
ACM TEMPLATE

UESTC_Lasagne

Last build at December 12, 2012

Contents

1	注意事项	5
2	字符串处理	7
2.1	*AC 自动机	7
2.1.1	指针	7
2.1.2	非指针	8
2.2	后缀数组	9
2.2.1	DC3	9
2.2.2	DA	10
2.2.3	调用	11
2.2.4	最长公共前缀	12
2.2.5	最长公共前缀大于等于某个值的区间	12
2.3	后缀三兄弟	13
2.3.1	例题	17
2.4	KMP	23
2.5	e-KMP	23
2.6	Manacher	24
2.7	不同回文串	25
2.8	* 字符串最小表示法	28
2.9	带 * 通配符的匹配	28
3	数学	31
3.1	扩展 GCD	31
3.2	模线性方程组	31
3.3	矩阵	32
3.4	康拓展开	33
3.5	FFT	34
3.6	爬山法计算器	37
3.7	线性筛	39
3.8	线性规划	40
3.9	分解质因数	43
3.9.1	米勒拉宾 + 分解因数	43
3.9.2	暴力版本	46
3.10	baby step giant step	46
3.10.1	BSGS	46
3.10.2	何老师的版	47
3.11	原根	49
3.12	逆元	50
3.13	卢卡斯	50
3.14	欧拉函数	50
3.14.1	分解质因数	50
3.14.2	一次预处理	50
3.15	费马降阶法	51
3.16	自适应 simp	52
3.17	组合数求模	53
3.18	高斯消元	54
3.19	整数拆分	55
3.20	佩尔方程	56
3.21	其它公式	58
3.21.1	Polya	58

3.21.2	拉格朗日插值法	58
3.21.3	正多面体顶点着色	59
3.21.4	求和公式	59
3.21.5	几何公式	59
3.21.6	小公式	59
3.21.7	马步问题	60
4	数据结构	62
4.1	*Splay	62
4.1.1	节点定义	62
4.1.2	维护序列	63
4.1.3	维护括号序列	67
4.2	动态树	73
4.2.1	维护点权	73
4.2.2	维护边权	78
4.3	可持久化线段树	82
4.4	treap 正式版	85
4.5	树链剖分	88
4.5.1	点权	88
4.5.2	边权	94
4.6	划分树	98
4.7	树状数组	100
5	图论	101
5.1	优先队列优化的 dijkstra	101
5.2	SAP 四版	102
5.3	费用流三版	104
5.4	网络单纯形	106
5.5	匈牙利	113
5.5.1	新版, 隐式图可解	113
5.5.2	邻接矩阵	113
5.5.3	邻接表	114
5.6	一般图最大加权匹配	114
5.7	一般图匹配带花树	116
5.8	KM	119
5.8.1	最大加权匹配	119
5.8.2	自认为正确的 Kuhn_Munkras	120
5.9	* 二维平面图的最大流	122
5.10	强联通	126
5.11	最大团以及相关知识	127
5.12	双连通分量	129
5.13	割点与桥	130
5.14	LCA	132
5.15	最优比例生成树	134
5.16	生成树计数	136
5.17	全局最小割	136
5.18	欧拉路	137
5.18.1	有向图	137
5.18.2	无向图	137
5.18.3	混合图	138
5.19	K 短路	139

5.20	稳定婚姻	141
5.21	最小树形图	143
6	计算几何	146
6.1	注意事项	146
6.2	基本函数	146
6.2.1	Point 定义	146
6.2.2	Line 定义	146
6.2.3	距离: 点到直线距离	147
6.2.4	距离: 点到线段距离	147
6.2.5	面积: 多边形	148
6.2.6	判断: 线段相交	148
6.2.7	判断: 点在线段上	148
6.2.8	判断: 点在多边形内	148
6.2.9	判断: 两凸包相交	149
6.2.10	排序: 叉积极角排序	150
6.3	三维几何	150
6.3.1	Point 定义	150
6.3.2	经度纬度转换	151
6.3.3	判断: 直线相交	151
6.3.4	判断: 线段相交	151
6.3.5	判断: 三维向量是否为 0	152
6.3.6	判断: 点在直线上	152
6.3.7	判断: 点在线段上	152
6.3.8	距离: 点到直线	152
6.3.9	夹角	152
6.4	圆	152
6.4.1	面积: 两圆相交	152
6.4.2	三角形外接圆	153
6.4.3	三角形内切圆	153
6.4.4	点对圆的两个切点	153
6.4.5	两圆公切点	154
6.4.6	两圆交点	154
6.5	三角形相关	155
6.6	矩阵	155
6.6.1	基本矩阵	155
6.6.2	刘汝佳的几何教室	155
6.7	重心	159
6.8	KD 树	159
6.8.1	例题	160
6.9	半平面交	163
6.10	凸包	164
6.11	直线与凸包求交点	165
6.12	点对凸包的两切点	166
6.13	三维凸包	168
6.14	旋转卡壳	173
6.14.1	单个凸包	173
6.14.2	两个凸包	174
6.14.3	外接矩形	174
6.15	三角形内点个数	175
6.15.1	无三点共线	175

6.15.2	有三点共线且点有类别之分	177
6.16	最近点对	179
6.16.1	类快排算法	179
6.16.2	随机增量法	179
6.17	多圆面积并	181
6.17.1	去重	181
6.17.2	圆并	182
6.18	一个圆与多边形面积交	184
6.19	精度问题	186
6.19.1	浮点数为啥会有精度问题	186
6.19.2	eps	186
6.19.3	eps 带来的函数越界	186
6.19.4	输出陷阱 I	187
6.19.5	输出陷阱 II	187
6.19.6	范围越界	187
6.19.7	关于 set	187
6.19.8	输入值波动过大	187
6.19.9	一些建议	187
7	搜索	188
7.1	Dancing Links	188
7.1.1	估价函数	188
7.1.2	DLX	188
8	动态规划	192
8.1	斜率优化	192
8.2	RMQ 二版	193
8.3	二维 LIS	193
8.4	插头 DP	194
9	杂物	200
9.1	高精度数	200
9.2	整数外挂	201
9.3	Java	201
9.3.1	文件操作	201
9.3.2	优先队列	202
9.3.3	Map	202
9.3.4	sort	202
9.4	hashmap	203
9.5	C++&STL 常用函数	204
9.5.1	lower_bound/upper_bound	204
9.5.2	rotate	205
9.5.3	nth_element	205
9.5.4	bitset	205
9.5.5	multimap	206
9.6	位运算	208
9.6.1	基本操作	208
9.6.2	枚举长为 n 含 k 个 1 的 01 串	208
9.7	其它	209
9.7.1	对跑脚本	209

1 注意事项

输入输出格式？调试信息？初始化？算术溢出？数组大小？

左右端点范围？ $\text{acos}/\text{asin}/\text{sqrt}$ 函数定义域？精度问题？

二分答案？暴力？单调性？凸性？块状结构？函数式？对偶问题？

排序的时候注意一下是否需要记录排序前的位置！

使用 `map` 进行映射的时候，不要用下面这种不安全写法

```
1 if (mp.find(s) == mp.end())
2     mp[s] = mp.size()-1;//挂成狗
3
4 if (mp.find(s) == mp.end())
5 {
6     int tmp = mp.size();
7     mp[s] = tmp;//正确
8 }
```

10^6 数量级慎用后缀数组

TLE 的时候要冷静哟。。

思考的时候结合具体步骤来的话会体会到一些不同的东西

C++ 与 G++ 是很不一样的。。

`map` 套字符串是很慢的。。

栈会被记录内存。。

浮点数最短路要注意取 \leq 来判断更新。。。

注意 long long

不要相信.size()

重复利用数组时小心数组范围

先构思代码框架每当实际拍马框架变化时停手重新思考

有时候四边形不等式也是帮得上忙的 dp 优化是可以水的

结构体里面带数组会非常慢, 有时候 BFS 把数组压成数字会快很多。

```
1 void fun(int a[])
2 {
3     printf("%d\n", sizeof(a));
4 }
```

结果是 sizeof(a[0]), 如果传数组指针然后要清空的话不要用 sizeof。

sqrt 某些时候会出现 sqrt(-0.00) 的问题。

将 code::blocks 的默认终端改成 gnome-terminal

```
1 gnome-terminal -t $TITLE -x
```

最小割割集找法在残量网络中从源点出发能到的点集记为 S 原图中 S 到 S' 的边即是最小割集

double 全局变量初始值可能不是 0

2 字符串处理

2.1 *AC 自动机

2.1.1 指针

```

1  const int CHAR=26;
2  const int TOTLEN=500000;
3  const int MAXLEN=1000000;
4  struct Vertex
5  {
6      Vertex *fail,*next[CHAR];
7      Vertex(){}
8      Vertex(bool flag)//为什么要这样写?
9      {
10         fail=0;
11         memset(next,0,sizeof(next));
12     }
13 };
14 int size;
15 Vertex vertex[TOTLEN+1];
16 void init()
17 {
18     vertex[0]=Vertex(0);
19     size=1;
20 }
21 void add(Vertex *pos,int cha)
22 {
23     vertex[size]=Vertex(0);
24     pos->next[cha]=&vertex[size++];
25 }
26 void add(vector<int> s)
27 {
28     int l=s.size();
29     Vertex *pos=&vertex[0];
30     for (int i=0; i<l; i++)
31     {
32         if (pos->next[s[i]]==NULL)
33             add(pos,s[i]);
34         pos=pos->next[s[i]];
35     }
36 }
37 void bfs()
38 {
39     queue<Vertex *> que;
40     Vertex *u=&vertex[0];
41     for (int i=0; i<CHAR; i++)
42         if (u->next[i]!=NULL)
43         {
44             que.push(u->next[i]);
45             u->next[i]->fail=u;
46         }

```

```

47     else
48         u->next[i]=u;
49     u->fail=NULL;
50     while (!que.empty())
51     {
52         u=que.front();
53         que.pop();
54         for (int i=0; i<CHAR; i++)
55             if (u->next[i]!=NULL)
56             {
57                 que.push(u->next[i]);
58                 u->next[i]->fail=u->fail->next[i];
59             }
60         else
61             u->next[i]=u->fail->next[i];
62     }
63 }

```

2.1.2 非指针

```

1  struct Trie
2  {
3      int next[50][10],fail[50];
4      bool end[50];
5      int L,root;
6
7      int newNode()
8      {
9          for (int i = 0;i < 10;i++)
10             next[L][i] = -1;
11         end[L] = false;
12         return L++;
13     }
14
15     void Init()
16     {
17         L = 0;
18         root = newNode();
19     }
20
21     void Insert(char s[])
22     {
23         int now = root;
24         for (int i = 0;s[i] != 0;i++)
25         {
26             if (next[now][s[i]-'0'] == -1)
27                 next[now][s[i]-'0'] = newNode();
28             now = next[now][s[i]-'0'];
29         }
30         end[now] = true;
31     }
32 }

```

```

33 void Build()
34 {
35     queue<int> Q;
36     for (int i = 0; i < 10; i++)
37         if (next[root][i] == -1)
38             next[root][i] = root;
39         else
40         {
41             fail[next[root][i]] = root;
42             Q.push(next[root][i]);
43         }
44     while (!Q.empty())
45     {
46         int now = Q.front();
47         Q.pop();
48         end[now] |= end[fail[now]];
49         for (int i = 0; i < 10; i++)
50             if (next[now][i] == -1)
51                 next[now][i] = next[fail[now]][i];
52             else
53             {
54                 fail[next[now][i]] = next[fail[now]][i];
55                 Q.push(next[now][i]);
56             }
57     }
58 }
59 };

```

2.2 后缀数组

2.2.1 DC3

所有下标都是 $0 \sim n-1$, $height[0]$ 无意义。

```

1 //所有相关数组都要开三倍
2 const int maxn = 300010;
3 # define F(x) ((x)/3+((x)%3==1?0:tb))
4 # define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
5 int wa[maxn * 3], wb[maxn * 3], wv[maxn * 3], ws[maxn * 3];
6 int c0(int *r, int a, int b)
7 {
8     return
9     r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
10 }
11 int c12(int k, int *r, int a, int b)
12 {
13     if (k == 2)
14         return r[a] < r[b] || r[a] == r[b] && c12(1, r, a + 1, b + 1);
15     else return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b + 1];
16 }
17 void sort(int *r, int *a, int *b, int n, int m)
18 {
19     int i;

```

```

20  for (i = 0; i < n; i++) wv[i] = r[a[i]];
21  for (i = 0; i < m; i++) ws[i] = 0;
22  for (i = 0; i < n; i++) ws[wv[i]]++;
23  for (i = 1; i < m; i++) ws[i] += ws[i - 1];
24  for (i = n - 1; i >= 0; i--) b[--ws[wv[i]]] = a[i];
25  return;
26 }
27 void dc3(int *r, int *sa, int n, int m)
28 {
29     int i, j, *rn = r + n;
30     int *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;
31     r[n] = r[n + 1] = 0;
32     for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
33     sort(r + 2, wa, wb, tbc, m);
34     sort(r + 1, wb, wa, tbc, m);
35     sort(r, wa, wb, tbc, m);
36     for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
37         rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
38     if (p < tbc) dc3(rn, san, tbc, p);
39     else for (i = 0; i < tbc; i++) san[rn[i]] = i;
40     for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i] * 3;
41     if (n % 3 == 1) wb[ta++] = n - 1;
42     sort(r, wb, wa, ta, m);
43     for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;
44     for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
45         sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
46     for (; i < ta; p++) sa[p] = wa[i++];
47     for (; j < tbc; p++) sa[p] = wb[j++];
48 }
49 //str 和 sa 也要三倍
50 void da(int str[], int sa[], int rank[], int height[], int n, int m)
51 {
52     for (int i = n; i < n * 3; i++)
53         str[i] = 0;
54     dc3(str, sa, n + 1, m);
55     int i, j, k;
56     for (i = 0; i < n; i++)
57     {
58         sa[i] = sa[i + 1];
59         rank[sa[i]] = i;
60     }
61     for (i = 0, j = 0, k = 0; i < n; height[rank[i + 1]] = k)
62         if (rank[i] > 0)
63             for (k ? k-- : 0, j = sa[rank[i] - 1];
64                 i + k < n && j + k < n && str[i + k] == str[j + k];
65                 k++);
66 }

```

2.2.2 DA

这份似乎就没啥要注意的了。

```

1  const int maxn = 200010;
2  int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];
3
4  bool cmp(int *r,int n,int a,int b,int l)
5  {
6      return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
7  }
8  void da(int str[],int sa[],int rank[],int height[],int n,int m)
9  {
10     int *s = str;
11     int *x=wx,*y=wy,*t,p;
12     int i,j;
13     for(i=0; i<m; i++)wss[i]=0;
14     for(i=0; i<n; i++)wss[x[i]=s[i]]++;
15     for(i=1; i<m; i++)wss[i]+=wss[i-1];
16     for(i=n-1; i>=0; i--)sa[--wss[x[i]]]=i;
17     for(j=1,p=1; p<n && j<n; j*=2,m=p)
18     {
19         for(i=n-j,p=0; i<n; i++)y[p++]=i;
20         for(i=0; i<n; i++)if(sa[i]-j>=0)y[p++]=sa[i]-j;
21         for(i=0; i<n; i++)wv[i]=x[y[i]];
22         for(i=0; i<m; i++)wss[i]=0;
23         for(i=0; i<n; i++)wss[wv[i]]++;
24         for(i=1; i<m; i++)wss[i]+=wss[i-1];
25         for(i=n-1; i>=0; i--)sa[--wss[wv[i]]]=y[i];
26         for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0; i<n; i++)
27             x[sa[i]]=cmp(y,n,sa[i-1],sa[i],j)?p-1:p++;
28     }
29     for(int i=0; i<n; i++) rank[sa[i]]=i;
30     for(int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
31         if(rank[i]>0)
32             for(k?k--:0,j=sa[rank[i]-1];
33                 i+k < n && j+k < n && str[i+k]==str[j+k];
34                 k++);
35 }

```

2.2.3 调用

注意几个数组的下标是不同的

```

1  char s[maxn];
2  int str[maxn],sa[maxn],rank[maxn],height[maxn];
3
4  int main()
5  {
6      scanf("%s",s);
7      int len = strlen(s);
8      for (int i = 0; i <= len; i++)
9          str[i] = s[i];
10     da(str,sa,rank,height,len,128);
11
12     for (int i = 0; i < len; i++)

```

```

13 {
14     printf("sa=%d,height=%d,s=%s\n",sa[i],height[i],s+sa[i]);
15 }
16 return 0;
17 }

```

2.2.4 最长公共前缀

记得不要忘记调用 lcpinit!

```

1 int f[maxn][20];
2 int lent[maxn];
3 void lcpinit()
4 {
5     int i,j;
6     int n = len,k = 1,l = 0;
7     for (i = 0; i < n; i++)
8     {
9         f[i][0] = height[i];
10        if (i+1 > k*2)
11        {
12            k *= 2;
13            l++;
14        }
15        lent[i+1] = l;
16    }
17    for (j = 1; (1<<j)-1<n; j++)
18        for (i = 0; i+(1<<j)-1<n; i++)
19            f[i][j] = min(f[i][j-1],f[i+(1<<(j-1))][j-1]);
20 }
21 int lcp(int x,int y)
22 {
23     if (x > y) swap(x,y);
24     if (x == y)
25         return x-sa[x]; //自己和自己当然是自己的长度啦lcp
26     x++;
27     int k = lent[y-x+1];
28     return min(f[x][k],f[y-(1<<k)+1][k]);
29 }

```

2.2.5 最长公共前缀大于等于某个值的区间

```

1 void getinterv(int pos,int comlen,int& pl,int& pr)
2 {
3     int l,r,mid,cp;
4     l = 0;
5     r = pos;
6     while (l < r)
7     {
8         mid = l+r>>1;
9         cp = lcp(mid,pos);

```

```

10     if (cp < comlen)
11         l = mid+1;
12     else
13         r = mid;
14 }
15 pl = l;
16
17 l = pos;
18 r = len-1;
19 while (l < r)
20 {
21     mid = l+r+1>>1;
22     cp = lcp(pos,mid);
23     if (cp < comlen)
24         r = mid-1;
25     else
26         l = mid;
27 }
28 pr = l;
29 }

```

2.3 后缀三兄弟

```

1  #include <cstdio>
2  #include <cstring>
3  #include <algorithm>
4  using namespace std;
5  const int CHAR = 26;
6  const int MAXN = 100000;
7  struct SAM_Node
8  {
9      SAM_Node *fa,*next[CHAR];
10     int len;
11     int id,pos;
12     SAM_Node() {}
13     SAM_Node(int _len)
14     {
15         fa = 0;
16         len = _len;
17         memset(next,0,sizeof(next));
18     }
19 };
20 SAM_Node SAM_node[MAXN * 2], *SAM_root, *SAM_last;
21 int SAM_size;
22 SAM_Node *newSAM_Node(int len)
23 {
24     SAM_node[SAM_size] = SAM_Node(len);
25     SAM_node[SAM_size].id=SAM_size;
26     return &SAM_node[SAM_size++];
27 }
28 SAM_Node *newSAM_Node(SAM_Node *p)
29 {

