%d",&T);
66
        while (T--){
67
            int k;
68
69
            int now = 1;
            scanf("%d",&k);
70
            while (k)
71
72
            {
                if (!k) break;
73
                for (int i = 0; i \le 26; ++i) {
74
                    if (ch[now][i]) {
75
76
                         if (count[ch[now][i]] >= k) {
                                                            //因为不能往前找, 所以找到的第一个大于k
       的就肯定是属于答案的子串中
77
                             putchar(i+'a');
                             --k;
78
79
                             now = ch[now][i];
                             break;
80
81
                        } else
82
                             k-=count[ch[now][i]];
                                                          //这个字母包含的不够多
83
                    }
                }
84
85
            }puts("");
86
        }
87
88 }
```

8.3 (SAM) 多串 lcs

```
// Created by acer on 2021/2/16.
   //
3
4 //判断子串,不同子串个数,所有子串字典序第i大,最长公共子串
5
6 #include <cstring>
7 #include <iostream>
  #include "string"
9
10
#define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 1e5 + 10;
14 int mxlen[MAXN<<1];</pre>
15 int anslen[MAXN<<1];</pre>
16 int len[MAXN << 1];</pre>
17 int ch[MAXN << 1][27];</pre>
18 int fa[MAXN << 1];</pre>
19 int id[MAXN << 1];</pre>
20 int last = 1;
21 int tot = 1;
22
23 int p;
24 int visit[MAXN<<1];</pre>
   void add(int c) {
26
        p = last;
27
        last = ++tot;
28
        int np = last;
        len[np] = len[p] + 1;
29
30
        for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
31
        if (!p) fa[np] = 1;
32
        else {
33
            int q = ch[p][c];
            if (len[q] == len[p] + 1) fa[np] = q;
34
            else {
35
36
                 int nq = ++tot;
37
                 memcpy(ch[nq],ch[q],sizeof(ch[nq]));
38
                 fa[nq] = fa[q];
39
                 len[nq] = len[p] + 1;
40
                 fa[np] = fa[q] = nq;
                 for (; p && ch[p][c] == q; p = fa[p]) {
41
42
                     ch[p][c] = nq;
43
44
            }
45
46
        }
47
48
   char s[MAXN], s1[MAXN];
   int mx = 0;
   void solve() {
51
52
        p = 1;
        int len1 = strlen(s1 + 1);
53
54
        int tmp = 0;
        for (int i = 1; i <= len1; ++i) {
   int c = s1[i] - 'a';</pre>
55
56
            if (ch[p][c]) {p = ch[p][c], tmp++;}
57
```

```
else {
58
                while (p \&\& !(ch[p][c])) p = fa[p];
59
                if (!p) p = 1, tmp = 0;
60
                else \{tmp = len[p] + 1; p = ch[p][c]; \}
61
62
            visit[p]++;
63
            mxlen[p] = max(mxlen[p],tmp);
64
65
        for (int j = tot; j >= 1; --j) {
66
            int k = id[j];
67
            anslen[k] = min(anslen[k], mxlen[k]);
68
69
            if (anslen[k] && fa[k]) mxlen[fa[k]] = max(mxlen[fa[k]],len[fa[k]]);
            mxlen[k] = 0;
70
        }
71
72
    }
73
74
75
    int main() {
76
        scanf("%s", s + 1);
77
        int l = strlen(s + 1);
78
79
        for (int i = 1; i <= l; ++i) {
            add(s[i] - 'a');
80
81
82
        for (int k = 1; k \le tot; ++k) {
                                                //给每个点赋权值
            anslen[len[k]] ++;
83
84
85
        for (int m = 1; m <= tot; ++m) {</pre>
                                              //获得长度小于等于m的点的总个数, 所以必然是不会相同的,
86
        故可以用来标记。
            anslen[m] += anslen[m-1];
87
88
89
        for (int n = 1; n <= tot; ++n) { //根据点出现顺序获得拓扑序
            id[anslen[len[n]]--] = n;
90
91
        while (scanf("%s", s1 + 1)!=E0F)
92
93
        {
94
            solve();
95
        }
96
        for (int j = 1; j <= tot; ++j) {
97
            mx = max(mx,anslen[j]);
98
        printf("%d",mx);
99
100
    }
    8.4 ac 自动机
   //多模式串匹配
 3
 4 #include <queue>
 5 #include <cstdlib>
 6 #include <cmath>
   #include <cstdio>
 7
 8 #include <string>
 9 #include <cstring>
 10 #include <iostream>
11 #include <algorithm>
```