```

```

30     SAM_node[SAM_size] = *p;
31     SAM_node[SAM_size].id=SAM_size;
32     return &SAM_node[SAM_size++];
33 }
34 void SAM_init()
35 {
36     SAM_size = 0;
37     SAM_root = SAM_last = newSAM_Node(0);
38     SAM_node[0].pos=0;
39 }
40 void SAM_add(int x,int len)
41 {
42     SAM_Node *p = SAM_last, *np = newSAM_Node(p->len + 1);
43     np->pos=len;
44     SAM_last = np;
45     for (; p && !p->next[x]; p = p->fa)
46         p->next[x] = np;
47     if (!p)
48     {
49         np->fa = SAM_root;
50         return ;
51     }
52     SAM_Node *q = p->next[x];
53     if (q->len == p->len + 1)
54     {
55         np->fa = q;
56         return ;
57     }
58     SAM_Node *nq = newSAM_Node(q);
59     nq->len = p->len + 1;
60     q->fa = nq;
61     np->fa = nq;
62     for (; p && p->next[x] == q; p = p->fa)
63         p->next[x] = nq;
64 }
65 void SAM_build(char *s)
66 {
67     SAM_init();
68     int l = strlen(s);
69     for (int i = 0; i < l; i++)
70         SAM_add(s[i] - 'a',i+1);
71 }
72
73 SAM_Node * SAM_add(SAM_Node *p, int x, int len)
74 {
75     SAM_Node *np = newSAM_Node(p->len + 1);
76     np->pos = len;
77     SAM_last = np;
78     for (; p && !p->next[x]; p = p->fa)
79         p->next[x] = np;
80     if (!p)

```



```

81     {
82         np->fa = SAM_root;
83         return np;
84     }
85     SAM_Node *q = p->next[x];
86     if (q->len == p->len + 1)
87     {
88         np->fa = q;
89         return np;
90     }
91     SAM_Node *nq = newSAM_Node(q);
92     nq->len = p->len + 1;
93     q->fa = nq;
94     np->fa = nq;
95     for (; p && p->next[x] == q; p = p->fa)
96         p->next[x] = nq;
97     return np;
98 }
99 void SAM_build(char *s)//多串建立 注意SAM_init()的调用
100 {
101     int l = strlen(s);
102     SAM_Node *p = SAM_root;
103     for (int i = 0; i < l; i++)
104     {
105         if (!p->next[s[i] - 'a'] || !(p->next[s[i] - 'a']->len == i +
106             1))
107             p = SAM_add(p, s[i] - 'a', i + 1);
108         else
109             p = p->next[s[i] - 'a'];
110     }
111 }
112 struct ST_Node
113 {
114     ST_Node *next[CHAR], *fa;
115     int len, pos;
116 } ST_node[MAXN*2], *ST_root;
117 int Sufpos[MAXN];
118 void ST_add(int u, int v, int chr, int len)
119 {
120     ST_node[u].next[chr] = &ST_node[v];
121     ST_node[v].len = len;
122 }
123 void init(int n)
124 {
125     for (int i = 0; i < n; i++)
126     {
127         ST_node[i].pos = -1;
128         ST_node[i].fa = 0;
129         memset(ST_node[i].next, 0, sizeof(ST_node[i].next));
130     }

```

```

131     ST_node[0].pos=0;
132     ST_root=&ST_node[0];
133 }
134 void ST_build(char *s)
135 {
136     int n=strlen(s);
137     reverse(s,s+n);
138     SAM_build(s);
139     init(SAM_size);
140     for (int i=1;i<SAM_size;i++)
141     {
142         ST_add(SAM_node[i].fa->id,
143             SAM_node[i].id,
144             s[SAM_node[i].pos-SAM_node[i].fa->len-1]-'a',
145             SAM_node[i].len-SAM_node[i].fa->len);
146         if (SAM_node[i].pos==SAM_node[i].len)
147         {
148             Sufpos[n-SAM_node[i].pos+1]=i;
149             ST_node[i].pos=n-SAM_node[i].pos+1;
150         }
151     }
152 }
153
154 int rank[MAXN],sa[MAXN+1];
155 int height[MAXN];
156 int L;
157 void ST_dfs(ST_Node *p)
158 {
159     if (p->pos!=-1)
160         sa[L++]=p->pos;
161     for (int i=0;i<CHAR;i++)
162         if (p->next[i])
163             ST_dfs(p->next[i]);
164 }
165 char s[MAXN+1];
166 int main()
167 {
168     gets(s);
169     ST_build(s);
170     L=0;
171     ST_dfs(ST_root);
172     int n=strlen(s);
173     for (int i=0; i<n; i++)
174         sa[i]=sa[i+1]-1;
175     for (int i=0; i<n; i++)
176         rank[sa[i]]=i;
177     reverse(s,s+n);
178     for (int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
179         if (rank[i])
180             for (k?k--:0,j=sa[rank[i]-1]; s[i+k]==s[j+k]; k++);
181 }

```

2.3.1 例题

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  using namespace std;
6
7  const int CHAR = 26;
8  const int MAXN = 100000;
9
10 struct SAM_Node
11 {
12     SAM_Node *fa,*next[CHAR];
13     int len;
14     int id;
15     int mat[9];
16     SAM_Node() {}
17     SAM_Node(int _len)
18     {
19         fa = 0;
20         len = _len;
21         memset(mat,0,sizeof(mat));
22         memset(next,0,sizeof(next));
23     }
24 };
25 SAM_Node SAM_node[MAXN*2],*SAM_root,*SAM_last;
26 int SAM_size;
27 SAM_Node *newSAM_Node(int len)
28 {
29     SAM_node[SAM_size] = SAM_Node(len);
30     SAM_node[SAM_size].id = SAM_size;
31     return &SAM_node[SAM_size++];
32 }
33 SAM_Node *newSAM_Node(SAM_Node *p)
34 {
35     SAM_node[SAM_size] = *p;
36     SAM_node[SAM_size].id = SAM_size;
37     return &SAM_node[SAM_size++];
38 }
39 void SAM_init()
40 {
41     SAM_size = 0;
42     SAM_root = SAM_last = newSAM_Node(0);
43 }
44 void SAM_add(int x,int len)
45 {
46     SAM_Node *p = SAM_last,*np = newSAM_Node(p->len+1);
47     SAM_last = np;
48     for (; p&&!p->next[x]; p=p->fa)
49         p->next[x] = np;

```

```

50     if (!p)
51     {
52         np->fa = SAM_root;
53         return;
54     }
55     SAM_Node *q = p->next[x];
56     if (q->len == p->len+1)
57     {
58         np->fa = q;
59         return;
60     }
61     SAM_Node *nq = newSAM_Node(q);
62     nq->len = p->len+1;
63     q->fa = nq;
64     np->fa = nq;
65     for (; p&& p->next[x] == q; p = p->fa)
66         p->next[x] = nq;
67 }
68 int getid(char ch)
69 {
70     return ch-'a';
71 }
72 void SAM_build(char *s)
73 {
74     SAM_init();
75     int l = strlen(s);
76     for (int i = 0; i < l; i++)
77         SAM_add(getid(s[i]),i+1);
78 }
79 char s[10][MAXN+1];
80 int ans;
81 int head[MAXN*2];
82 struct Edge
83 {
84     int to,next;
85 } edge[MAXN*2];
86 int M;
87 int n;
88 void add_edge(int u,int v)
89 {
90     edge[M].to=v;
91     edge[M].next=head[u];
92     head[u]=M++;
93 }
94 void dfs(int u)
95 {
96     for (int i=head[u]; i!=-1; i=edge[i].next)
97     {
98         int v=edge[i].to;
99         dfs(v);
100     }

```

```

101     SAM_node[u].mat[j]=max(SAM_node[v].mat[j],SAM_node[u].mat[j]);
102 }
103 int tmp=SAM_node[u].len;
104 for (int i=0; i<n-1; i++)
105     tmp=min(tmp,SAM_node[u].mat[i]);
106 ans=max(ans,tmp);
107 }
108 int main()
109 {
110
111     while (scanf("%s",s[n])!=EOF)
112         n++;
113     int L=strlen(s[0]);
114     ans=M=0;
115     SAM_build(s[0]);
116     for (int j=1; j<n; j++)
117     {
118         int l=strlen(s[j]),len=0;
119         SAM_Node *p=SAM_root;
120         for (int i=0; i<l; i++)
121         {
122             if (p->next[getid(s[j][i])])
123             {
124                 p=p->next[getid(s[j][i])];
125                 p->mat[j-1]=max(p->mat[j-1],++len);
126             }
127             else
128             {
129                 while (p && !p->next[getid(s[j][i])])
130                     p=p->fa;
131                 if (!p)
132                 {
133                     p=SAM_root;
134                     len=0;
135                 }
136                 else
137                 {
138                     len=p->len+1;
139                     p=p->next[getid(s[j][i])];
140                 }
141                 p->mat[j-1]=max(p->mat[j-1],len);
142             }
143         }
144     }
145     memset(head,-1,4*SAM_size);
146     for (int i=1; i<SAM_size; i++)
147         add_edge(SAM_node[i].fa->id,i);
148     dfs(0);
149     printf("%d\n",ans);
150     return 0;
151 }

```

LCS2

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  using namespace std;
6
7  const int CHAR = 26;
8  const int MAXN = 100000;
9
10 struct SAM_Node
11 {
12     SAM_Node *fa,*next[CHAR];
13     int len;
14     int id;
15     int mat[9];
16     SAM_Node() {}
17     SAM_Node(int _len)
18     {
19         fa = 0;
20         len = _len;
21         memset(mat,0,sizeof(mat));
22         memset(next,0,sizeof(next));
23     }
24 };
25 SAM_Node SAM_node[MAXN*2],*SAM_root,*SAM_last;
26 int SAM_size;
27 SAM_Node *newSAM_Node(int len)
28 {
29     SAM_node[SAM_size] = SAM_Node(len);
30     SAM_node[SAM_size].id = SAM_size;
31     return &SAM_node[SAM_size++];
32 }
33 SAM_Node *newSAM_Node(SAM_Node *p)
34 {
35     SAM_node[SAM_size] = *p;
36     SAM_node[SAM_size].id = SAM_size;
37     return &SAM_node[SAM_size++];
38 }
39 void SAM_init()
40 {
41     SAM_size = 0;
42     SAM_root = SAM_last = newSAM_Node(0);
43 }
44 void SAM_add(int x,int len)
45 {
46     SAM_Node *p = SAM_last,*np = newSAM_Node(p->len+1);
47     SAM_last = np;
48     for (; p&&!p->next[x]; p=p->fa)
49         p->next[x] = np;

```

```

50     if (!p)
51     {
52         np->fa = SAM_root;
53         return;
54     }
55     SAM_Node *q = p->next[x];
56     if (q->len == p->len+1)
57     {
58         np->fa = q;
59         return;
60     }
61     SAM_Node *nq = newSAM_Node(q);
62     nq->len = p->len+1;
63     q->fa = nq;
64     np->fa = nq;
65     for (; p&& p->next[x] == q; p = p->fa)
66         p->next[x] = nq;
67 }
68 int getid(char ch)
69 {
70     return ch-'a';
71 }
72 void SAM_build(char *s)
73 {
74     SAM_init();
75     int l = strlen(s);
76     for (int i = 0; i < l; i++)
77         SAM_add(getid(s[i]),i+1);
78 }
79 char s[MAXN+1];
80 int ans;
81 int head[MAXN*2];
82 struct Edge
83 {
84     int to,next;
85 } edge[MAXN*2];
86 int M;
87 int n;
88 void add_edge(int u,int v)
89 {
90     edge[M].to=v;
91     edge[M].next=head[u];
92     head[u]=M++;
93 }
94 void dfs(int u)
95 {
96     for (int i=head[u]; i!=-1; i=edge[i].next)
97     {
98         int v=edge[i].to;
99         dfs(v);
100     }

```

```

101     SAM_node[u].mat[j]=max(SAM_node[v].mat[j],SAM_node[u].mat[j]);
102 }
103 int tmp=SAM_node[u].len;
104 for (int i=0; i<n; i++)
105     tmp=min(tmp,SAM_node[u].mat[i]);
106 ans=max(ans,tmp);
107 }
108 int main()
109 {
110     //freopen("in.txt","r",stdin);
111     //freopen("out.txt","w",stdout);
112     n=0;
113     gets(s);
114     SAM_build(s);
115     while (gets(s))
116     {
117         int l=strlen(s),len=0;
118         SAM_Node *p=SAM_root;
119         for (int i=0; i<l; i++)
120         {
121             if (p->next[getid(s[i])])
122             {
123                 p=p->next[getid(s[i])];
124                 p->mat[n]=max(p->mat[n],++len);
125             }
126             else
127             {
128                 while (p && !p->next[getid(s[i])])
129                     p=p->fa;
130                 if (!p)
131                 {
132                     p=SAM_root;
133                     len=0;
134                 }
135                 else
136                 {
137                     len=p->len+1;
138                     p=p->next[getid(s[i])];
139                 }
140                 p->mat[n]=max(p->mat[n],len);
141             }
142             //printf("%d %d %d\n",i,len,p->id);
143         }
144         n++;
145     }
146     memset(head,-1,4*SAM_size);
147     for (int i=1; i<SAM_size; i++)
148         add_edge(SAM_node[i].fa->id,i);
149     dfs(0);
150     printf("%d\n",ans);
151     return 0;

```


152 | }

2.4 KMP

求 $A[0..i]$ 的一个后缀最多能匹配 B 的前缀多长。先对 B 进行自匹配然后与 A 匹配。 $KMP[i]$ 就是对应答案， $p[i+1]$ 是 $B[0..i]$ 的一个后缀最多能匹配 B 的前缀多长。

```

1 //自匹配过程
2 int j;
3 p[0] = j = -1;
4 for ( int i = 1; i < lb; i++)
5 {
6     while (j >= 0 && b[j + 1] != b[i]) j = p[j];
7     if (b[j + 1] == b[i]) j ++;
8     p[i] = j;
9 }
10 //下面是匹配过程
11 j = -1;
12 for ( int i = 0; i < la; i++)
13 {
14     while (j >= 0 && b[j + 1] != a[i]) j = p[j];
15     if (b[j + 1] == a[i]) j ++;
16     KMP[i] = j + 1;
17 }
```

2.5 e-KMP

求 $A[i..len-1]$ 和 B 的最长公共前缀有多长。先对 B 进行自匹配然后与 A 匹配。 $eKMP[i]$ 就是对应答案。 $p[i]$ 是 $B[i..len-1]$ 和 B 的最长公共前缀有多长。

```

1 //自匹配过程
2 int j = 0;
3 while (j < lb && b[j] == b[j + 1])
4     j++;
5 p[0] = lb, p[1] = j;
6 int k = 1;
7 for (int i = 2; i < lb; i++)
8 {
9     int Len = k + p[k] - 1, L = p[i - k];
10    if (L < Len - i + 1)
11        p[i] = L;
12    else
13    {
14        j = max(0, Len - i + 1);
15        while (i + j < lb && b[i + j] == b[j])
16            j++;
17        p[i] = j, k = i;
18    }
19 }
20 //下面是匹配过程
21 j = 0;
22 while (j < la && j < lb && a[j] == b[j])
23     j++;
```

```

24 eKMP[0] = j;
25 k = 0;
26 for (int i = 1; i < la; i++)
27 {
28     int Len = k + eKMP[k] - 1, L = p[i - k];
29     if (L < Len - i + 1)
30         eKMP[i] = L;
31     else
32     {
33         j = max(0, Len - i + 1);
34         while (i + j < la && j < lb && a[i + j] == b[j])
35             j++;
36         eKMP[i] = j, k = i;
37     }
38 }

```

2.6 Manacher

```

1  const int maxn = 110000;
2
3  char Ma[maxn*2];
4  int Mp[maxn*2];
5  void Manacher(char s[],int len)
6  {
7      int l = 0;
8      Ma[l++] = '.';
9      Ma[l++] = ',';
10     for (int i = 0; i < len; i++)
11     {
12         Ma[l++] = s[i];
13         Ma[l++] = ',';
14     }
15     Ma[l] = 0;
16     int pnow = 0, pid = 0;
17     for (int i = 1; i < l; i++)
18     {
19         if (pnow > i)
20             Mp[i] = min(Mp[2*pid-i], pnow-i);
21         else
22             Mp[i] = 1;
23         for (; Ma[i-Mp[i]] == Ma[i+Mp[i]]; Mp[i]++);
24         if (i+Mp[i] > pnow)
25         {
26             pnow = i+Mp[i];
27             pid = i;
28         }
29     }
30 }
31 /*
32 abaaba
33 . , a , b , a , a , b , a ,

```

```

34 0 1 2 1 4 1 2 7 2 1 4 1 2 1
35 */

```

2.7 不同回文串

往 hash 表中插入新东西的时候就说明找到了一个新回文字串
一共 $O(n)$ 个

```

1  typedef unsigned int uint;
2
3  const int maxn = 110000;
4
5  char Ma[maxn*2];
6  int Mp[maxn*2];
7  void Manacher(char s[],int len)
8  {
9      int l = 0;
10     Ma[l++] = '.';
11     Ma[l++] = ',';
12     for (int i = 0; i < len; i++)
13     {
14         Ma[l++] = s[i];
15         Ma[l++] = ',';
16     }
17     Ma[l] = 0;
18     int pnow = 0,pid = 0;
19     for (int i = 1; i < l; i++)
20     {
21         if (pnow > i)
22             Mp[i] = min(Mp[2*pid-i],pnow-i);
23         else
24             Mp[i] = 1;
25         for (; Ma[i-Mp[i]] == Ma[i+Mp[i]]; Mp[i]++);
26         if (i+Mp[i] > pnow)
27         {
28             pnow = i+Mp[i];
29             pid = i;
30         }
31     }
32 }
33
34 char s[maxn*2];
35 int len;
36 int p[maxn*2];
37 const int muts = 129;
38 uint sum[maxn];
39 uint mutpower[maxn];
40
41 struct hash_map
42 {
43     const static int mod = 300007;
44     int head[mod];
45     struct hash_tables

```

```

46 {
47     uint key1;
48     int key2;
49     int next;
50 } ele[maxn*10];
51 int N;
52 void init()
53 {
54     memset(head,-1,sizeof(head));
55     N = 0;
56 }
57 int totlen[mod];
58 void clear()
59 {
60     for (int i = 0; i < N; i++)
61         head[ele[i].key1%mod] = -1;
62     N = 0;
63 }
64 int find(uint x,int len)
65 {
66     int hashcode = x%mod;
67     for (int i = head[hashcode]; i != -1; i = ele[i].next)
68         if (ele[i].key1 == x && ele[i].key2 == len)
69             return i;
70     return -1;
71 }
72 void insert(uint x,int len)
73 {
74     int tmp = x%mod;
75     ele[N].key1 = x;
76     ele[N].key2 = len;
77     ele[N].next = head[tmp];
78     head[tmp] = N++;
79 }
80 };
81
82 hash_map hash;
83
84 uint gethashcode(int l,int r)
85 {
86     uint ret;
87     ret = sum[r];
88     if (l)
89         ret -= sum[l-1]*mutpower[r-l+1];
90     return ret;
91 }
92
93 int calc(char s[])
94 {
95     len = strlen(s);
96     Manacher(s,len);

```

```

97
98 sum[0] = s[0];
99 for (int i = 1; i < len; i++)
100     sum[i] = sum[i-1]*muts+s[i];
101
102 int res = 0;
103 uint tmp;
104 int nt = 0;
105 hash.clear();
106 //odd
107 for (int i = 0; i < len; i++)
108     if (Mp[i*2+2]%2 == 0)
109     {
110         int pl = Mp[i*2+2]/2;
111         if (i+pl < nt || pl == 0) continue;
112         for (int j = i-pl+1; j <= i; j++)
113         {
114             tmp = gethashcode(j,i);
115             if (hash.find(tmp,i-j+1) != -1) break;
116             hash.insert(tmp,i-j+1);
117         }
118         nt = i+pl;
119     }
120 res += hash.N;
121
122 nt = 0;
123 hash.clear();
124 //even
125 for (int i = 0; i < len; i++)
126     if (Mp[i*2+3] > 1)
127     {
128         int pl = Mp[i*2+3]/2;
129         if (i+pl < nt || pl == 0) continue;
130         for (int j = i-pl+1; j <= i; j++)
131         {
132             tmp = gethashcode(j,i);
133             if (hash.find(tmp,i-j+1) != -1) break;
134             hash.insert(tmp,i-j+1);
135         }
136         nt = i+pl;
137     }
138 res += hash.N;
139 return res;
140 }
141
142 int main()
143 {
144     mutpower[0] = 1;
145     for (int i = 1; i < maxn; i++)
146         mutpower[i] = mutpower[i-1]*muts;
147     hash.init();

```

```

148
149     int totcas;
150     scanf("%d",&totcas);
151     for (int cas = 1; cas <= totcas; cas++)
152     {
153         scanf("%s",s);
154
155         printf("Case_#%d:_%d\n",cas,calc(s));
156     }
157     return 0;
158 }

```

2.8 * 字符串最小表示法

```

1  int Gao(char a[],int len)
2  {
3      int i = 0,j = 1,k = 0;
4      while (i < len && j < len && k < len)
5      {
6          int cmp = a[(j+k)%len]-a[(i+k)%len];
7          if (cmp == 0)
8              k++;
9          else
10         {
11             if (cmp > 0)
12                 j += k+1;
13             else
14                 i += k+1;
15             if (i == j) j++;
16             k = 0;
17         }
18     }
19     return min(i,j);
20 }

```

2.9 带 * 通配符的匹配

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  using namespace std;
6
7  char a[110],b[110],sp[110][110],tot,place[110];
8  int n,la,lb,ll;
9
10 bool check(int id,int pos)
11 {
12     for (int i = 0;sp[id][i] != 0;i++)
13         if (b[pos+i] != sp[id][i])
14             return false;

```

```

15     return true;
16 }
17
18 bool check()
19 {
20     lb = strlen(b);
21     int pre = 0;
22     for (int i = 0; i < tot; i++)
23     {
24         bool find = false;
25         for (int j = pre; j < lb; j++)
26             if (check(i, j) == true)
27             {
28                 place[i] = j;
29                 pre = place[i] + 1;
30                 find = true;
31                 break;
32             }
33         if (find == false) return false;
34     }
35     if (a[0] != '*')
36         if (place[0] != 0)
37             return false;
38     if (a[la-1] != '*')
39         if (check(tot-1, lb-1) == false)
40             return false;
41     return true;
42 }
43
44 int main()
45 {
46     while (scanf("%s", a) != EOF)
47     {
48         tot = 0;
49         for (int i = 0; a[i] != 0; i++)
50             if (a[i] != '*')
51             {
52                 int j;
53                 for (j = i; a[j] != 0 && a[j] != '*'; j++)
54                     sp[tot][j-i] = a[j];
55                 sp[tot++][j-i] = 0;
56                 i = j;
57             }
58         la = strlen(a);
59         ll = strlen(sp[tot-1]);
60         scanf("%d", &n);
61         for (int i = 0; i < n; i++)
62         {
63             scanf("%s", b);
64             if (check() == true)
65                 puts(b);

```

```
66     }
67 }
68     return 0;
69 }
70 /*
71 Sample Input 1
72 *.*
73 4
74 main.c
75 a.out
76 readme
77 yacc
78
79 Sample Input 2
80 *a*a*a
81 4
82 aaa
83 aaaaa
84 aaaaax
85 abababa
86
87 Sample Output 1
88 main.c
89 a.out
90
91 Sample Output 2
92 aaa
93 aaaaa
94 abababa
95 */
```


3 数学

3.1 扩展 GCD

求 $ax+by=\gcd(a,b)$ 的一组解

```

1 long long ex_gcd(long long a,long long b,long long &x,long long &y)
2 {
3     if (b)
4     {
5         long long ret = ex_gcd(b,a%b,x,y),tmp = x;
6         x = y;
7         y = tmp-(a/b)*y;
8         return ret;
9     }
10    else
11    {
12        x = 1;
13        y = 0;
14        return a;
15    }
16 }
```

3.2 模线性方程组

```

1 //有更新
2 int m[10],a[10]; //模数m 余数a
3 bool solve(int &m0,int &a0,int m,int a) //模线性方程组
4 {
5     int y,x;
6     int g=ex_gcd(m0,m,x,y);
7     if (abs(a-a0)%g) return 0;
8     x*=(a-a0)/g;
9     x%=m/g;
10    a0=(x*m0+a0);
11    m0*=m/g;
12    a0%=m0;
13    if (a0<0) a0+=m0;
14    return 1;
15 }
16 int MLES()
17 {
18     bool flag=1;
19     int m0=1,a0=0;
20     for (int i=0; i<n; i++)
21         if (!solve(m0,a0,m[i],a[i]))
22         {
23             flag=0;
24             break;
25         }
26     if (flag)
```

```

27     return a0;
28     else
29         return -1;
30 }

```

3.3 矩阵

乘法的时候将 B 数组转置一下然后 $C[i][j] = \sum A[i][k] \times B[j][k]$ 会有奇效。

```

1  struct Matrix
2  {
3      int a[52][52];
4      void clear()
5      {
6          memset(a,0,sizeof(a));
7      }
8      int det(int n)//求行列式的值模上一个数，需要预处理逆元
9      {
10         for (int i = 0;i < n;i++)
11             for (int j = 0;j < n;j++)
12                 a[i][j] = (a[i][j]%mod+mod)%mod;
13         int res = 1;
14         for (int i = 0;i < n;i++)
15         {
16             for (int j = i;j < n;j++)
17                 if (a[j][i] != 0)
18                 {
19                     for (int k = i;k < n;k++)
20                         swap(a[i][k],a[j][k]);
21                     if (i != j)
22                         res = (res+mod)%mod;
23                     break;
24                 }
25             if (a[i][i] == 0)
26             {
27                 res = -1;//不存在
28                 break;
29             }
30             for (int j = i+1;j < n;j++)
31             {
32                 int mut = (a[j][i]*inv[a[i][i]])%mod;
33                 for (int k = i;k < n;k++)
34                     a[j][k] = (a[j][k]-(a[i][k]*mut)%mod+mod)%mod;
35             }
36             res = (res*a[i][i])%mod;
37         }
38         return res;
39     }
40     Matrix operator * (const Matrix &b)const
41     {
42         Matrix res;
43         for (int i = 0; i < 52; i++)
44             for (int j = 0; j < 52; j++)

```

```

45     {
46         res.a[i][j] = 0;
47         for (int k = 0; k < 52; k++)
48             res.a[i][j] += a[i][k] * b.a[k][j];
49     }
50     return res;
51 }
52 Matrix operator ^ (int y) const
53 {
54     Matrix res, x;
55     for (int i = 0; i < 52; i++)
56     {
57         for (int j = 0; j < 52; j++)
58             res.a[i][j] = 0, x.a[i][j] = a[i][j];
59         res.a[i][i] = 1;
60     }
61     for (; y; y >>= 1, x = x * x)
62         if (y & 1)
63             res = res * x;
64     return res;
65 }
66 };

```

3.4 康拓展开

```

1  const int PermSize = 12;
2  int factory[PermSize] =
3  {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, 39916800};
4  int Cantor(int a[])
5  {
6      int i, j, counted;
7      int result = 0;
8      for (i = 0; i < PermSize; ++i)
9      {
10         counted = 0;
11         for (j = i + 1; j < PermSize; ++j)
12             if (a[i] > a[j])
13                 ++counted;
14         result = result + counted * factory[PermSize - i - 1];
15     }
16     return result;
17 }
18
19 bool h[13];
20
21 void UnCantor(int x, int res[])
22 {
23     int i, j, l, t;
24     for (i = 1; i <= 12; i++)
25         h[i] = false;
26     for (i = 1; i <= 12; i++)

```

```

27 {
28     t = x / factory[12 - i];
29     x -= t * factory[12 - i];
30     for (j = 1, l = 0; l <= t; j++)
31         if (!h[j]) l++;
32     j--;
33     h[j] = true;
34     res[i - 1] = j;
35 }
36 }

```

3.5 FFT

```

1  const double PI= acos(-1.0);
2  struct vir
3  {
4      double re,im; //实部和虚部
5      vir(double a=0,double b=0)
6      {
7          re=a;
8          im=b;
9      }
10     vir operator +(const vir &b)
11     {return vir(re+b.re,im+b.im);}
12     vir operator -(const vir &b)
13     {return vir(re-b.re, im-b.im);}
14     vir operator *(const vir &b)
15     {return vir(re*b.re-im*b.im , re*b.im+im*b.re);}
16 };
17 vir x1[200005],x2[200005];
18 void change(vir *x,int len,int loglen)
19 {
20     int i,j,k,t;
21     for(i=0;i<len;i++)
22     {
23         t=i;
24         for(j=k=0; j<loglen; j++,t>>=1)
25             k= (k<<1)|(t&1);
26         if(k<i)
27         {
28             // printf("%d %d\n",k,i);
29             vir wt=x[k];
30             x[k]=x[i];
31             x[i]=wt;
32         }
33     }
34 }
35 void fft(vir *x,int len,int loglen)
36 {
37     int i,j,t,s,e;
38     change(x,len,loglen);
39     t=1;

```

```

40  for(i=0;i<loglen;i++,t<=1)
41  {
42      s=0;
43      e=s+t;
44      while(s<len)
45      {
46          vir a,b,wo(cos(PI/t),sin(PI/t)),wn(1,0);
47          for(j=s;j<s+t;j++)
48          {
49              a=x[j];
50              b=x[j+t]*wn;
51              x[j]=a+b;
52              x[j+t]=a-b;
53              wn=wn*wo;
54          }
55          s=e+t;
56          e=s+t;
57      }
58  }
59 }
60 void dit_fft(vir *x,int len,int loglen)
61 {
62     int i,j,s,e,t=1<<loglen;
63     for(i=0;i<loglen;i++)
64     {
65         t>>=1;
66         s=0;
67         e=s+t;
68         while(s<len)
69         {
70             vir a,b,wn(1,0),wo(cos(PI/t),-sin(PI/t));
71             for(j=s;j<s+t;j++)
72             {
73                 a=x[j]+x[j+t];
74                 b=(x[j]-x[j+t])*wn;
75                 x[j]=a;
76                 x[j+t]=b;
77                 wn=wn*wo;
78             }
79             s=e+t;
80             e=s+t;
81         }
82     }
83     change(x,len,loglen);
84     for(i=0;i<len;i++)
85         x[i].re/=len;
86 }
87 int main()
88 {
89     char a[100005],b[100005];
90     int i,len1,len2,len,loglen;

```

```

91  int t,over;
92  while(scanf("%s%s",a,b)!=EOF)
93  {
94      len1=strlen(a)<<1;
95      len2=strlen(b)<<1;
96      len=1;loglen=0;
97      while(len<len1)
98      {
99          len<<=1; loglen++;
100     }
101     while(len<len2)
102     {
103         len<<=1; loglen++;
104     }
105     for(i=0;a[i];i++)
106     {
107         x1[i].re=a[i]-'0';
108         x1[i].im=0;
109     }
110     for(;i<len;i++)
111         x1[i].re=x1[i].im=0;
112     for(i=0;b[i];i++)
113     {
114         x2[i].re=b[i]-'0';
115         x2[i].im=0;
116     }
117     for(;i<len;i++)
118         x2[i].re=x2[i].im=0;
119     fft(x1,len,loglen);
120     fft(x2,len,loglen);
121     for(i=0;i<len;i++)
122         x1[i] = x1[i]*x2[i];
123     dit_fft(x1,len,loglen);
124     for(i=(len1+len2)/2-2,over=len=0;i>=0;i--)
125     {
126         t=(int)(x1[i].re+over+0.5);
127         a[len++]= t%10;
128         over = t/10;
129     }
130     while(over)
131     {
132         a[len++]=over%10;
133         over/=10;
134     }
135     for(len--;len>=0&&!a[len];len--);
136     if(len<0)
137         putchar('0');
138     else
139         for(;len>=0;len--)
140             putchar(a[len]+'0');
141     putchar('\n');

```

```

142     }
143     return 0;
144 }

```

3.6 爬山法计算器

注意灵活运用。

双目运算符在 `calc()` 中，左结合单目运算符在 `P()` 中，右结合单目运算符在 `calc_exp` 中。（但是还没遇到过。。）

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  #include <string>
6  using namespace std;
7
8  char s[100000];
9  int n, cur;
10 const string OP = "+-*/";
11
12 char next_char()
13 {
14     if (cur >= n) return EOF;
15     return s[cur];
16 }
17
18 int get_priority(char ch)
19 {
20     if (ch == '*') return 2;
21     return 1;
22 }
23
24 int P();
25
26 int calc(int a, char op, int b)
27 {
28     if (op == '+')
29         return a+b;
30     if (op == '-')
31         return a-b;
32     if (op == '*')
33         return a*b;
34 }
35
36 int calc_exp(int p)
37 {
38     int a = P();
39     while ((OP.find(next_char()) != OP.npos) &&
40         (get_priority(next_char()) >= p))
41     {
42         char op = next_char();

```

```
43     cur++;
44     a = calc(a,op,calc_exp(get_priority(op)+1));
45 }
46 return a;
47 }
48
49 int totvar,m,var[26],varid[26];
50
51 int P()
52 {
53     if (next_char() == '-')
54     {
55         cur++;
56         return -P();
57     }
58     else if (next_char() == '+')
59     {
60         cur++;
61         return P();
62     }
63     else if (next_char() == '(')
64     {
65         cur++;
66         int res = calc_exp(0);
67         cur++;
68         return res;
69     }
70     else
71     {
72         cur++;
73         return var[varid[s[cur-1]-'a']];
74     }
75 }
76
77 int id[26],minid;
78
79 int main()
80 {
81     while (true)
82     {
83         scanf("%d%d",&totvar,&var[0]);
84         if (totvar == 0 && var[0] == 0) break;
85         for (int i = 1;i < totvar;i++)
86             scanf("%d",&var[i]);
87         scanf("%d",&m);
88         scanf("%s",s);
89         for (int i = 0;i < 26;i++)
90             id[i] = -1;
91         minid = 0;
92         n = strlen(s);
93         for (int i = 0;i < n;i++)
```



```

94     if (s[i] >= 'a' && s[i] <= 'z')
95     {
96         if (id[s[i]-'a'] == -1)
97         {
98             id[s[i]-'a'] = minid;
99             minid++;
100        }
101        s[i] = 'a'+id[s[i]-'a'];
102    }
103    for (int i = 0; i < totvar; i++)
104        varid[i] = i;
105    int res = 0;
106    do
107    {
108        cur = 0;
109        int tmp = calc_exp(0);
110        if (tmp == m)
111        {
112            res++;
113            break;
114        }
115    }
116    while (next_permutation(varid, varid+totvar));
117    //puts(s);
118    if (res > 0)
119        puts("YES");
120    else
121        puts("NO");
122 }
123 return 0;
124 }

```

3.7 线性筛

```

1  int N;
2  bool isPrime[10001];
3  int prime[10000];
4  void getPrime(int n)
5  {
6      memset(isPrime, 1, ++n);
7      N=0;
8      isPrime[0]=isPrime[1]=0;
9      for (int i=2; i<n; i++)
10     {
11         if (isPrime[i])
12             prime[N++]=i;
13         for (int j=0; j<N && prime[j]*i<n; j++)
14         {
15             isPrime[i*prime[j]]=0;
16             if (i%prime[j]==0)
17                 break;

```

```

18     }
19 }
20 }

```

3.8 线性规划

```

1  #define MAXM 20 //max num of basic variables
2  #define INF 1E200
3
4  double A[MAXM+5][MAXN+MAXM+5];
5  double b[MAXM+5],c[MAXN+MAXM+5];
6  int N[MAXN+5],B[MAXM+5];
7  double X[MAXN+MAXM+5],V;
8  int n,m,R,C,nCnt,bCnt;
9  int v1[MAXN],v2[MAXN];
10
11 int fcmp(double a,double b)
12 {
13     if(fabs(a-b)<1E-7) return 0;
14     if(a>b) return 1;
15     return -1;
16 }
17
18 void Pivot(int l,int e)
19 {
20     double t=A[l][e],p=c[e];
21     b[l]=b[l]/t;
22     for(int i=1;i<=C;i++)
23         A[l][i]/=t;
24     V=V-c[e]*b[l];
25     for(int i=1;i<=R;i++)
26     {
27         if(i==l || fcmp(A[i][e],0.0)==0)
28             continue;
29         t=A[i][e];
30         b[i]=b[i]-t*b[l];
31         for(int j=1;j<=C;j++)
32             A[i][j]=A[i][j]-t*A[l][j];
33     }
34     for(int i=1;i<=C;i++)
35         c[i]=c[i]-p*A[l][i];
36     for(int i=1;i<=nCnt;i++)
37     {
38         if(N[i]==e)
39         {
40             N[i]=B[l];
41             break;
42         }
43     }
44     B[l]=e;
45 }
46

```

```

47 bool Process(double P[])
48 {
49     while(true)
50     {
51         int e=-1;
52         double mV=-INF;
53         for(int i=1;i<=nCnt;i++)
54             if(fcmp(P[N[i]],mV)==1)
55                 mV=P[N[i]],e=N[i];
56
57         if(fcmp(mV,0.0)<=0) break;
58         int l=-1;
59         mV=INF;
60         for(int i=1;i<=bCnt;i++)
61         {
62             if(fcmp(A[i][e],0.0)==1)
63             {
64                 double t=b[i]/A[i][e];
65                 if(fcmp(mV,t)==1 || (fcmp(mV,t)==0&&(l==-1 || B[l]>B[i])))
66                     mV=t,l=i;
67             }
68         }
69         if(l==-1) return false;
70         Pivot(l,e);
71     }
72     return true;
73 }
74
75 bool initSimplex()
76 {
77     nCnt=bCnt=0;
78     for(int i=1;i<=n;i++)
79         N[++nCnt]=i;
80     for(int i=1;i<=m;i++)
81         B[++bCnt]=i+n,A[i][n+i]=1.0;
82     R=bCnt,C=bCnt+nCnt;
83     double minV=INF;
84     int p=-1;
85     for(int i=1;i<=m;i++)
86         if(fcmp(minV,b[i])==1)
87             minV=b[i],p=i;
88     if(fcmp(minV,0.0)>=0)
89         return true;
90     N[++nCnt]=n+m+1;R++,C++;
91     for(int i=0;i<=C;i++)
92         A[R][i]=0.0;
93     for(int i=1;i<=R;i++)
94         A[i][n+m+1]=-1.0;
95     Pivot(p,n+m+1);
96     if(!Process(A[R])) return false;
97     if(fcmp(b[R],0.0)!=0)