```
12 using namespace std;
   const int maxn = 500000+9;
   int trie[maxn][26]; //字典树
   int cntword[maxn]; //记录该单词出现次数
16
   int fail[maxn];
                         //失败时的回溯指针
   int cnt = 0;
17
   void insertWords(string s){
18
        int root = 0;
19
20
        for(int i=0;i<s.size();i++){</pre>
            int next = s[i] - 'a';
21
22
            if(!trie[root][next])
23
                trie[root][next] = ++cnt;
            root = trie[root][next];
24
25
        }
26
        cntword[root]++;
   }
27
   void getFail(){
28
        queue <int>q;
29
        for(int i=0;i<26;i++){</pre>
30
                                      //将第二层所有出现了的字母扔进队列
            if(trie[0][i]){
31
                fail[trie[0][i]] = 0;
32
                q.push(trie[0][i]);
33
            }
34
35
36
        while(!q.empty()){
            int now = q.front();
37
            q.pop();
38
            for(int i=0;i<26;i++){</pre>
39
                if(trie[now][i]){
40
                     fail[trie[now][i]] = trie[fail[now]][i];
41
42
                     q.push(trie[now][i]);
43
                else trie[now][i] = trie[fail[now]][i];
44
            }
45
        }
46
   }
47
48
49
   int query(string s){
        int now = 0, ans = 0;
50
        for(int i=0;i<s.size();i++){</pre>
51
52
            now = trie[now][s[i]-'a'];
            for(int j=now; j && cntword[j]!=-1; j=fail[j]){
53
                ans += cntword[j];
54
55
                cntword[j] = -1;
            }
56
57
        return ans;
58
59
  void init()
60
61
   {
62
        for(int i = 0 ; i <= cnt ; i++)</pre>
63
        {
            memset(trie[i], 0, sizeof(trie[i]));
64
65
        memset(cntword,0,sizeof(cntword));
66
        cnt = 0;
67
68
   }
```

8.5 kmp

```
1 //
2 // Created by acer on 2021/2/8.
3 //
4 #define int long long
5 const int maxn = 1000000;
6 int nxt[maxn];
   void search(string s)
7
8
        int k=-1;
9
        nxt[0]=-1;
10
        int j=0;
11
12
        while(j<s.length())</pre>
13
            if(k<0 \mid | s[k]==s[j])
14
15
            {
16
                 j++;
17
                 k++;
                 nxt[j] = k;
18
            }
19
            else {
20
21
                 k=nxt[k];
            }
22
23
24
        }
25 }
```

8.6 二维字符串哈希

```
#include <iostream>
   #include <algorithm>
3
   #include <unordered_map>
4
5 using namespace std;
6
   typedef unsigned long long ULL;
7
8
9
   const int N = 1010, M = N * N, P = 131;
10
11
   int n, m, a, b;
   ULL hashv[N][N], p[M];
12
13
   char str[N];
14
15 ULL calc(ULL f[], int l, int r)
16
   {
        return f[r] - f[l - 1] * p[r - l + 1];
17
   }
18
19
  int main()
20
21
22
        scanf("%d%d%d%d", &n, &m, &a, &b);
23
24
        p[0] = 1;
        for (int i = 1; i \le n * m; i ++ ) p[i] = p[i - 1] * P;
25
26
        for (int i = 1; i <= n; i ++ )
27
28
        {
```