```

```

98     return false;
99     p=-1;
100    for(int i=1;i<=bCnt&& p==-1;i++)
101        if(B[i]==n+m+1) p=i;
102    if(p!=-1)
103    {
104        for(int i=1;i<=nCnt;i++)
105        {
106            if(fcmp(A[p][N[i]],0.0)!=0)
107            {
108                Pivot(p,N[i]);
109                break;
110            }
111        }
112    }
113    bool f=false;
114    for(int i=1;i<=nCnt;i++)
115    {
116        if(N[i]==n+m+1) f=true;
117        if(f&& i+1<=nCnt)
118            N[i]=N[i+1];
119    }
120    nCnt--;
121    R--,C--;
122    return true;
123 }
124
125 // -1: no solution 1: no bound 0: has a solution -V
126 int Simplex()
127 {
128     if(!initSimplex())
129         return -1;
130     if(!Process(c))
131         return 1;
132     for(int i=1;i<=nCnt;i++)
133         X[N[i]]=0.0;
134     for(int i=1;i<=bCnt;i++)
135         X[B[i]]=b[i];
136     return 0;
137 }
138
139 int main()
140 {
141     //n = 1;m=1;
142     //V= 0.0;
143     //c[1] = 1.0;
144     //A[1][1] = 1.0;
145     //b[1] = 5.0;
146     //Simplex();
147     //printf("V = %.3f\n",V);
148

```

```

149 while(scanf("%d",&v1[1]) == 1)
150 {
151     for(int i = 2; i<=6;i++)
152         scanf("%d",&v1[i]);
153     n = 4; m = 6;
154     for(int i = 0 ; i<=m+1;i++)
155         for(int j=0;j<=n+m+2;j++)
156             A[i][j] = c[j] = 0;
157     memset(b,0,sizeof(b));
158     V = 0.0;
159     /*
160     n 为未知数个数
161     m 为约束个数
162     目标: siama(c[i]*xi)
163     约束: sigma(A[i][j]*xj) <=b[i]; j = 1 ... n
164     解存在 X 里面
165     */
166     b[1] = v1[1] ; A[1][1] = 1;A[1][4] = 1;
167     b[2] = v1[2] ; A[2][1] = 1;A[2][3] = 1;
168     b[3] = v1[3] ; A[3][3] = 1;A[3][4] = 1;
169     b[4] = v1[4] ; A[4][2] = 1;A[4][3] = 1;
170     b[5] = v1[5] ; A[5][2] = 1;A[5][4] = 1;
171     b[6] = v1[6] ; A[6][1] = 1;A[6][2] = 1;
172     c[1] = 1;c[2] = 1;c[3] = 1;c[4] = 1;
173     Simplex();
174     //printf("V = %.3f\n",V);
175     printf("%.3f_%.3f_%.3f_%.3f\n",X[1],X[2],X[3],X[4]);
176
177 }
178 return 0;
179 }

```

3.9 分解质因数

3.9.1 米勒拉宾 + 分解因数

```

1  #include<ctime>
2  #include<iostream>
3  #define bint long long
4  using namespace std;
5  const int TIME = 8;//测试次数, 够了8~10
6  int factor[100],fac_top = -1;
7
8  //计算两个数的gcd
9  bint gcd(bint small,bint big)
10 {
11     while(small)
12     {
13         swap(small,big);
14         small%=big;
15     }
16     return abs(big);

```

```

17 }
18
19 //ret = (a*b)%n (n<2^62)
20 bint muti_mod(bint a,bint b,bint n)
21 {
22     bint exp = a%n, res = 0;
23     while(b)
24     {
25         if(b&1)
26         {
27             res += exp;
28             if(res>n) res -= n;
29         }
30         exp <<= 1;
31         if (exp>n) exp -= n;
32         b>>=1;
33     }
34     return res;
35 }
36
37 // ret = (a^b)%n
38 bint mod_exp(bint a,bint p,bint m)
39 {
40     bint exp=a%m, res=1; //
41     while(p>1)
42     {
43         if(p&1)
44             res=muti_mod(res,exp,m);
45         exp = muti_mod(exp,exp,m);
46         p>>=1;
47     }
48     return muti_mod(res,exp,m);
49 }
50
51 //miller—法测试素数rabin, time 测试次数
52 bool miller_rabin(bint n, int times)
53 {
54     if(n==2)return 1;
55     if(n<2||!(n&1))return 0;
56     bint a, u=n-1, x, y;
57     int t=0;
58     while(u%2==0)
59     {
60         t++;
61         u/=2;
62     }
63     srand(time(0));
64     for(int i=0; i<times; i++)
65     {
66         a = rand() % (n-1) + 1;
67         x = mod_exp(a, u, n);

```

```

68     for(int j=0; j<t; j++)
69     {
70         y = muti_mod(x, x, n);
71         if ( y == 1 && x != 1 && x != n-1 )
72             return false; //must not
73         x = y;
74     }
75     if( y!=1) return false;
76 }
77 return true;
78 }
79
80 bint pollard_rho(bint n,int c)//找出一个因子
81 {
82     bint x,y,d,i = 1,k = 2;
83     srand(time(0));
84     x = rand()%(n-1)+1;
85     y = x;
86     while(true)
87     {
88         i++;
89         x = (muti_mod(x,x,n) + c) % n;
90         d = gcd(y-x, n);
91         if (1 < d && d < n) return d;
92         if( y == x) return n;
93         if(i == k)
94         {
95             y = x;
96             k <<= 1;
97         }
98     }
99 }
100
101 void findFactor(bint n,int k)//二分找出所有质因子，存入factor
102 {
103     if(n==1)return;
104     if(miller_rabin(n, TIME))
105     {
106         factor[++fac_top] = n;
107         return;
108     }
109     bint p = n;
110     while(p >= n)
111         p = pollard_rho(p,k—); //值变化，防止死循环k
112     findFactor(p,k);
113     findFactor(n/p,k);
114 }
115
116 int main()
117 {
118     bint cs,n,min;

```

```

119  cin>>cs;
120  while (cs—)
121  {
122      cin>>n;
123      fac_top = min = -1;
124      if(miller_rabin(n,TIME)) cout<<"Prime"<<endl;
125      else
126      {
127          findFactor(n,107);
128          for(int i=0; i<=fac_top; i++)
129          {
130              if(min<0||factor[i]<min)
131                  min = factor[i];
132          }
133          cout<<min<<endl;
134      }
135  }
136  return 0;
137 }

```

3.9.2 暴力版本

```

1  int N;
2  int num[30],fac[30];
3  void getFactor(int x)
4  {
5      N=0;
6      memset(num,0,sizeof(num));
7      for (int i=0; prime[i]*prime[i]<=x && i<L; i++)
8      {
9          if (x%prime[i]==0)
10         {
11             while (x%prime[i]==0)
12             {
13                 x/=prime[i];
14                 num[N]++;
15             }
16             fac[N++]=prime[i];
17         }
18     }
19     if (x>1)
20     {
21         num[N]=1;
22         fac[N++]=x;
23     }
24 }

```

3.10 baby step giant step

3.10.1 BSGS

```

1  #define MOD 76543
2  int hs[MOD], head[MOD], next[MOD], id[MOD], top;

```



```

3 void insert(int x, int y)
4 {
5     int k = x%MOD;
6     hs[top] = x, id[top] = y, next[top] = head[k], head[k] = top++;
7 }
8 int find(int x)
9 {
10    int k = x%MOD;
11    for (int i = head[k]; i; i = next[i]) if (hs[i] == x)
12        return id[i];
13    return -1;
14 }
15 int BSGS(int a, int b, int n)
16 {
17     memset(head, 0, sizeof(head));
18     top = 1;
19     if (b==1) return 0;
20     int m = sqrt(n+.0), j;
21     long long x = 1, p = 1;
22     for (int i = 0; i < m; ++i, p = p*a%n) insert(p*b%n, i);
23     for (long long i = m; ; i += m)
24     {
25         if ((j = find(x=x*p%n)) != -1) return i-j;
26         if (i > n) break;
27     }
28     return -1;
29 }

```

3.10.2 何老师的版

```

1 //离散对数
2 #include <cstdio>
3 #include <cstring>
4 #include <cmath>
5 #include <algorithm>
6 using namespace std;
7 typedef long long LL;
8 struct Hash
9 {
10     static const int MOD = 100007;
11     static const int MaxN = 100005;
12     struct Node
13     {
14         LL k, v; //A^k = v
15         Node *nxt;
16     } buf[MaxN], *g[MaxN], *pt;
17     void init()
18     {
19         memset(g, 0, sizeof(g));
20         pt = buf;
21     }
22     LL find(LL v)

```

```

23     {
24         for (Node *now = g[v%MOD]; now; now = now->nxt)
25             if (now->v == v)
26                 return now->k;
27         return -1;
28     }
29 void Ins(LL k, LL v)
30 {
31     if ( find (v) != -1)return;
32     pt->k = k;
33     pt->v = v;
34     pt->nxt = g[v % MOD];
35     g[v % MOD] = pt++;
36 }
37 }hash;
38 LL gcd(LL x, LL y)
39 {
40     return y==0?x:gcd(y,x%y);
41 }
42 LL e_gcd(LL a, LL b, LL &x, LL &y)
43 {
44     if (b==0)
45     {
46         x = 1;
47         y = 0;
48         return a;
49     }
50     LL ret = e_gcd(b, a%b, y, x) ;
51     y = y - a/b*x;
52     return ret ;
53 }
54 LL Baby(LL A, LL B, LL C)//A^x = B (mod C)
55 {
56     B %= C;
57     A %= C;
58     LL x = 1%C, y;
59     for (int i = 0; i <= 64; i++)
60     {
61         if (x==B)return i;
62         x = x*A % C;
63     }
64
65     LL D = 1%C, g;
66     int cnt = 0;
67     while((g = gcd(A,C)) != 1)
68     {
69         if (B%g) return -1;
70         cnt++;
71         C /= g;
72         B /= g;
73         D = A/g * D % C;

```

```

74     }
75     hash. init () ;
76     int m = (int)sqrt(C);
77     LL Am = 1%C;
78     hash.Ins(0,Am);
79     for (int i = 1; i <= m; i++)
80     {
81         Am = Am*A % C;
82         hash. Ins ( i ,Am);
83     }
84     for (int i = 0; i <= m; i++)
85     {
86         //D*x = B (mod C), D*x + C*y = B
87         g = e_gcd(D,C,x,y);
88         x = (x*B/g%C+C)%C;
89         LL k = hash.find(x) ;
90         if (k != -1) return i*m+k+cnt;
91         D = D*Am % C;
92     }
93     return -1;
94 }
95 int main()
96 {
97     int A,B,C;
98     while(scanf("%d%d%d",&A,&C,&B) == 3 && (A+B+C))
99     {
100         if (B>=C)
101         {
102             puts("Orz,I' cant_find_D!");
103             continue;
104         }
105         LL ret = Baby(A,B,C);
106         if ( ret == -1)puts("Orz,I' cant_find_D!");
107         else printf ("%I64d\n",ret);
108     }
109     return 0;
110 }

```

3.11 原根

```

1 int getPriRoot(int p)
2 {
3     if (p==2) return 1;
4     int phi = p - 1;
5     getFactor(phi);
6     for (int g = 2; g < p; ++g)
7     {
8         bool flag=1;
9         for (int i = 0; flag && i < N; ++i)
10             if (power(g, phi/fac[i], p) == 1)
11                 flag=0;
12         if (flag)

```

```

13     return g;
14 }
15 }

```

3.12 逆元

```

1 void getInv2(int x)
2 {
3     inv[1]=1;
4     for (int i=2; i<=x; i++)
5         inv[i]=(mod-(mod/i)*inv[mod%i]%mod)%mod;
6 }
7 int getInv(int x)//为素数mod
8 {
9     return power(x,mod-2);
10 }

```

3.13 卢卡斯

卢卡斯, $num[i]$ 阶乘也

```

1 int comLucus(int n,int m,int p)
2 {
3     int ans=1;
4     for (; n && m && ans; n/=p,m/=p)
5     {
6         if (n%p>=m%p)
7             ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p
8                 *getInv(num[n%p-m%p])%p;
9         else
10            ans=0;
11     }
12     return ans;
13 }

```

3.14 欧拉函数

3.14.1 分解质因数

```

1 int getEuler(int x)
2 {
3     getFactor(x);
4     int ret=x;
5     for (int i=0; i<N; i++)
6         ret = ret/fac[i]*(fac[i]-1);
7     return ret;
8 }

```

3.14.2 一次预处理

```

1 void getEuler2()
2 {
3     memset(euler,0,sizeof(euler));
4     euler[1] = 1;

```

```

5   for (int i = 2; i <= 3000000; i++)
6   {
7       if (!euler[i])
8       {
9           for (int j = i; j <= 3000000; j += i)
10          {
11              if (!euler[j])
12                  euler[j] = j;
13              euler[j] = euler[j]/i*(i-1);
14          }
15      }
16  }
17 }

```

3.15 费马降阶法

分解素数 p 为 $x^2 + y^2$ 的费马降阶法，失败返回 -1 ，主程序调用 `calcu(p,x,y)`

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  int p,expp,A,B,aa,ans,tt;
5  long long M;
6  long long exp(int a,int b,long long mod)
7  {
8      long long ans=1,num=a;
9      while (b!=0)
10     {
11         if (b&1)
12         {
13             ans=((ans%mod)*(num%mod))%mod;
14         }
15         num=((num%mod)*(num%mod))%mod;
16         b>>=1;
17     }
18     return ans;
19 }
20 int calcu(int p,int &x,int &y)
21 {
22     if (p%4!=1) return -1;
23     else
24     {
25         expp=(p-1)/4;
26         A,B;
27         while (1)
28         {
29             aa=rand()%p;
30             if (aa==0) continue;
31             A=exp(aa,expp,p);
32             ans=((long long)A%p)*((long long)A%p)%p;
33             if (ans==p-1) break;
34         }
35         B=1;

```

```

36     M=((long long)A*(long long)A+(long long)B*(long long)B)/p;
37     if (M!=1) B=p;
38     while (M!=1)
39     {
40         if (B>A)
41             {tt=A; A=B; B=tt;}
42         tt=A;
43         A=B;
44         B=tt%B;
45         M=((long long)A*(long long)A
46            +(long long)B*(long long)B)/p;
47     }
48     if (B<=A)
49     {
50         x=B;
51         y=A;
52     }
53     else
54     {
55         x=A;
56         y=B;
57     }
58 }
59 }
60 int main()
61 {
62     while (scanf("%d",&p)!=EOF)
63     {
64         int x,y;
65         if (calcu(p,x,y)!=-1)
66         }
67         return 0;
68 }

```

3.16 自适应 simp

过了哈尔滨积分题，精度要求不高的时候可以考虑使用。
暂时我只能用这个做做类似于凸函数或者凹函数的函数。

```

1 double Simp(double l,double r)
2 {
3     double h = (r-l)/2.0;
4     return h*(calc(l)+4*calc((l+r)/2.0)+calc(r))/3.0;
5 }
6
7 double rSimp(double l,double r)
8 {
9     double mid = (l+r)/2.0;
10    if (abs((Simp(l,r)-Simp(l,mid)-Simp(mid,r)))/15 < eps)
11        return Simp(l,r);
12    else
13        return rSimp(l,mid)+rSimp(mid,r);

```

14 | }

3.17 组合数求模

模是质数

```

1  #include<cstdio>
2  #include<cstring>
3  #include<iostream>
4  using namespace std;
5  int mod;
6  long long num[100000];
7  int ni[100],mi[100];
8  int len;
9  void init(int p)
10 {
11     mod=p;
12     num[0]=1;
13     for (int i=1; i<p; i++)
14         num[i]=i*num[i-1]%p;
15 }
16 void get(int n,int ni[],int p)
17 {
18     for (int i = 0; i < 100; i++)
19         ni[i] = 0;
20     int tlen = 0;
21     while (n != 0)
22     {
23         ni[tlen++] = n%p;
24         n /= p;
25     }
26     len = tlen;
27 }
28 long long power(long long x,long long y)
29 {
30     long long ret=1;
31     for (long long a=x%mod; y; y>>=1,a=a*a%mod)
32         if (y&1)
33             ret=ret*a%mod;
34     return ret;
35 }
36 long long getInv(long long x)//mod 为素数
37 {
38     return power(x,mod-2);
39 }
40 long long calc(int n,int m,int p)//C(n,m)%p
41 {
42     init(p);
43     long long ans=1;
44     for (; n && m && ans; n/=p,m/=p)
45     {
46         if (n%p>=m%p)

```

```

47     ans = ans*num[n%p]%p*getInv(num[m%p]%p)%p
48         *getInv(num[n%p-m%p])%p;
49     else
50         ans=0;
51 }
52 return ans;
53 }
54 int main()
55 {
56     int t;
57     scanf("%d",&t);
58     while (t--)
59     {
60         int n,m,p;
61         scanf("%d%d%d",&n,&m,&p);
62         printf("%I64d\n",calc(n+m,m,p));
63     }
64     return 0;
65 }

```

3.18 高斯消元

```

1  const double eps = 1e-8;
2
3  void Guess(int n)
4  {
5      for (int i = 0; i < n; i++)
6      {
7          for (int j = i; j < n; j++)
8              if (fabs(a[j][i]) > eps)
9              {
10                 for (int k = i; k <= n; k++)
11                     swap(a[i][k],a[j][k]);
12                 break;
13             }
14
15             if (fabs(a[i][i]) < eps) continue;
16
17             for (int j = 0; j < n; j++)
18                 if (i != j && fabs(a[j][i]) > eps)
19                 {
20                     double det = a[j][i]/a[i][i];
21                     for (int k = i; k <= n; k++)
22                         a[j][k] -= a[i][k]*det;
23                 }
24         }
25
26         for (int i = 0; i < n; i++)
27         {
28             if (fabs(a[i][i]) < eps)
29             {

```



```

30     if (fabs(a[i][n]) > eps)
31     {
32         //无解
33         puts("Fuck");
34     }
35     //否则  $x_i$  可以是任意解
36 }
37 else
38 {
39     a[i][n] /= a[i][i];
40     if (fabs(a[i][n]) < eps)
41         a[i][n] = 0;
42 }
43 }
44
45 }

```

3.19 整数拆分

```

1  #include <cstdio>
2  #include <cmath>
3  #include <cstring>
4  #include <map>
5  #include <algorithm>
6  using namespace std;
7  bool check(int x)
8  {
9      for (int i=2; i*i<=x; i++)
10         if (x%i==0)
11             return 0;
12     return 1;
13 }
14 int p[100000];
15 inline int calc(int x)
16 {
17     return x*(x*3-1)/2;
18 }
19 int main()
20 {
21     p[0]=1;
22     for (int i=1; i<100000; i++)
23     {
24         for (int j=1, k=1; calc(j)<=i; j++, k*=-1)
25         {
26             p[i]+=k*p[i-calc(j)];
27             if (p[i]<0)
28                 p[i]+=1000000;
29             if (p[i]>=1000000)
30                 p[i]-=1000000;
31             if (calc(-j)<=i)
32                 p[i]+=k*p[i-calc(-j)];
33             if (p[i]<0)

```

```

34     p[i]+=1000000;
35     if (p[i]>=1000000)
36         p[i]-=1000000;
37 }
38 if (!p[i])
39     printf("%d\n",i);
40 }
41 return 0;
42 }

```

3.20 佩尔方程

写的不好稍微收一下

```

1 import java.math.BigInteger;
2 import java.util.*;
3 public class Main
4 {
5     public static class Fraction
6     {
7         public BigInteger num,den;
8         public Fraction()
9         {
10             num=BigInteger.ZERO;
11             den=BigInteger.ONE;
12         }
13         public Fraction(int _num,int _den)
14         {
15             num=BigInteger.valueOf(_num);
16             den=BigInteger.valueOf(_den);
17         }
18         public Fraction(BigInteger _num,BigInteger _den)
19         {
20             num=_num;
21             den=_den;
22         }
23         public Fraction gen()
24         {
25             BigInteger g=num.gcd(den);
26             return new Fraction(num.divide(g),den.divide(g));
27         }
28         public Fraction add(Fraction x)
29         {
30             return new Fraction(x.num.multiply(den).add(num.multiply(x.
31                 den)),x.den.multiply(den)).gen();
32         }
33         public Fraction reciprocal()
34         {
35             return new Fraction(den,num);
36         }
37         public void out()
38         {
39             System.out.println(num+"/"+den);

```

```

39     }
40 }
41 public static BigInteger sqrt(BigInteger a)
42 {
43     BigInteger b=a;
44     while (a.compareTo(b.multiply(b))<0)
45         b=b.multiply(b).add(a).divide(b.multiply(BigInteger.
            valueOf(2)));
46     return b;
47 }
48 public static boolean check(Fraction x,int n)
49 {
50     return x.num.multiply(x.num).add(x.den.multiply(x.den.multiply(
        BigInteger.valueOf(n))).negate()).compareTo(BigInteger.ONE)
        ==0;
51 }
52 static int p[]=new int[1000];
53 static int l;
54 public static void main(String[] args)
55 {
56     BigInteger ans=BigInteger.ZERO;
57     int idx=0;
58     for (int n=2,r=2;n<=1000;n++)
59     {
60         if (n==r*r)
61         {
62             r++;
63             continue;
64         }
65         int tmp=calc(n,0,1),a=tmp,b=n-tmp*tmp;
66         p[0]=tmp;
67         l=1;
68         while (true)
69         {
70             tmp=calc(n,a,b);
71             p[l++]=tmp;
72             a=a-tmp*b;
73             Fraction x=getFrac();
74             if (check(x,n))
75             {
76                 if (ans.compareTo(x.num)<0)
77                 {
78                     ans=x.num;
79                     idx=n;
80                 }
81                 break;
82             }
83             a=-a;
84             b=(n-a*a)/b;
85         }
86     }

```

```

87     System.out.println(idx);
88 }
89 private static Fraction getFrac() {
90     Fraction ret=new Fraction(p[l-1],1);
91     for (int i=l-2;i>=0;i--)
92         ret=new Fraction(p[i],1).add(ret.reciprocal());
93     return ret;
94 }
95 private static int calc(int n, int a, int b) {
96     for (long i=2;;i++)
97         if ((i*b-a)*(i*b-a)>n)
98             return (int)i-1;
99 }
100 }

```

3.21 其它公式

3.21.1 Polya

设 G 是 p 个对象的一个置换群，用 k 种颜色去染这 p 个对象，若一种染色方案在群 G 的作用下变为另一种方案，则这两个方案当作是同一种方案，这样的不同染色方案数为：

$$L = \frac{1}{|G|} \times \sum (k^{C(f)}), f \in G$$

$C(f)$ 为循环节， $|G|$ 表示群的置换方法数

对于有 n 个位置的手镯，有 n 种旋转置换和 n 种翻转置换

对于旋转置换：

$$C(f_i) = \gcd(n, i), i \text{ 表示一次转过 } i \text{ 颗宝石}, i = 0 \text{ 时 } c = n;$$

对于翻转置换：

如果 n 为偶数： 则有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2}$ ，有 $\frac{n}{2}$ 个置换 $C(f) = \frac{n}{2} + 1$

如果 n 为奇数： $C(f) = \frac{n}{2} + 1$

3.21.2 拉格朗日插值法

已知 $y = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$ 曲线上的 n 个点 $(x_1, y_1), (x_2, y_2), (x_3, y_3) \cdots (x_n, y_n)$ 用拉格朗日插值法可以不求系数可知任意 x 对应的 y 值。

$$\begin{aligned}
 y = & y_1 \frac{(x-x_2)(x-x_3) \cdots (x-x_n)}{(x_1-x_2)(x_1-x_3) \cdots (x_1-x_n)} \\
 & + y_2 \frac{(x-x_1)(x-x_3) \cdots (x-x_n)}{(x_2-x_1)(x_2-x_3) \cdots (x_2-x_n)} \\
 & + \cdots \\
 & + y_n \frac{(x-x_1)(x-x_2) \cdots (x-x_{n-1})}{(x_n-x_1)(x_n-x_2) \cdots (x_n-x_{n-1})}
 \end{aligned}$$

特别的, 如果 $x_1 \sim x_n$ 为连续自然数, 那么对于下一个自然数对应的 y 值为:

$$y_{n+1} = (-1)^{n-1} C_n^0 y_1 + (-1)^{n-2} C_n^1 y_2 + \cdots + (-1)^0 C_n^{n-1} y_n$$

这个组合系数可以通过高斯消元求出来, 前提是要猜到它满足递推关系。

3.21.3 正多面体顶点着色

$$\text{正四面体: } N = \frac{(n^4 + 11 \times n^2)}{12}$$

$$\text{正六面体: } N = \frac{(n^8 + 17 \times n^4 + 6 \times n^2)}{24}$$

$$\text{正八面体: } N = \frac{(n^6 + 3 \times n^4 + 12 \times n^3 + 8 \times n^2)}{24}$$

$$\text{正十二面体: } N = \frac{(n^{20} + 15 \times n^{10} + 20 \times n^8 + 24 \times n^4)}{60}$$

$$\text{正二十面体: } N = \frac{(n^{12} + 15 \times n^6 + 44 \times n^4)}{60}$$

3.21.4 求和公式

$$\sum k = \frac{n \times (n+1)}{2}$$

$$\sum 2k - 1 = n^2$$

$$\sum k^2 = \frac{n \times (n+1) \times (2n+1)}{6}$$

$$\sum (2k-1)^2 = \frac{n \times (4n^2-1)}{3}$$

$$\sum k^3 = \left(\frac{n \times (n+1)}{2} \right)^2$$

$$\sum (2k-1)^3 = n^2 \times (2n^2 - 1)$$

$$\sum k^4 = \frac{n \times (n+1) \times (2n+1) \times (3n^2 + 3n - 1)}{30}$$

$$\sum k^5 = \frac{n^2 \times (n+1)^2 \times (2n^2 + 2n - 1)}{12}$$

$$\sum k \times (k+1) = \frac{n \times (n+1) \times (n+2)}{3}$$

$$\sum k \times (k+1) \times (k+2) = \frac{n \times (n+1) \times (n+2) \times (n+3)}{4}$$

$$\sum k \times (k+1) \times (k+2) \times (k+3) = \frac{n \times (n+1) \times (n+2) \times (n+3) \times (n+4)}{5}$$

3.21.5 几何公式

球扇形:

全面积: $T = \pi r(2h + r_0)$, h 为球冠高, r_0 为球冠底面半径

体积: $V = \frac{2\pi r^2 h}{3}$

3.21.6 小公式

Pick 公式: $A = E \times 0.5 + I - 1$ (A 是多边形面积, E 是边界上的整点, I 是多边形内部的整点)

海伦公式: $S = \sqrt{p(p-a)(p-b)(p-c)}$, 其中 $p = \frac{(a+b+c)}{2}$, abc 为三角形的三条边长

求 $\binom{n}{k}$ 中素因子 P 的个数:

1. 把 n 转化为 P 进制, 并记它每个位上的和为 $S1$

2. 把 $n - k$, k 做同样的处理, 得到 S_2 , S_3

则 $\binom{n}{k}$ 中素因子 P 的个数: $\frac{S_2+S_3-S_1}{P-1}$

部分错排公式:

$n + m$ 个数中 m 个数必须错排求排列数

```
1 dp[i] = n*dp[i-1]+(i-1)*(dp[i-1]+dp[i-2]);
2 dp[0] = n!;
3 dp[1] = n*n!;
dp[m] 为所求解
```

3.21.7 马步问题

任意步长 (p, q) 无限棋盘可达性判定

```
1 bool check(int dx, int dy, int p, int q)
2 {
3     if (p < 0) p = -p;
4     if (q < 0) q = -q;
5     LL g = gcd(p, q);
6     if (dx % g || dy % g) return false;
7     dx /= g, dy /= g, p = (p / g) & 1, q = (q / g) & 1;
8     return !(p == q && ((dx ^ dy) & 1));
9 }
```

拓展:

若可选马步可以有 N 种 (p_i, q_i) , 令 $g = \gcd(p_1, q_1, p_2, q_2 \cdots p_N, q_N)$, 则不在 g 的整数倍点上的节点肯定不可达。坐标除 $2g$, 同时将可选马步除 g 之后放缩到 2×2 之内, 即 $(\frac{p_i}{g} \bmod 2, \frac{q_i}{g} \bmod 2)$ 。若放缩后马步中有 $(1, 0)$ 或 $(0, 1)$, 则全放缩后全棋盘可达, 否则只可达偶点。

$(2, 1)$ 马步无限棋盘最小距离

```
1 int dis(int dx, int dy)
2 {
3     if (dx < 0) dx = -dx;
4     if (dy < 0) dy = -dy;
5     if (dx < dy) swap(dx, dy);
6     if (dx & 1)
7     {
8         if (dy & 1) return dis(dx+1, dy-1);
9         if (dx == 1 && dy == 0) return 3;
10        return dis(dx+3, dy)-1;
11    }
12    if (dy & 1)
13    {
14        if (dx == 4 && dy == 3) return 3;
15        return dis(dx-2, dy-1)+1;
16    }
17    if (dx == 0 && dy == 0) return 0;
18    if (dx == 2 && dy == 2) return 4;
19    int c = (((dx-1) / 4)+1)*2;
20    if (dx & 2) dy -= 2;
```

```
21 | if (dy <= c) return c;  
22 | dy -= c;  
23 | return c+(dy-2) / 6*2+2;  
24 | }
```

4 数据结构

4.1 *Splay

持续学习中。

注意节点的 `size` 值不一定是真实的值！如果有需要需要特别维护！

1. 旋转和 Splay 操作
2. rank 操作
3. insert 操作（。。很多题目都有）
4. del 操作（郁闷的出纳员）
5. 由数组建立 Splay
6. 前驱后继（营业额统计）
7. Pushdown Pushup 的位置
8. *。。。暂时想不起了

4.1.1 节点定义

```

1 | const int MaxN = 50003;
2 |
3 | struct Node
4 | {
5 |     int size,key;
6 |
7 |     Node *c[2];
8 |     Node *p;
9 | } mem[MaxN], *cur, *nil;

```

无内存池的几个初始化函数。

```

1 | Node *newNode(int v, Node *p)
2 | {
3 |     cur->c[0] = cur->c[1] = nil, cur->p = p;
4 |     cur->size = 1;
5 |     cur->key = v;
6 |     return cur++;
7 | }
8 |
9 | void Init()
10 | {
11 |     cur = mem;
12 |     nil = newNode(0, cur);
13 |     nil->size = 0;
14 | }

```

带内存池的几个函数。


```

1  int emp[MaxN], totemp;
2
3  Node *newNode(int v, Node *p)
4  {
5      cur = mem + emp[--totemp];
6      cur->c[0] = cur->c[1] = nil, cur->p = p;
7      cur->size = 1;
8      cur->key = v;
9      return cur;
10 }
11
12 void Init()
13 {
14     for (int i = 0; i < MaxN; ++i)
15         emp[i] = i;
16     totemp = MaxN;
17     cur = mem + emp[--totemp];
18     nil = newNode(0, cur);
19     nil->size = 0;
20 }
21
22 void Recycle(Node *p)
23 {
24     if (p == nil) return;
25     Recycle(p->c[0]), Recycle(p->c[1]);
26     emp[totemp++] = p - mem;
27 }

```

4.1.2 维护序列

一切下标从 0 开始。

```

1  struct SplayTree
2  {
3      Node *root;
4      void Init()
5      {
6          root = nil;
7      }
8      void Pushup(Node *x)
9      {
10         if (x == nil) return;
11         Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
12         x->size = x->c[0]->size + x->c[1]->size + 1;
13     }
14     void Pushdown(Node *x)
15     {
16         if (x == nil) return;
17         //do something
18     }
19     void Rotate(Node *x, int f)
20     {

```

```

21     if (x == nil)    return;
22     Node *y = x->p;
23     y->c[f ^ 1] = x->c[f], x->p = y->p;
24     if (x->c[f] != nil)
25         x->c[f]->p = y;
26     if (y->p != nil)
27         y->p->c[y->p->c[1] == y] = x;
28     x->c[f] = y, y->p = x;
29     Pushup(y);
30 }
31 void Splay(Node *x, Node *f)
32 {
33     static Node *stack[maxn];
34     int top = 0;
35     stack[top++] = x;
36     for (Node *y = x; y != f; y = y->p)
37         stack[top++] = y->p;
38     while (top)
39         Pushdown(stack[--top]);
40
41     while (x->p != f)
42     {
43         Node *y = x->p;
44         if (y->p == f)
45             Rotate(x, x == y->c[0]);
46         else
47         {
48             int fd = y->p->c[0] == y;
49             if (y->c[fd] == x)
50                 Rotate(x, fd ^ 1), Rotate(x, fd);
51             else
52                 Rotate(y, fd), Rotate(x, fd);
53         }
54     }
55     Pushup(x);
56     if (f == nil)
57         root = x;
58 }
59 void Select(int k, Node *f)
60 {
61     Node *x = root;
62     Pushdown(x);
63     int tmp;
64     while ((tmp = x->c[0]->size) != k)
65     {
66         if (k < tmp)    x = x->c[0];
67         else
68             x = x->c[1], k -= tmp + 1;
69         Pushdown(x);
70     }
71     Splay(x, f);

```

```

72     }
73     void Select(int l, int r)
74     {
75         Select(l, nil), Select(r + 2, root);
76     }
77     Node *Make_tree(int a[], int l, int r, Node *p)
78     {
79         if (l > r) return nil;
80         int mid = l + r >> 1;
81         Node *x = newNode(a[mid], p);
82         x->c[0] = Make_tree(a, l, mid - 1, x);
83         x->c[1] = Make_tree(a, mid + 1, r, x);
84         Pushup(x);
85         return x;
86     }
87     void Insert(int pos, int a[], int n)
88     {
89         Select(pos, nil), Select(pos + 1, root);
90         root->c[1]->c[0] = Make_tree(a, 0, n - 1, root->c[1]);
91         Splay(root->c[1]->c[0], nil);
92     }
93     void Insert(int v)
94     {
95         Node *x = root, *y = nil;
96         while (x != nil)
97         {
98             y = x;
99             y->size++;
100            x = x->c[v >= x->key];
101        }
102        y->c[v >= y->key] = x = newNode(v, y);
103        Splay(x, nil);
104    }
105    void Remove(int l, int r)
106    {
107        Select(l, r);
108        //Recycle(root->c[1]->c[0]);
109        root->c[1]->c[0] = nil;
110        Splay(root->c[1], nil);
111    }
112 };

```

例题：旋转区间赋值求和求最大子序列。

注意打上懒标记后立即 Pushup。Pushup(root-c[1]-c[0]),Pushup(root-c[1]),Pushup(root);

```

1     void Pushup(Node *x)
2     {
3         if (x == nil) return;
4         Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
5         x->size = x->c[0]->size+x->c[1]->size+1;
6
7         x->sum = x->c[0]->sum+x->c[1]->sum+x->key;

```

```

8     x->lsum = max(x->c[0]->lsum,
9     x->c[0]->sum+x->key+max(0,x->c[1]->lsum));
10    x->rsum = max(x->c[1]->rsum,
11    x->c[1]->sum+x->key+max(0,x->c[0]->rsum));
12    x->maxsum = max(max(x->c[0]->maxsum,x->c[1]->maxsum),
13    x->key+max(0,x->c[0]->rsum)+max(0,x->c[1]->lsum));
14 }
15 void Pushdown(Node *x)
16 {
17     if (x == nil) return;
18     if (x->rev)
19     {
20         x->rev = 0;
21         x->c[0]->rev ^= 1;
22         x->c[1]->rev ^= 1;
23         swap(x->c[0],x->c[1]);
24
25         swap(x->lsum,x->rsum);
26     }
27     if (x->same)
28     {
29         x->same = false;
30         x->key = x->lazy;
31         x->sum = x->key*x->size;
32         x->lsum = x->rsum = x->maxsum = max(x->key,x->sum);
33         x->c[0]->same = true, x->c[0]->lazy = x->key;
34         x->c[1]->same = true, x->c[1]->lazy = x->key;
35     }
36 }
37
38 int main()
39 {
40     int totcas;
41     scanf("%d",&totcas);
42     for (int cas = 1;cas <= totcas;cas++)
43     {
44         Init();
45         sp.Init();
46         nil->lsum = nil->rsum = nil->maxsum = -Inf;
47         sp.Insert(0);
48         sp.Insert(0);
49
50         int n,m;
51         scanf("%d%d",&n,&m);
52         for (int i = 0;i < n;i++)
53             scanf("%d",&a[i]);
54         sp.Insert(0,a,n);
55
56         for (int i = 0;i < m;i++)
57         {
58             int pos,tot,c;

```

```

59     scanf("%s",buf);
60     if (strcmp(buf,"MAKE-SAME") == 0)
61     {
62         scanf("%d%d",&pos,&tot,&c);
63         sp.Select(pos-1,pos+tot-2);
64         sp.root->c[1]->c[0]->same = true;
65         sp.root->c[1]->c[0]->lazy = c;
66         sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
67     }
68     else if (strcmp(buf,"INSERT") == 0)
69     {
70         scanf("%d",&pos,&tot);
71         for (int i = 0;i < tot;i++)
72             scanf("%d",&a[i]);
73         sp.Insert(pos,a,tot);
74     }
75     else if (strcmp(buf,"DELETE") == 0)
76     {
77         scanf("%d",&pos,&tot);
78         sp.Remove(pos-1,pos+tot-2);
79     }
80     else if (strcmp(buf,"REVERSE") == 0)
81     {
82         scanf("%d",&pos,&tot);
83         sp.Select(pos-1,pos+tot-2);
84         sp.root->c[1]->c[0]->rev ^= 1;
85         sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
86     }
87     else if (strcmp(buf,"GET-SUM") == 0)
88     {
89         scanf("%d",&pos,&tot);
90         sp.Select(pos-1,pos+tot-2);
91         printf("%d\n",sp.root->c[1]->c[0]->sum);
92     }
93     else if (strcmp(buf,"MAX-SUM") == 0)
94     {
95         sp.Select(0,sp.root->size-3);
96         printf("%d\n",sp.root->c[1]->c[0]->maxsum);
97     }
98 }
99 }
100 return 0;
101 }

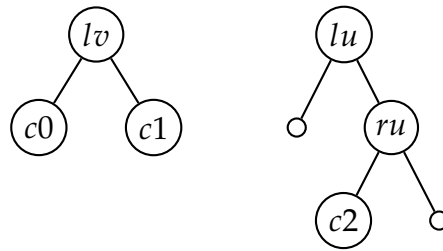
```

4.1.3 维护括号序列

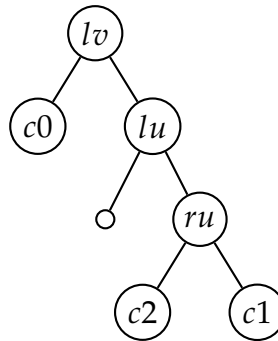
不需要哨兵。

合并操作：

先转成下面的样子：

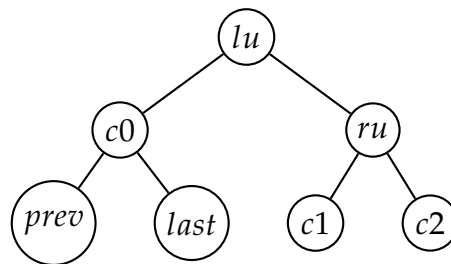


再链接成这样：

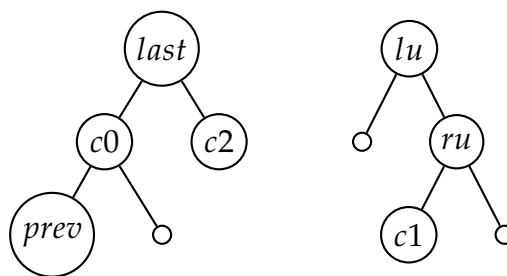


分离操作：

先把 lu 和 ru 转上去：



把 c0 和 c2 从原来的位置断开
然后接上：