```
scanf("%s", str + 1);
29
            for (int j = 1; j <= m; j ++ ) hashv[i][j] = hashv[i][j - 1] * P + str[j] - '0'
30
       ,}
31
32
       unordered_map<ULL, int> S;
33
34
       for (int i = b; i <= m; i ++ )
35
       {
           ULL s = 0;
36
            int l = i - b + 1, r = i;
37
38
            for (int j = 1; j <= n; j ++ )
39
                s = s * p[b] + calc(hashv[j], l, r);
40
                if (j - a > 0) s -= calc(hashv[j - a], l, r) * p[a * b];
41
                if (j >= a) S[s] = 1;
42
           }
43
       }
44
45
46
       int Q;
       scanf("%d", &Q);
47
       while (Q -- )
48
49
           ULL s = 0;
50
            for (int i = 0; i < a; i ++)
51
52
            {
                scanf("%s", str);
53
                for (int j = 0; j < b; j ++ ) s = s * P + str[j] - '0';
54
55
            if (S[s]) puts("1");
56
            else puts("0");
57
       }
58
59
60
       return 0;
61 }
   8.7 后缀自动机
2 // Created by acer on 2021/2/16.
3 //
4 //判断子串,不同子串个数,所有子串字典序第i大,最长公共子串
5
  #include <cstring>
6
7
   #include <iostream>
   #include "string"
9
10
#define mem(x, i) memset(x,i,sizeof(x))
12 using namespace std;
13 const int MAXN = 3e5 + 10;
15 int len[MAXN << 1];</pre>
16 int ch[MAXN << 1][27];</pre>
17 int fa[MAXN << 1];</pre>
18 int last = 1;
19 int tot = 1;
20 int p;
21
```

```
void add(int c) {
22
23
       p = last;
24
       last = ++tot;
25
       int np = last;
26
       len[np] = len[p] + 1;
       for (; p && !(ch[p][c]); p = fa[p]) ch[p][c] = np;
27
       if (!p) fa[np] = 1;
28
29
       else {
30
            int q = ch[p][c];
            if (len[q] == len[p] + 1) fa[np] = q;
31
            else {
32
33
                int nq = ++tot;
                memcpy(ch[nq],ch[q],sizeof(ch[nq]));
34
                fa[nq] = fa[q];
35
                len[nq] = len[p] + 1;
36
                fa[np] = fa[q] = nq;
37
                for (; p && ch[p][c] == q; p = fa[p]) {
38
39
                    ch[p][c] = nq;
40
                }
            }
41
42
       }
43
44
45 }
46
   char s[MAXN], s1[MAXN];
47
48
   int solve() {
49
       int len1 = strlen(s1 + 1);
50
        int ans =0; int tmp =0;
51
52
       for (int i = 1; i <= len1; ++i) {
53
            int c = s1[i] - 'a';
54
            if (ch[p][c]) p = ch[p][c], tmp++;
55
            else {
56
                while (p && !(ch[p][c]))
57
                    p = fa[p];
58
59
                if (!p) p = 1, tmp = 0;
60
                else {
                    tmp = len[p] + 1;
61
                    p = ch[p][c];
62
                }
63
64
            ans = max(ans,tmp);
65
66
       return ans;
67
  }
68
  int id[MAXN<<1];</pre>
  int weig[MAXN << 1];</pre>
71
   void getTuopu()
72
   {
73
       for (int k = 1; k \le tot; ++k) {
                                                //给每个点赋权值
74
            weig[len[k]] ++;
75
       for (int m = 1; m <= tot; ++m) {</pre>
76
                                              //获得长度大于等于m的点的总个数, 所以必然是不会相同的,
       故可以用来标记。
77
            weig[m] += weig[m-1];
78
79
       for (int n = 1; n <= tot; ++n) { //根据点出现顺序获得拓扑序
```