```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  using namespace std;
5
6  const int maxn = 500000;
7  const int mod = 99990001;
8  struct Node
9  {
10     int size, key;
11

```

```

12  int a,b;
13  int minid,id;
14
15  Node *c[2];
16  Node *p;
17 }mem[maxn],*cur,*nil;
18 Node *l[maxn],*r[maxn];//左括号右括号定义在前面
19
20 int emp[maxn],totemp;
21 Node *newNode(int v,Node *p)
22 {
23     cur->c[0] = cur->c[1] = nil,cur->p = p;
24     cur->size = 1;
25     cur->key = v;
26
27     cur->a = 1;
28     cur->b = 0;
29     cur->minid = cur->id = maxn;
30
31     return cur++;
32 }
33 void Init()
34 {
35     cur = mem;
36     nil = newNode(0,cur);
37     nil->size = 0;
38 }
39
40 struct SplayTree
41 {
42     Node *root;
43     void Init()
44     {
45         root = nil;
46     }
47     void Pushup(Node *x)
48     {
49         if (x == nil) return;
50         Pushdown(x);
51         Pushdown(x->c[0]);
52         Pushdown(x->c[1]);
53         x->size = x->c[0]->size+x->c[1]->size+1;
54
55         x->minid = x->id;
56         for (int i = 0;i < 2;i++)
57             if (x->c[i] != nil)
58                 x->minid = min(x->minid,x->c[i]->minid);
59     }
60     void Pushdown(Node *x)
61     {
62         if (x == nil) return;

```

```

63
64     x->key = ((long long)x->key*x->a%mod+x->b)%mod;
65     for (int i = 0;i < 2;i++)
66         if (x->c[i] != nil)
67         {
68             x->c[i]->a = (long long)x->c[i]->a*x->a%mod;
69             x->c[i]->b = ((long long)x->c[i]->b*x->a%mod+x->b)%mod;
70         }
71     x->a = 1;
72     x->b = 0;
73 }
74 void Rotate(Node *x,int f)
75 {
76     if (x == nil) return;
77     Node *y = x->p;
78     y->c[f^1] = x->c[f], x->p = y->p;
79     if (x->c[f] != nil)
80         x->c[f]->p = y;
81     if (y->p != nil)
82         y->p->c[y->p->c[1] == y] = x;
83     x->c[f] = y, y->p = x;
84     Pushup(y);
85 }
86 void Splay(Node *x,Node *f)
87 {
88     static Node *stack[maxn];
89     int top = 0;
90     stack[top++] = x;
91     for (Node *y = x; y != f;y = y->p)
92         stack[top++] = y->p;
93     while (top)
94         Pushdown(stack[--top]);
95
96     while (x->p != f)
97     {
98         Node *y = x->p;
99         if (y->p == f)
100             Rotate(x,x == y->c[0]);
101         else
102         {
103             int fd = y->p->c[0] == y;
104             if (y->c[fd] == x)
105                 Rotate(x, fd^1), Rotate(x,fd);
106             else
107                 Rotate(y,fd), Rotate(x,fd);
108         }
109     }
110     Pushup(x);
111     if (f == nil)
112         root = x;
113 }

```



```

114 Node *Last(Node *now)
115 {
116     Splay(now,nil);
117     while (now->c[1] != nil)
118         now = now->c[1];
119     return now;
120 }
121 //把 u 接到 v 下面去, 边权为 w
122 //需要保证 u 是某棵树的根
123 void Link(int u,int v,int w)
124 {
125     Splay(l[v],nil);
126
127     Splay(l[u],nil);
128     l[u]->key = w;
129     Pushup(l[u]);
130     Splay(r[u],l[u]);
131
132     Node *c1 = l[v]->c[1];
133     l[v]->c[1] = l[u];
134     r[u]->c[1] = c1;
135     l[u]->p = l[v];
136     c1->p = r[u];
137     Pushup(r[u]);
138     Pushup(l[u]);
139     Pushup(l[v]);
140     Splay(l[u],nil);
141 }
142 //把 u 为根的子树分离开
143 int Split(int u)
144 {
145     Splay(l[u],nil);
146
147     int ret = l[u]->key;
148     Splay(r[u],l[u]);
149     Node *c0 = l[u]->c[0], *c2 = r[u]->c[1];
150
151     l[u]->key = 0; //去掉边权
152     l[u]->c[0] = r[u]->c[1] = c0->p = c2->p = nil;
153     Pushup(r[u]);
154     Pushup(l[u]);
155
156     Node *last = Last(c0);
157     Splay(last,nil);
158     last->c[1] = c2;
159     c2->p = last;
160     Pushup(last);
161
162     //对拆分后的两部份进行处理
163     Node *nu = last;
164     Node *nv = l[u];

```

```

165     if (nu->size > nv->size || (nu->size == nv->size && nu->minid >
        nv->minid))
166         swap(nu,nv);
167     nu->a = (long long)nu->a*ret%mod;
168     nu->b = (long long)nu->b*ret%mod;
169     nv->b = (nv->b+ret)%mod;
170
171     return ret;//返回原边权
172 }
173 };
174
175 SplayTree sp;
176 int n;
177 struct Edge
178 {
179     int to,next,w,id;
180 };
181 Edge edge[maxn];
182 int head[maxn],L;
183 int eid[maxn],toid[maxn];
184
185 void addedge(int u,int v,int w,int id)
186 {
187     edge[L].to = v;
188     edge[L].w = w;
189     edge[L].id = id;
190     edge[L].next = head[u];
191     head[u] = L++;
192 }
193
194 void DFS(int now,int fa)
195 {
196     for (int i = head[now];i != -1;i = edge[i].next)
197         if (edge[i].to != fa)
198         {
199             sp.Link(edge[i].to,now,edge[i].w);
200             eid[edge[i].id] = edge[i].to;
201             toid[edge[i].id] = now;
202
203             DFS(edge[i].to,now);
204         }
205 }
206
207 int main()
208 {
209     Init();
210     sp.Init();
211
212     scanf("%d",&n);
213
214     for (int i = 0;i < n;i++)

```

```

215 {
216     l[i] = newNode(0,nil);
217     r[i] = newNode(0,nil);
218     l[i]->id = r[i]->id = i;
219     l[i]->c[1] = r[i], r[i]->p = l[i];
220     sp.Pushup(l[i]);
221
222     head[i] = -1;
223 }
224 L = 0;
225
226 for (int i = 0; i < n-1; i++)
227 {
228     int u,v,w;
229     scanf("%d%d%d",&u,&v,&w);
230     u--,v--;
231
232     addedge(u,v,w,i);
233     addedge(v,u,w,i);
234 }
235
236 DFS(0,-1);
237
238 for (int i = 0; i < n-1; i++)
239 {
240     fflush(stdout);
241
242     int id;
243     scanf("%d",&id);
244     id--;
245
246     int ret = sp.Split(eid[id]);
247     printf("%d\n",ret);
248 }
249
250 return 0;
251 }

```

4.2 动态树

懒标记是否及时 Pushdown 了?
修改之后有没有及时 Pushup?

4.2.1 维护点权

查询链上的最长字段和
GetRoute 是用换根写的

```

1 | const int MaxN = 110000;
2 |

```

```

3 struct Node
4 {
5     int size, key;
6     bool rev;
7
8     // bool same;
9     // int lsum, rsum, sum, maxsum, sa;
10
11     Node *c[2];
12     Node *p;
13 } mem[MaxN], *cur, *nil, *pos[MaxN];
14
15 Node *newNode(int v, Node *p)
16 {
17     cur->c[0] = cur->c[1] = nil, cur->p = p;
18     cur->size = 1;
19     cur->key = v;
20     cur->rev = false;
21
22     // cur->same = false;
23     // cur->sa = 0;
24     // cur->lsum = cur->rsum = cur->maxsum = 0;
25     // cur->sum = v;
26
27     return cur++;
28 }
29
30 void Init()
31 {
32     cur = mem;
33     nil = newNode(0, cur);
34     nil->size = 0;
35 }
36
37 struct SplayTree
38 {
39     void Pushup(Node *x)
40     {
41         if (x == nil) return;
42         Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
43         x->size = x->c[0]->size + x->c[1]->size + 1;
44
45         // x->sum = x->c[0]->sum + x->c[1]->sum + x->key;
46         // x->lsum = max(x->c[0]->lsum,
47         //     x->c[0]->sum + x->key + max(0, x->c[1]->lsum));
48         // x->rsum = max(x->c[1]->rsum,
49         //     x->c[1]->sum + x->key + max(0, x->c[0]->rsum));
50         // x->maxsum = max(max(x->c[0]->maxsum, x->c[1]->maxsum),
51         //     x->key + max(0, x->c[0]->rsum) + max(0, x->c[1]->lsum));
52
53     }

```

```

54 void Pushdown(Node *x)
55 {
56     if (x == nil) return;
57     if (x->rev)
58     {
59         x->rev = 0;
60         x->c[0]->rev ^= 1;
61         x->c[1]->rev ^= 1;
62         swap(x->c[0], x->c[1]);
63 //注意修改与位置有关的量
64 //     swap(x->lsum, x->rsum);
65     }
66
67 //     if (x->same)
68 //     {
69 //         x->same = false;
70 //         x->key = x->sa;
71 //         x->sum = x->sa * x->size;
72 //         x->lsum = x->rsum = x->maxsum = max(0, x->sum);
73 //         if (x->c[0] != nil)
74 //             x->c[0]->same = true, x->c[0]->sa = x->sa;
75 //         if (x->c[1] != nil)
76 //             x->c[1]->same = true, x->c[1]->sa = x->sa;
77 //     }
78 }
79 bool isRoot(Node *x)
80 {
81     return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x);
82 }
83 void Rotate(Node *x, int f)
84 {
85     if (isRoot(x)) return;
86     Node *y = x->p;
87     y->c[f ^ 1] = x->c[f], x->p = y->p;
88     if (x->c[f] != nil)
89         x->c[f]->p = y;
90     if (y != nil)
91     {
92         if (y == y->p->c[1])
93             y->p->c[1] = x;
94         else if (y == y->p->c[0])
95             y->p->c[0] = x;
96     }
97     x->c[f] = y, y->p = x;
98     Pushup(y);
99 }
100 void Splay(Node *x)
101 {
102     static Node *stack[MaxN];
103     int top = 0;
104     stack[top++] = x;

```

```

105     for (Node *y = x; !isRoot(y); y = y->p)
106         stack[top++] = y->p;
107     while (top)
108         Pushdown(stack[--top]);
109
110     while (!isRoot(x))
111     {
112         Node *y = x->p;
113         if (isRoot(y))
114             Rotate(x, x == y->c[0]);
115         else
116         {
117             int fd = y->p->c[0] == y;
118             if (y->c[fd] == x)
119                 Rotate(x, fd ^ 1), Rotate(x, fd);
120             else
121                 Rotate(y, fd), Rotate(x, fd);
122         }
123     }
124     Pushup(x);
125 }
126 Node *Access(Node *u)
127 {
128     Node *v = nil;
129     while (u != nil)
130     {
131         Splay(u);
132         v->p = u;
133         u->c[1] = v;
134         Pushup(u);
135         u = (v = u)->p;
136         if (u == nil)
137             return v;
138     }
139 }
140 Node *LCA(Node *u, Node *v)
141 {
142     Access(u);
143     return Access(v);
144 }
145 Node *Link(Node *u, Node *v)
146 {
147     Access(u);
148     Splay(u);
149     u->rev = true;
150     u->p = v;
151 }
152 void ChangeRoot(Node *u)
153 {
154     Access(u)->rev ^= 1;
155 }

```

```

156 Node *GetRoute(Node *u, Node *v)
157 {
158     ChangeRoot(u);
159     return Access(v);
160 }
161 };
162
163 int n, m;
164 SplayTree sp;
165
166 int main(int argc, char const *argv[])
167 {
168     while (scanf("%d", &n) != EOF)
169     {
170         Init();
171         for (int i = 0; i < n; i++)
172         {
173             int v;
174             scanf("%d", &v);
175             pos[i] = newNode(v, nil);
176         }
177         for (int i = 0; i < n - 1; i++)
178         {
179             int u, v;
180             scanf("%d%d", &u, &v);
181             u--, v--;
182             sp.Link(pos[u], pos[v]);
183         }
184
185         // scanf("%d", &m);
186         // for (int i = 0; i < m; i++)
187         // {
188         //     int typ, u, v, c;
189         //     scanf("%d%d%d", &typ, &u, &v);
190         //     u--, v--;
191         //     if (typ == 1)
192         //         printf("%d\n", sp.GetRoute(pos[u], pos[v])->maxsum);
193         //     else
194         //     {
195         //         scanf("%d", &c);
196         //         Node *p = sp.GetRoute(pos[u], pos[v]);
197         //         p->same = true;
198         //         p->sa = c;
199         //     }
200         // }
201     }
202     return 0;
203 }

```

4.2.2 维护边权

刘汝佳的 Happy Painting!
 查询链上边的不同颜色数量
 不能换根，但是可以 Link 和 Cut

```

1  const int MaxN = 60000;
2
3  struct Node
4  {
5      int size, key;
6
7      int msk, lazy;
8
9      Node *c[2];
10     Node *p;
11 } mem[MaxN], *cur, *nil, *pos[MaxN];
12
13 Node *newNode(int v, Node *p)
14 {
15     cur->c[0] = cur->c[1] = nil, cur->p = p;
16     cur->size = 1;
17     cur->key = v;
18
19     cur->msk = 0;
20     cur->lazy = -1;
21
22     return cur++;
23 }
24
25 void Init()
26 {
27     cur = mem;
28     nil = newNode(0, cur);
29     nil->size = 0;
30 }
31
32 struct SplayTree
33 {
34     void Pushup(Node *x)
35     {
36         if (x == nil) return;
37         Pushdown(x);
38         Pushdown(x->c[0]);
39         Pushdown(x->c[1]);
40         x->size = x->c[0]->size + x->c[1]->size + 1;
41
42         x->msk = x->c[0]->msk | x->c[1]->msk | (1<<x->key);
43     }
44     void Pushdown(Node *x)
45     {

```



```

46     if (x == nil) return;
47
48     if (x->lazy != -1)
49     {
50         x->key = x->lazy;
51         x->msk = (1<<x->key);
52         x->c[0]->lazy = x->c[1]->lazy = x->lazy;
53         x->lazy = -1;
54     }
55 }
56 bool isRoot(Node *x)
57 {
58     return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x);
59 }
60 void Rotate(Node *x, int f)
61 {
62     if (isRoot(x)) return;
63     Node *y = x->p;
64     y->c[f ^ 1] = x->c[f], x->p = y->p;
65     if (x->c[f] != nil)
66         x->c[f]->p = y;
67     if (y != nil)
68     {
69         if (y == y->p->c[1])
70             y->p->c[1] = x;
71         else if (y == y->p->c[0])
72             y->p->c[0] = x;
73     }
74     x->c[f] = y, y->p = x;
75     Pushup(y);
76 }
77 void Splay(Node *x)
78 {
79     static Node *stack[MaxN];
80     int top = 0;
81     stack[top++] = x;
82     for (Node *y = x; !isRoot(y); y = y->p)
83         stack[top++] = y->p;
84     while (top)
85         Pushdown(stack[--top]);
86
87     while (!isRoot(x))
88     {
89         Node *y = x->p;
90         if (isRoot(y))
91             Rotate(x, x == y->c[0]);
92         else
93         {
94             int fd = y->p->c[0] == y;
95             if (y->c[fd] == x)
96                 Rotate(x, fd ^ 1), Rotate(x, fd);

```

```

97         else
98             Rotate(y, fd), Rotate(x, fd);
99     }
100 }
101 Pushup(x);
102 }
103 Node *Access(Node *u)
104 {
105     Node *v = nil;
106     while (u != nil)
107     {
108         Splay(u);
109         v->p = u;
110         u->c[1] = v;
111         Pushup(u);
112         u = (v = u)->p;
113         if (u == nil) return v;
114     }
115 }
116 Node *Root(Node *u)
117 {
118     Access(u);
119     Splay(u);
120     for (Pushdown(u); u->c[0] != nil; u = u->c[0])
121         Pushdown(u);
122     Splay(u);
123     return u;
124 }
125 Node *LCA(Node *u, Node *v)
126 {
127     if (Root(u) != Root(v))
128         return nil;
129     Access(u);
130     return Access(v);
131 }
132 void Cut(Node *u)
133 {
134     Access(u);
135     Splay(u);
136     u->c[0] = u->c[0]->p = nil;
137     Pushup(u);
138 }
139 void Link(Node *u, Node *v, int val)
140 {
141     Access(u);
142     Splay(u);
143     u->p = v;
144     u->key = val;
145     Pushup(u);
146 }
147 };

```

```

148
149 int cntbit(int x)
150 {
151     x = (x & 0x55555555) + ((x >> 1) & 0x55555555);
152     x = (x & 0x33333333) + ((x >> 2) & 0x33333333);
153     x = (x & 0x0F0F0F0F) + ((x >> 4) & 0x0F0F0F0F);
154     x = (x & 0x00FF00FF) + ((x >> 8) & 0x00FF00FF);
155     x = (x & 0x0000FFFF) + ((x >> 16) & 0x0000FFFF);
156     return x;
157 }
158
159 SplayTree sp;
160 int n,Q,f[MaxN];
161
162 int main(int argc, char const *argv[])
163 {
164     while (scanf("%d%d",&n,&Q) != EOF)
165     {
166         Init();
167         for (int i = 0; i < n; i++)
168         {
169             scanf("%d",&f[i]);
170             pos[i] = newNode(0, nil);
171         }
172         for (int i = 0; i < n; i++)
173         {
174             int col;
175             scanf("%d",&col);
176             if (f[i] > 0)
177                 sp.Link(pos[i],pos[f[i]-1],col-1);
178         }
179         for (int q = 0; q < Q; q++)
180         {
181             int typ,x,y,c;
182             scanf("%d%d%d",&typ,&x,&y);
183             x--,y--;
184             if (typ == 3)
185             {
186                 Node *lca = sp.LCA(pos[x],pos[y]);
187                 if (lca == nil || x == y)
188                 {
189                     printf("0_0\n");
190                     continue;
191                 }
192                 int totedge = lca->c[1]->size;
193                 int msk = lca->c[1]->msk;
194
195                 if (pos[x] != lca)
196                 {
197                     sp.Splay(pos[x]);
198                     totedge += pos[x]->size;

```

```

199         msk |= pos[x]→msk;
200     }
201
202     printf("%d_%d\n", totedge, cntbit(msk));
203 }
204 else
205 {
206     scanf("%d",&c);
207     c--;
208     if (typ == 1)
209     {
210         if (x == y) continue;
211
212         Node *lca = sp.LCA(pos[x], pos[y]);
213         if (pos[x] == lca) continue;
214
215         sp.Cut(pos[x]);
216         sp.Link(pos[x], pos[y], c);
217     }
218     else
219     {
220         Node *lca = sp.LCA(pos[x], pos[y]);
221
222         if (lca == nil || x == y)
223             continue;
224
225         lca→c[1]→lazy = c;
226         sp.Pushup(lca→c[1]);
227         sp.Pushup(lca);
228         if (pos[x] != lca)
229         {
230             sp.Splay(pos[x]);
231             pos[x]→lazy = c;
232             sp.Pushup(pos[x]);
233         }
234     }
235 }
236 }
237 }
238 }
239 return 0;
240 }

```

4.3 可持久化线段树

区间第 k 小数，内存压缩版，POJ2014。

```

1 #include <cstdio>
2 #include <algorithm>
3 using namespace std;
4
5 const int MAXN=100000,MAXM=100000;

```

```
6
7 struct node
8 {
9     node *l,*r;
10    int sum;
11 }tree[MAXN*4+MAXM*20];
12
13 int N;
14 node *newnode()
15 {
16     tree[N].l=tree[N].r=NULL;
17     tree[N].sum=0;
18     return &tree[N++];
19 }
20 node *newnode(node *x)
21 {
22     tree[N].l=x->l;
23     tree[N].r=x->r;
24     tree[N].sum=x->sum;
25     return &tree[N++];
26 }
27 node *build(int l,int r)
28 {
29     node *x=newnode();
30     if (l<r)
31     {
32         int mid=l+r>>1;
33         x->l=build(l,mid);
34         x->r=build(mid+1,r);
35         x->sum=x->l->sum+x->r->sum;
36     }
37     else
38         x->sum=0;
39     return x;
40 }
41 node *update(node *x,int l,int r,int p,int v)
42 {
43     if (l<r)
44     {
45         int mid=l+r>>1;
46         node *nx=newnode(x);
47         if (p<=mid)
48         {
49             node *ret=update(x->l,l,mid,p,v);
50             nx->l=ret;
51         }
52         else
53         {
54             node *ret=update(x->r,mid+1,r,p,v);
55             nx->r=ret;
56         }
```

```

57     nx->sum=nx->l->sum+nx->r->sum;
58     return nx;
59 }
60 else
61 {
62     node *nx=newnode(x);
63     nx->sum+=v;
64     return nx;
65 }
66 }
67 int query(node *x1,node *x2,int l,int r,int k)
68 {
69     if (l<r)
70     {
71         int mid=l+r>>1;
72         int lsum=x2->l->sum-x1->l->sum;
73         if (lsum>=k)
74             return query(x1->l,x2->l,l,mid,k);
75         else
76             return query(x1->r,x2->r,mid+1,r,k-lsum);
77     }
78     else
79         return l;
80 }
81 char s[10];
82 node *root[MAXM+1];
83 int a[MAXN],b[MAXN];
84 int init(int n)
85 {
86     for (int i=0;i<n;i++)
87         b[i]=a[i];
88     sort(b,b+n);
89     int tn=unique(b,b+n)-b;
90     for (int i=0;i<n;i++)
91     {
92         int l=0,r=tn-1;
93         while (l<r)
94         {
95             int mid=l+r>>1;
96             if (b[mid]>=a[i])
97                 r=mid;
98             else
99                 l=mid+1;
100         }
101         a[i]=l;
102     }
103     return tn;
104 }
105 int main()
106 {
107     int cas=1,n;

```

```

108 while (scanf("%d",&n)!=EOF)
109 {
110     printf("Case_␣%d:\n",cas++);
111     for (int i=0;i<n;i++)
112         scanf("%d",&a[i]);
113     int tn=init(n);
114     N=0;
115     root[0]=build(0,tn-1);
116     for (int i=1;i<=n;i++)
117         root[i]=update(root[i-1],0,tn-1,a[i-1],1);
118     int m;
119     scanf("%d",&m);
120     for (int i=0;i<m;i++)
121     {
122         int s,t;
123         scanf("%d%d",&s,&t);
124         printf("%d\n",b[query(root[s-1],root[t],0,tn-1,t-s+2>>1)]);
125     }
126 }
127 return 0;
128 }

```

4.4 treap 正式版

支持翻转。

```

1 #include <cstdio>
2 #include <cstdlib>
3 #include <algorithm>
4 using namespace std;
5
6 const int MAXN = 100000;
7 const int MAXM = 100000;
8 const int inf = 0x7fffffff;
9 int a[MAXN];
10 struct Treap
11 {
12     int N;
13     Treap()
14     {
15         N = 0;
16         root = NULL;
17     }
18     void init()
19     {
20         N = 0;
21         root = NULL;
22     }
23     struct Treap_Node
24     {
25         Treap_Node *son[2]; //left & right
26         int value, fix;

```

```

27     bool lazy;
28     int size;
29     Treap_Node() {}
30     Treap_Node(int _value)
31     {
32         son[0] = son[1] = NULL;
33         value = _value;
34         fix = rand() * rand();
35         lazy = 0;
36         size = 1;
37     }
38     int sonSize(bool flag)
39     {
40         if (son[flag] == NULL)
41             return 0;
42         else
43             return son[flag]→size;
44     }
45 } node[MAXN], *root, *pos[MAXN];
46 void up(Treap_Node *p)
47 {
48     p→size = p→sonSize(0) + p→sonSize(1) + 1;
49 }
50 void down(Treap_Node *p)
51 {
52     if (!p→lazy)
53         return ;
54     for (int i = 0; i < 2; i++)
55         if (p→son[i])
56             p→son[i]→lazy = !p→son[i]→lazy;
57     swap(p→son[0], p→son[1]);
58     p→lazy = 0;
59 }
60 Treap_Node *merge(Treap_Node *p, Treap_Node *q)
61 {
62     if (p == NULL)
63         return q;
64     else if (q == NULL)
65         return p;
66     if (p→fix <= q→fix)
67     {
68         down(p);
69         p→son[1] = merge(p→son[1], q);
70         up(p);
71         return p;
72     }
73     else
74     {
75         down(q);
76         q→son[0] = merge(p, q→son[0]);
77         up(q);

```



```

78     return q;
79 }
80 }
81 pair<Treap_Node *, Treap_Node *> split(Treap_Node *p, int n)
82 {
83     if (p == NULL)
84         return make_pair((Treap_Node *)NULL, (Treap_Node *)NULL);
85     if (!n)
86         return make_pair((Treap_Node *)NULL, p);
87     if (n == p->size)
88         return make_pair(p, (Treap_Node *)NULL);
89     down(p);
90     if (p->sonSize(0) >= n)
91     {
92         pair<Treap_Node *, Treap_Node *> ret = split(p->son[0], n);
93         p->son[0] = ret.second;
94         up(p);
95         return make_pair(ret.first, p);
96     }
97     else
98     {
99         pair<Treap_Node *, Treap_Node *>
100         ret = split(p->son[1], n - p->sonSize(0) - 1);
101         p->son[1] = ret.first;
102         up(p);
103         return make_pair(p, ret.second);
104     }
105 }
106 int smalls(Treap_Node *p, int value)
107 {
108     if (p==NULL)
109         return 0;
110     if (p->value<=value)
111         return 1+p->sonSize(0)+smalls(p->son[1],value);
112     else
113         return smalls(p->son[0],value);
114 }
115 void insert(int value)
116 {
117     Treap_Node *p = &node[N++];
118     *p = Treap_Node(value);
119     pair<Treap_Node *, Treap_Node *>
120     ret = split(root, smalls(root, value));
121     root = merge(merge(ret.first, p), ret.second);
122 }
123 void remove(int value)
124 {
125     pair<Treap_Node *, Treap_Node *> ret = split(root, smalls(root,
126         value) - 1);
127     root = merge(ret.first, split(ret.second, 1).second);

```

```

128 Treap_Node *build(int s, int t)
129 {
130     int idx = t + s >> 1;
131     Treap_Node *p = &node[N++];
132     *p = Treap_Node(a[idx]);
133     pos[a[idx]] = p;
134     if (idx > s)
135         p = merge(build(s, idx - 1), p);
136     if (idx < t)
137         p = merge(p, build(idx + 1, t));
138     up(p);
139     return p;
140 }
141 void build(int n)
142 {
143     root = build(0, n - 1);
144 }
145 void *reverse(int s, int t)
146 {
147     pair<Treap_Node *, Treap_Node *> tmp1, tmp2;
148     tmp1 = split(root, s - 1);
149     tmp2 = split(tmp1.second, t - s + 1);
150     tmp2.first->lazy = !tmp2.first->lazy;
151     root = merge(tmp1.first, merge(tmp2.first, tmp2.second));
152 }
153 };
154 Treap treap;
155 int main()
156 {
157     treap.init();
158     int n;
159     scanf("%d", &n);
160     for (int i = 0; i < n; i++)
161         scanf("%d", &a[i]);
162     treap.build(n);
163 }

```

4.5 树链剖分

4.5.1 点权

```

1 #include <cstdio>
2 #include <cstring>
3 #include <cstdlib>
4 #include <algorithm>
5 using namespace std;
6 const int MAX = 12000;
7 const int LOG = 15;
8 const int oo = 0x3f3f3f3f;
9 struct Edge
10 {

```

```

11     int to, w, id;
12     Edge* next;
13 } memo[MAX<<1], *cur, *g[MAX], *pree[MAX], *solid[MAX], *valid[MAX];
14 int dp[MAX][LOG], pos[MAX], lst[MAX], dep[MAX], cnt[MAX], h[MAX], K
    , n;
15 void init()
16 {
17     for (int i = 1; i <= n; i++)
18     {
19         g[i] = NULL;
20         valid[i] = NULL;
21         solid[i] = NULL;
22         pree[i] = NULL;
23     }
24     for (int i = 0; i < LOG; i++)
25     {
26         dp[1][i] = 1;
27     }
28     cur = memo;
29     K = 0;
30 }
31 void add(int u, int v, int w, int id)
32 {
33     cur->to = v;
34     cur->w = w;
35     cur->id = id;
36     cur->next = g[u];
37     g[u] = cur++;
38 }
39 void dfsLCA(int d, int u, int f)
40 {
41     dep[u] = d;
42     dp[u][0] = f;
43     cnt[u] = 1;
44     for (int i = 1; i < LOG; i++)
45     {
46         dp[u][i] = dp[dp[u][i - 1]][i - 1];
47     }
48     for (Edge* it = g[u]; it; it = it->next)
49     {
50         int v = it->to;
51         if (v != f)
52         {
53             pree[v] = it;
54             valid[it->id] = it;
55             dfsLCA(d + 1, v, u); //RE
56             cnt[u] += cnt[v];
57             if (solid[u] == NULL || cnt[solid[u]->to] < cnt[v])
58             {
59                 solid[u] = it;
60             }
61         }
62     }
63 }

```

```

61     }
62 }
63 }
64 void dfsChain(int u, int head)
65 {
66     h[u] = head;
67     if (solid[u])
68     {
69         lst[pos[u] = K++] = u;
70         dfsChain(solid[u]→to, head);
71     }
72     else
73     for (Edge* it = g[u]; it; it = it→next)
74     {
75         int v = it→to;
76         if (it != solid[u] && v != dp[u][0])
77         {
78             dfsChain(v, v);
79         }
80     }
81 }
82 int getLCA(int u, int v)
83 {
84     if (dep[u] < dep[v])
85         swap(u, v);
86     for (int st = 1 << (LOG - 1), i = LOG - 1; i >= 0; i—, st >>= 1)
87     {
88         if (st <= dep[u] - dep[v])
89         {
90             u = dp[u][i];
91         }
92     }
93     if (u == v)
94         return u;
95     for (int i = LOG - 1; i >= 0; i—)
96     {
97         if (dp[u][i] != dp[v][i])
98         {
99             u = dp[u][i];
100            v = dp[v][i];
101        }
102    }
103    return dp[u][0];
104 }
105 struct Node
106 {
107     int l, r, ma, mi;
108     bool rev;
109 } seg[MAX << 2];
110 void reverse(int k)
111 {

```

```

112     seg[k].mi *= -1;
113     seg[k].ma *= -1;
114     seg[k].rev ^= 1;
115     swap(seg[k].mi, seg[k].ma);
116 }
117 void pushdown(int k)
118 {
119     if (seg[k].rev)
120     {
121         reverse(k << 1);
122         reverse(k << 1 | 1);
123         seg[k].rev = false;
124     }
125 }
126 void update(int k)
127 {
128     seg[k].mi = min(seg[k << 1].mi, seg[k << 1 | 1].mi);
129     seg[k].ma = max(seg[k << 1].ma, seg[k << 1 | 1].ma);
130 }
131 void init(int k, int l, int r)
132 {
133     seg[k].l = l;
134     seg[k].r = r;
135     seg[k].rev = false;
136     if (l == r)
137     {
138         seg[k].mi = seg[k].ma = solid[lst[l]]->w; //solid WA
139         return;
140     }
141     int mid = l + r >> 1;
142     init(k << 1, l, mid);
143     init(k << 1 | 1, mid + 1, r);
144     update(k);
145 }
146 void update(int k, int id, int v)
147 {
148     if (seg[k].l == seg[k].r)
149     {
150         seg[k].mi = seg[k].ma = solid[lst[id]]->w = v;
151         return;
152     }
153     pushdown(k);
154     int mid = seg[k].l + seg[k].r >> 1;
155     if (id <= mid)
156         update(k << 1, id, v);
157     else
158         update(k << 1 | 1, id, v);
159     update(k);
160 }
161 void reverse(int k, int l, int r)
162 {

```

```

163     if (seg[k].l > r || seg[k].r < l)
164         return;
165     if (seg[k].l >= l && seg[k].r <= r)
166     {
167         reverse(k);
168         return;
169     }
170     pushdown(k);
171     reverse(k << 1, l, r);
172     reverse(k << 1 | 1, l, r);
173     update(k);
174 }
175 int read(int k, int l, int r)
176 {
177     if (seg[k].l > r || seg[k].r < l)
178         return -oo;
179     if (seg[k].l >= l && seg[k].r <= r)
180         return seg[k].ma;
181     pushdown(k);
182     return max(read(k << 1, l, r), read(k << 1 | 1, l, r));
183 }
184 void setEdge(int id, int v)
185 {
186     Edge* it = valid[id];
187     if (h[it->to] != it->to)
188     {
189         update(1, pos[dp[it->to][0]], v);
190     }
191     else
192     {
193         it->w = v;
194     }
195 }
196 void negateLCA(int t, int u)
197 {
198     while (t != u)
199     {
200         int tmp = h[u];
201         if (dep[tmp] < dep[t])
202             tmp = t;
203         if (h[u] == u)
204         {
205             pree[u]->w *= -1;
206             u = dp[u][0];
207         }
208         else
209         {
210             reverse(1, pos[tmp], pos[dp[u][0]]);
211             u = tmp;
212         }
213     }

```

```

214 }
215 void negate(int u, int v)
216 {
217     int t = getLCA(u, v);
218     negateLCA(t, u);
219     negateLCA(t, v);
220 }
221 int maxLCA(int t, int u)
222 {
223     int ret = -oo;
224     while (t != u)
225     {
226         int tmp = h[u];
227         if (dep[tmp] < dep[t])
228             tmp = t;
229         if (h[u] == u)
230         {
231             ret = max(ret, pree[u]—>w);
232             u = dp[u][0];
233         }
234         else
235         {
236             ret = max(ret, read(1, pos[tmp], pos[dp[u][0]]));
237             u = tmp;
238         }
239     }
240     return ret;
241 }
242 int query(int u, int v)
243 {
244     int t = getLCA(u, v);
245     return max(maxLCA(t, u), maxLCA(t, v));
246 }
247 int main()
248 {
249     int T;
250     int u, v, w;
251     char op[15];
252     scanf("%d", &T);
253     while (T—)
254     {
255         scanf("%d", &n);
256         init();
257         for (int i = 1; i < n; i++)
258         {
259             scanf("%d%d%d", &u, &v, &w);
260             add(u, v, w, i);
261             add(v, u, w, i);
262         }
263         dfsLCA(0, 1, 1);
264         dfsChain(1, 1);

```

```

265     init(1, 0, K - 1);
266     while (scanf("%s", op), op[0] != 'D')
267     {
268         scanf("%d%d", &u, &v);
269         if (op[0] == 'C')
270         {
271             setEdge(u, v);
272         }
273         else if (op[0] == 'N')
274         {
275             negate(u, v);
276         }
277         else
278         {
279             printf("%d\n", query(u, v));
280         }
281     }
282 }
283 return 0;
284 }

```

4.5.2 边权

```

1  #include <cstdio>
2  #include <iostream>
3  #include <cstdlib>
4  #include <algorithm>
5  #include <cmath>
6  #include <cstring>
7  using namespace std;
8  int n,m,sum,pos;
9  int head[50005],e;
10 int s[50005],from[50005];
11 int fa[50005][20],deep[50005],num[50005];
12 int solid[50005],p[50005],fp[50005];
13 struct N
14 {
15     int l,r,mid;
16     int add,w;
17 }nod[50005*4];
18 struct M
19 {
20     int v,next;
21 }edge[100005];
22 void addedge(int u,int v)
23 {
24     edge[e].v=v;
25     edge[e].next=head[u];
26     head[u]=e++;
27
28     edge[e].v=u;
29     edge[e].next=head[v];

```



```

30     head[v]=e++;
31 }
32 void LCA(int st,int f,int d)
33 {
34     deep[st]=d;
35     fa[st][0]=f;
36     num[st]=1;
37     int i,v;
38     for(i=1;i<20;i++)
39         fa[st][i]=fa[fa[st][i-1]][i-1];
40     for(i=head[st];i!=-1;i=edge[i].next)
41     {
42         v=edge[i].v;
43         if(v!=f)
44         {
45             LCA(v,st,d+1);
46             num[st]+=num[v];
47             if(solid[st]==-1||num[v]>num[solid[st]])
48                 solid[st]=v;
49         }
50     }
51 }
52 void getpos(int st,int sp)
53 {
54     from[st]=sp;
55     if(solid[st]!=-1)
56     {
57         p[st]=pos++;
58         fp[p[st]]=st;
59         getpos(solid[st],sp);
60     }
61     else
62     {
63         p[st]=pos++;
64         fp[p[st]]=st;
65         return;
66     }
67     int i,v;
68     for(i=head[st];i!=-1;i=edge[i].next)
69     {
70         v=edge[i].v;
71         if(v!=solid[st]&&v!=fa[st][0])
72             getpos(v,v);
73     }
74 }
75 int getLCA(int u,int v)
76 {
77     if(deep[u]<deep[v])
78         swap(u,v);
79     int d=1<<19,i;
80     for(i=19;i>=0;i--)

```

```

81     {
82         if(d<=deep[u]-deep[v])
83             u=fa[u][i];
84         d>>=1;
85     }
86     if(u==v)
87         return u;
88     for(i=19;i>=0;i--)
89         if(fa[u][i]!=fa[v][i])
90         {
91             u=fa[u][i];
92             v=fa[v][i];
93         }
94     return fa[u][0];
95 }
96 void init(int p,int l,int r)
97 {
98     nod[p].l=l;
99     nod[p].r=r;
100    nod[p].mid=(l+r)>>1;
101    nod[p].add=0;
102    if(l==r)
103        nod[p].w=s[fp[l]];
104    else
105    {
106        init(p<<1,l,nod[p].mid);
107        init(p<<1|1,nod[p].mid+1,r);
108    }
109 }
110 void lazy(int p)
111 {
112     if(nod[p].add!=0)
113     {
114         nod[p<<1].add+=nod[p].add;
115         nod[p<<1|1].add+=nod[p].add;
116         nod[p].add=0;
117     }
118 }
119 void update(int p,int l,int r,int v)
120 {
121     if(nod[p].l==l&&nod[p].r==r)
122     {
123         nod[p].add+=v;
124         return;
125     }
126     lazy(p);
127     if(nod[p].mid<l)
128         update(p<<1|1,l,r,v);
129     else if(nod[p].mid>=r)
130         update(p<<1,l,r,v);
131     else

```