```
id[weig[len[n]]--] = n;
80
              cout << weig[len[n]]+1 <<' ' << id[weig[len[n]]+1] << endl;</pre>
81 //
82
          for (int j = 1; j <= tot; ++j) {
    cout << id[j] << endl;</pre>
83 //
84 //
85 //
86 }
87 int main() {
88 //
          freopen("in.txt","r",stdin);
89 //
          freopen("out.txt","w",stdout);
        scanf("%s", s + 1);
90
91
        int l = strlen(s + 1);
        for (int i = 1; i <= l; ++i) {
92
            add(s[i] - 'a');
93
94
        scanf("%s", s1 + 1);
95
96
        cout << solve();</pre>
97 }
         回文自动机
   8.8
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int MAXN = 500000, MAXK = 26;
4 char str[ MAXN + 5 ];long long weight[MAXN+5];
   struct Palindrome_Automaton{
6
        int Size //总节点
        , Last , Root0 , Root1 , Trans[ MAXN + 5 ][ MAXK + 5 ] , Fail[ MAXN + 5 ]; long long Len[ MAXN + 5 ];
7
8
9
10
        Palindrome_Automaton( ) {
            Root0 = Size ++ , Root1 = Size; Last = Root1;
Len[ Root0 ] = 0  , Fail[ Root0 ] = Root1;
11
12
            Len[Root1] = -1, Fail[Root1] = Root1;
13
14
        void Extend( int ch , int dex ) {
15
            int u = Last;
17
            while (str[ dex - Len[ u ] - 1 ] != str[ dex ] ) u = Fail[ u ]; //找到合格的后缀
            if( !Trans[ u ][ ch ] ) {
                                                //无现成的边
18
                int Newnode = ++ Size , v = Fail[ u ];
19
                                                             //防止取掉整串
                Len[ Newnode ] = Len[ u ] + 2;
20
                while (str[ dex - Len[ v ] - 1 ] != str[ dex ] ) v = Fail[ v ];
21
                                                                                         //给他找
         -个fail指针
                Fail[ Newnode ] = Trans[ v ][ ch ] , Trans[ u ][ ch ] = Newnode;
22
23
            Last = Trans[u][ch];
24
25
            weight[Last]++;
26
        }
27
        void Build( char *str ) {
28
29
            int len = strlen( str );
            for( int i = 0 ; i < len ; i ++ ) {
30
                 Extend( str[ i ] - 'a' + 1 , i );
31
32
        }
33
34
        void getWeight()
35
36
        {
```

```
long long ma =0;
37
           for (int k = Size; k >= 0; --k) {
38
               weight[Fail[k]] += weight[k];
39
40
41
           for (int j = 0; j <= Size; ++j) {
               ma = max(ma,Len[j]*weight[j]);
42
43
           cout << ma << endl;</pre>
44
       }
45
46
   }PAM;
47
48
   signed main( ) {
49
       scanf("%s", str );
PAM.Build( str );
50
51
       PAM.getWeight();
52
53
       return 0;
54 }
   8.9
        拓展 kmp
1 #include<bits/stdc++.h>
2 #include "ext/pb_ds/assoc_container.hpp"
3 //using namespace __gnu_pbds;
4 using namespace std;
5 const int N = 1e4 + 10;
6 //typedef long long ll;
7 #define int long long
8 using namespace std;
9 const int MAXN = 2e7 + 5;
10 char p[MAXN], s[MAXN];
int pl, sl, z[MAXN], ext[MAXN];
12 //z[i]:
                   i ~ strlen (B) -1
               B串
                                             部分与 B自身的最长相同前缀
13 //
14 //ext[i]:
                  A串 i ~ strlen (A) -1
                                                  部分与B的最长相同前缀,也就是我们要求的东西。
15 void getZ() {
16
       z[0] = pl;//从0号位置开始, LCP就是全部字符串
17
       //从1开始, 先暴力算
18
       int now = 0;
19
       while (now + 1 < pl && p[now] == p[now + 1]) now++;
       z[1] = now;
20
       int p0 = 1;
21
22
       //p0是最远情况的起点
       for (int i = 2; i < pl; ++i) {
23
24
           //p0+z[p0]是此时最远处
           //i-p0对应着主串的i,加i就是能到的距离
25
           if (i + z[i - p0] < p0 + z[p0]) {
26
               z[i] = z[i - p0];//第一种情况
27
           } else {
28
               now = p0 + z[p0] - i;
29
               now = max(now, 011);
30
               while (now + i < pl && p[now] == p[now + i]) now++;
31
32
               z[i] = now;
33
               p0 = i;
           }
34
35
       }
  }
36
37
```

```
void exkmp() {
38
39
        getZ();
        //先暴力算ext[0]
40
41
        int now = 0;
42
        while (now < pl && now < sl && p[now] == s[now]) now++;
        ext[0] = now;
43
        int p0 = 0;
44
        for (int i = 1; i < sl; ++i) {
45
            if (i + z[i - p0] < p0 + ext[p0]) {
46
                ext[i] = z[i - p0];
47
            } else {
48
49
                now = p0 + ext[p0] - i;
                now = max(now, 0ll);//防止i太大
50
                while (now < pl && now + i < sl && p[now] == s[now + i]) now++;
51
52
                ext[i] = now;
                p0 = i;
53
            }
54
        }
55
   }
56
57
   signed main() {
58
        scanf("%s%s", s, p);
59
        pl = strlen(p);
60
61
        sl = strlen(s);
62
        exkmp();
        int ans1 = 0, ans2=0;
63
        for (int i = 0; i < pl; ++i) {
64
            ans1 ^= 1LL * (i + 1) * (z[i] + 1);
65
               cout << z[i] << ' ';
   //
66
        }
67
   //
          cout << '\n';</pre>
68
        for (int i = 0; i < sl; ++i) {
69
            ans2 ^= 1LL * (i + 1) * (ext[i] + 1);
70
   //
              cout << ext[i] << ' ';
71
        }
72
73
74
        printf("%lld\n%lld\n", ans1, ans2);
75
        return 0;
76 }
   8.10
          字典树
   #include <bits/stdc++.h>
   using namespace std;
   const int N = 100010;
4 int tire[26*N][26], n, m;
5 int cnt, level[N*26];
6
   void insert(string s) {
7
8
        int p = 0;
        for (int i = 0; i < s.length(); i ++) {</pre>
9
            int v = s[i] - 'a';
10
            if (!tire[p][v]) tire[p][v] = ++cnt;
11
12
            p = tire[p][v];
13
        level[p] ++;
14
15
   }
16
```

```
int query(string s) {
17
        int p = 0;
18
        int res = 0;
19
        for (int i = 0; i < s.length(); i ++) {
    p = tire[p][s[i]-'a'];</pre>
20
21
22
            if (!p) break;
            res += level[p];
23
24
        }
25
        return res;
  }
26
   8.11 字符串哈希
1
2
   char s[N];
   unsigned long long f[N], p[N];
3
4
   unsigned long long get(int l, int r) {//获取哈希值
5
        return f[r] - f[l-1] * p[r-l+1];
6
7
   }
8
9 f[i] = f[i-1] * base + s[i];
   p[i] = p[i-1] * base;
10
11
12
13 /*
14
15 H(T) = H(S + T) - H(S) - p \land (length(T))
16
   */
17
   8.12 最小表示法
1 //
2 // Created by acer on 2021/2/1.
3 //
  //求循环字符串的长度为k的最小子串
4
   int MinimumRepresentation(int *s, int l)
5
6
        int i,j,k;
7
8
        i=0; j=1; k=0;
9
        while(i<l&&j<l)
10
            k=0;
11
            while(s[i+k]==s[j+k]&&k<1) k++;
12
13
            if(k==1) return i;
            if(s[i+k]>s[j+k])
14
                if(i+k+1>j) i=i+k+1;
15
16
                else i=j+1;
17
            else if(j+k+1>i) j=j+k+1;
            else j=i+1;
18
19
        if(i<l) return i;</pre>
20
        else return j;
21
22 }
```