```

132     {
133         update(p<<1,l,nod[p].mid,v);
134         update(p<<1|1,nod[p].mid+1,r,v);
135     }
136 }
137 int read(int p,int l,int r)
138 {
139     if(nod[p].l==l&&nod[p].r==r)
140         return nod[p].w+nod[p].add;
141     lazy(p);
142     if(nod[p].mid<l)
143         return read(p<<1|1,l,r);
144     else if(nod[p].mid>=r)
145         return read(p<<1,l,r);
146 }
147 void jump(int st,int ed,int val)
148 {
149     while(deep[st]>=deep[ed])
150     {
151         int tmp=from[st];
152         if(deep[tmp]<deep[ed])
153             tmp=ed;
154         update(1,p[tmp],p[st],val);
155         st=fa[tmp][0];
156     }
157 }
158 void change(int st,int ed,int val)
159 {
160     int lca=getLCA(st,ed);
161     jump(st,lca,val);
162     jump(ed,lca,val);
163     jump(lca,lca,-val);
164 }
165 int main()
166 {
167     while(scanf("%d%d%d",&n,&m,&sum)==3)
168     {
169         int i;
170         s[0]=0;pos=0;deep[0]=-1;
171         memset(fa,0,sizeof(fa));
172         for(i=1;i<=n;i++)
173         {
174             solid[i]=-1;
175             scanf("%d",&s[i]);
176         }
177         memset(head,-1,sizeof(head));
178         e=0;
179         for(i=0;i<m;i++)
180         {
181             int a,b;
182             scanf("%d%d",&a,&b);

```

```

183     addedge(a,b);
184 }
185 LCA(1,0,0);
186 getpos(1,1);
187 init(1,0,pos-1);
188 for(i=0;i<sum;i++)
189 {
190     char que[5];
191     scanf("%s",que);
192     if(que[0]!='Q')
193     {
194         int a,b,c;
195         scanf("%d%d%d",&a,&b,&c);
196         if(que[0]=='D')
197             c=-c;
198         change(a,b,c);
199     }
200     else
201     {
202         int a;
203         scanf("%d",&a);
204         printf("%d\n",read(1,p[a],p[a]));
205     }
206 }
207 }
208 return 0;
209 }

```

4.6 划分树

```

1  int n,m;
2  struct elem
3  {
4      int v,index;
5  }a[120000];
6  int d[30][120000];
7  int s[30][120000];
8
9  bool cmp(elem a,elem b)
10 {
11     if (a.v == b.v)
12         return a.index <= b.index;
13     return a.v < b.v;
14 }
15
16 void build(int depth,int l,int r)
17 {
18     if (l == r)
19         return;
20     int mid = (l+r)/2;
21     int tl,tr;
22     tl = tr = 0;

```

```

23  for (int i = l; i <= r; i++)
24  {
25      if (cmp(a[d[depth][i]], a[mid]))
26      {
27          d[depth+1][l+tl] = d[depth][i];
28          tl++;
29      }
30      else
31      {
32          d[depth+1][mid+1+tr] = d[depth][i];
33          tr++;
34      }
35      s[depth][i] = tl;
36  }
37  build(depth+1, l, mid);
38  build(depth+1, mid+1, r);
39 }
40
41 int find(int depth, int dl, int dr, int fl, int fr, int k)
42 {
43     if (fl == fr)
44         return a[d[depth][fl]].v;
45     int ls, rs;
46     int mid = (dl+dr)/2;
47     ls = (fl == dl)? 0 : s[depth][fl-1];
48     rs = s[depth][fr];
49     return (rs-ls < k)?
50         find(depth+1, mid+1, dr, mid+fl-dl-ls+1, mid+fr-dl-rs+1, k-(rs-ls))
51         : find(depth+1, dl, mid, dl+ls, dl+rs-1, k);
52 }
53
54 int main()
55 {
56     while (scanf("%d%d", &n, &m) != EOF)
57     {
58         for (int i = 1; i <= n; i++)
59         {
60             scanf("%d", &a[i].v);
61             a[i].index = i;
62         }
63         sort(a+1, a+n+1, cmp);
64         for (int i = 1; i <= n; i++)
65             d[0][a[i].index] = i;
66         build(0, 1, n);
67         int l, r, k;
68         for (int i = 1; i <= m; i++)
69         {
70             scanf("%d%d%d", &l, &r, &k);
71             printf("%d\n", find(0, 1, n, l, r, k));
72         }
73     }

```

```
74 | return 0;  
75 | }
```

4.7 树状数组

```
1 | int read(int k)  
2 | {  
3 |     int sum = 0;  
4 |     for (; k; k^=k&-k)  
5 |         sum+=tree[k];  
6 |     return sum;  
7 | }  
8 | void update(int k, int v)  
9 | {  
10 |     for (; k<=MaxN; k+=k&-k)  
11 |         tree[k]+=v;  
12 | }  
13 | int find_Kth(int k)  
14 | {  
15 |     int idx = 0;  
16 |     for(int i=20; i>=0; i--)  
17 |     {  
18 |         idx |= 1 << i;  
19 |         if(idx <= MaxN && tree[idx] < k)  
20 |             k -= tree[idx];  
21 |         else idx ^= 1 << i;  
22 |     }  
23 |     return idx + 1;  
24 | }
```

5 图论

5.1 优先队列优化的 dijkstra

```

1  #include<cstdio>
2  #include<cstring>
3  #include<iostream>
4  #include<algorithm>
5  #include<queue>
6  #include<vector>
7  using namespace std;
8  const int MAXN=100;
9  const int MAXM=1000;
10 int N,L;
11 int head[MAXN];
12 struct edges
13 {
14     int to,next,cost;
15 } edge[MAXM];
16 int dist[MAXN];
17 class states
18 {
19 public:
20     int cost,id;
21 };
22 class cmp
23 {
24 public:
25     bool operator()(const states &i,const states &j)
26     {
27         return i.cost>j.cost;
28     }
29 };
30 void init(int n)
31 {
32     N=n;
33     L=0;
34     for (int i=0; i<n; i++)
35         head[i]=-1;
36 }
37 void add_edge(int x,int y,int cost)
38 {
39     edge[L].to=y;
40     edge[L].cost=cost;
41     edge[L].next=head[x];
42     head[x]=L++;
43 }
44 int dijkstra(int s,int t)
45 {
46     memset(dist,63,sizeof(dist));
47     states u;

```

```

48     u.id=s;
49     u.cost=0;
50     dist[s]=0;
51     priority_queue<states,vector<states>,cmp> q;
52     q.push(u);
53     while (!q.empty())
54     {
55         u=q.top();
56         q.pop();
57         if (u.id==t) return dist[t];
58         if (u.cost!=dist[u.id]) continue;
59         for (int i=head[u.id]; i!=-1; i=edge[i].next)
60         {
61             states v=u;
62             v.id=edge[i].to;
63             if (dist[v.id]>dist[u.id]+edge[i].cost)
64             {
65                 v.cost=dist[v.id]=dist[u.id]+edge[i].cost;
66                 q.push(v);
67             }
68         }
69     }
70     return -1;
71 }
72 int main()
73 {
74     int n,m;
75     scanf("%d%d",&n,&m);
76     init(n);
77     for (int i=0; i<m; i++)
78     {
79         int x,y,z;
80         scanf("%d%d%d",&x,&y,&z);
81         add_edge(x,y,z);
82         add_edge(y,x,z);
83     }
84     int s,t;
85     scanf("%d%d",&s,&t);
86     printf("%d\n",dijkstra(s,t));
87     return 0;
88 }

```

5.2 SAP 四版

```

1  const int MAXEDGE=20400;
2  const int MAXN=400;
3  const int inf=0x3fffffff;
4  struct edges
5  {
6      int cap,to,next,flow;
7  } edge[MAXEDGE+100];
8  struct nodes

```



```

9 {
10     int head,label,pre,cur;
11 } node[MAXN+100];
12 int L,N;
13 int gap[MAXN+100];
14 void init(int n)
15 {
16     L=0;
17     N=n;
18     for (int i=0; i<N; i++)
19         node[i].head=-1;
20 }
21 void add_edge(int x,int y,int z,int w)
22 {
23     edge[L].cap=z;
24     edge[L].flow=0;
25     edge[L].to=y;
26     edge[L].next=node[x].head;
27     node[x].head=L++;
28     edge[L].cap=w;
29     edge[L].flow=0;
30     edge[L].to=x;
31     edge[L].next=node[y].head;
32     node[y].head=L++;
33 }
34 int maxflow(int s,int t)
35 {
36     memset(gap,0,sizeof(gap));
37     gap[0]=N;
38     int u,ans=0;
39     for (int i=0; i<N; i++)
40     {
41         node[i].cur=node[i].head;
42         node[i].label=0;
43     }
44     u=s;
45     node[u].pre=-1;
46     while (node[s].label<N)
47     {
48         if (u==t)
49         {
50             int min=inf;
51             for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
52                 if (min>edge[i].cap-edge[i].flow)
53                     min=edge[i].cap-edge[i].flow;
54             for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
55             {
56                 edge[i].flow+=min;
57                 edge[i^1].flow-=min;
58             }
59             u=s;

```

```

60     ans+=min;
61     continue;
62 }
63 bool flag=false;
64 int v;
65 for (int i=node[u].cur; i!=-1; i=edge[i].next)
66 {
67     v=edge[i].to;
68     if (edge[i].cap-edge[i].flow &&
69         node[v].label+1==node[u].label)
70     {
71         flag=true;
72         node[u].cur=node[v].pre=i;
73         break;
74     }
75 }
76 if (flag)
77 {
78     u=v;
79     continue;
80 }
81 node[u].cur=node[u].head;
82 int min=N;
83 for (int i=node[u].head; i!=-1; i=edge[i].next)
84     if (edge[i].cap-edge[i].flow && node[edge[i].to].label<min)
85         min=node[edge[i].to].label;
86 gap[node[u].label]--;
87 if (!gap[node[u].label]) return ans;
88 node[u].label=min+1;
89 gap[node[u].label]++;
90 if (u!=s) u=edge[node[u].pre^1].to;
91 }
92 return ans;
93 }

```

5.3 费用流三版

T 了可以改成栈。

```

1  const int MAXM=60000;
2  const int MAXN=400;
3  const int inf=0x3fffffff;
4  int L,N;
5  int K;
6  struct edges
7  {
8      int to,next,cap,flow,cost;
9  } edge[MAXM];
10 struct nodes
11 {
12     int dis,pre,head;
13     bool visit;

```

```
14 } node[MAXN];
15 void init(int n)
16 {
17     N=n;
18     L=0;
19     for (int i=0; i<N; i++)
20         node[i].head=-1;
21 }
22 void add_edge(int x,int y,int cap,int cost)
23 {
24     edge[L].to=y;
25     edge[L].cap=cap;
26     edge[L].cost=cost;
27     edge[L].flow=0;
28     edge[L].next=node[x].head;
29     node[x].head=L++;
30     edge[L].to=x;
31     edge[L].cap=0;
32     edge[L].cost=-cost;
33     edge[L].flow=0;
34     edge[L].next=node[y].head;
35     node[y].head=L++;
36 }
37 bool spfa(int s,int t)
38 {
39     queue<int> q;
40     for (int i=0; i<N; i++)
41     {
42         node[i].dis=0x3fffffff;
43         node[i].pre=-1;
44         node[i].visit=0;
45     }
46     node[s].dis=0;
47     node[s].visit=1;
48     q.push(s);
49     while (!q.empty())
50     {
51         int u=q.front();
52         node[u].visit=0;
53         for (int i=node[u].head; i!=-1; i=edge[i].next)
54         {
55             int v=edge[i].to;
56             if (edge[i].cap>edge[i].flow &&
57                 node[v].dis>node[u].dis+edge[i].cost)
58             {
59                 node[v].dis=node[u].dis+edge[i].cost;
60                 node[v].pre=u;
61                 if (!node[v].visit)
62                 {
63                     node[v].visit=1;
64                     q.push(v);
```

```

65     }
66     }
67     }
68     q.pop();
69 }
70 if (node[t].pre==--1)
71     return 0;
72 else
73     return 1;
74 }
75 int mcmf(int s,int t,int &cost)
76 {
77     int flow=0;
78     while (spfa(s,t))
79     {
80         int max=inf;
81         for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
82         {
83             if (max>edge[i].cap-edge[i].flow)
84                 max=edge[i].cap-edge[i].flow;
85         }
86         for (int i=node[t].pre; i!=-1; i=node[edge[i^1].to].pre)
87         {
88             edge[i].flow+=max;
89             edge[i^1].flow-=max;
90             cost+=edge[i].cost*max;
91         }
92         flow+=max;
93     }
94     return flow;
95 }

```

5.4 网络单纯形

返回 pair(流量, 费用)

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  using namespace std;
6
7  const int N = 1100;
8  const int M = 50000*2*5+2;
9  struct Node
10 {
11     int depth,pot,id;
12     Node *parent,*thread,*prev,*last;
13 };
14 struct Arc
15 {
16     int tail,head,u,c;

```

```
17 };
18 Arc arc[M];
19 int cnt_arc;
20 Node nnode[N],*rec[N];
21 int cr;
22 const int inf = 0x22222222;
23 void dfs(Node *n)
24 {
25     cr=0;
26     rec[cr++]=n;
27     while(cr)
28     {
29         Node *nn=n->thread;
30         nn->prev=n;
31         n=nn;
32         int d=n->depth;
33         if(d<=rec[cr-1]->depth)
34         {
35             Node *l=rec[cr-1];
36             while(cr&&d<=rec[cr-1]->depth)
37             {
38                 rec[--cr]->last=l;
39             }
40         }
41         if(cr) rec[cr++]=n;
42     }
43 }
44 Node*pivot(Node*n1,Node*n2,Node*n3,Node*n4)
45 {
46     dfs(n1);
47     Node *n1l=n1->last,*n3l,*n3p,*end;
48     if(n1l->thread==n2)
49     {
50         n2->parent=n1;
51         dfs(n3);
52         return n3->last->thread;
53     }
54     if(n1l->thread!=n3)
55     {
56         dfs(n4);
57         n3p=n3->prev;
58         n3l=n3->last;
59         n3p->thread=n3l->thread;
60         if(n3l==n1l)
61             n3l->thread=n1->thread,n1->thread=n2;
62         else
63             n3l->thread=n1l->thread,n1l->thread=n2;
64     }
65     else
66     {
67         dfs(n3);
```

```

68     n3l=n3->last;
69     n1l->thread=n2;
70 }
71 end=n3l->thread;
72 Node *n2l=n2->last,*n2lp=NULL,*p2=n1,*pp=NULL;
73 int pid;
74 int first=0;
75 while(n2!=n3)
76 {
77     n2l=n2->last;
78     Node *xp=pp;
79     if(n2l!=n2lp) xp=n2l,n2lp=n2l;
80     pp=n2->prev;
81     pp->thread=xp->thread;
82     xp->thread=n2->parent;
83     Node *n2n=n2->parent;
84     int ppid=n2->id;
85     if(first) n2->id=pid^1;
86     pid=ppid;
87     first++;
88     n2->parent=p2;
89     p2=n2;
90     n2=n2n;
91 }
92 if(first) n2->id=pid^1;
93 n2->parent=p2;
94 return end;
95 }
96 void update(Node *n1,Node *n2)
97 {
98     n1=n1->thread;
99     while(n1!=n2)
100     {
101         Node *n1p=n1->parent;
102         n1->depth=n1p->depth+1;
103         n1->pot = n1p->pot - arc[n1->id].c;
104         n1=n1->thread;
105     }
106 }
107 void pivot(int id)
108 {
109     int tail=arc[id].tail,head=arc[id].head;
110     int u=arc[id].u,c=arc[id].c;
111     Node *uu=nnode+tail,*vv=nnode+head;
112     Node *mu=0,*mv=0;
113     int mn_f = u;
114     while(uu!=vv)
115     {
116         int nid;
117         if(uu->depth>vv->depth)
118             nid=uu->id,uu=uu->parent;

```

```

119     else nid=vv->id^1,vv=vv->parent;
120     if(arc[nid].u<mn_f) mn_f=arc[nid].u;
121 }
122 uu=nnode+tail,vv=nnode+head;
123 arc[id].u-=mn_f;
124 arc[id^1].u+=mn_f;
125 while(uu!=vv)
126 {
127     int nid;
128     if(uu->depth>vv->depth)
129     {
130         nid=uu->id;
131         arc[nid].u-=mn_f;
132         arc[nid^1].u+=mn_f;
133         if(arc[nid].u==0&&mu==0)
134             mu=uu,mv=0;
135         uu=uu->parent;
136     }
137     else
138     {
139         nid=vv->id^1;
140         arc[nid].u-=mn_f;
141         arc[nid^1].u+=mn_f;
142         if(arc[nid].u==0&&mu==0)
143             mv=vv;
144         vv=vv->parent;
145     }
146 }
147 if(arc[id].u)
148 {
149     if(mv)
150     {
151         int n3=mv-nnode,n4=mv->parent-nnode;
152         Node *nx=pivot(nnode+tail,nnode+head,mv,mv->parent);
153         nnode[head].id=id;
154         update(nnode+tail,nx);
155     }
156     else
157     {
158         int n3=mu-nnode,n4=mu->parent-nnode;
159         Node *nx=pivot(nnode+head,nnode+tail,mu,mu->parent);
160         nnode[tail].id=id^1;
161         update(nnode+head,nx);
162     }
163 }
164 }
165 struct List
166 {
167     int v;
168     List *R;
169 };

```

```

170 List lst[M],*cl,*hd;
171 pair<int,int> cost(int s,int t,int cnt_node)
172 {
173     int tmp_arc=cnt_arc;
174     int i;
175     int sid=-1;
176     nnode[cnt_node].depth=0;
177     nnode[cnt_node].parent=nnode+cnt_node;
178     nnode[cnt_node].pot=0;
179     nnode[cnt_node].thread=nnode;
180     for(i=0; i<cnt_node; i++)
181     {
182         nnode[i].depth=1;
183         nnode[i].parent=nnode+cnt_node;
184         nnode[i].thread=nnode+(i+1);
185         nnode[i].id=tmp_arc;
186         int pot=-inf,u=0,tot=0;
187         if(i==s)
188             pot=-pot,tot=inf,u=inf,sid=tmp_arc+1;
189         if(i==t) tot=inf;
190         nnode[i].pot=pot;
191         arc[tmp_arc].tail=cnt_node;
192         arc[tmp_arc].head=i;
193         arc[tmp_arc].u=u;
194         arc[tmp_arc].c=-pot;
195         tmp_arc++;
196         arc[tmp_arc].tail=i;
197         arc[tmp_arc].head=cnt_node;
198         arc[tmp_arc].u=tot-u;
199         arc[tmp_arc].c=pot;
200         tmp_arc++;
201     }
202     while(1)
203     {
204         bool fd=false;
205         int i,tot=0;
206         hd=cl=lst;
207         cl->R=cl++;
208         int tail,head,c,cc;
209         const int MAXQ = cnt_node/4+5;
210         const int MINQ = cnt_node/10+2;
211         for(i=0; i<cnt_arc; i++)
212         {
213             if(arc[i].u==0) continue;
214             tail=arc[i].tail;
215             head=arc[i].head,c=arc[i].c;
216             cc=c-nnode[tail].pot+nnode[head].pot;
217             if(cc<0)
218             {
219                 fd=true;
220                 cl->v=i;

```



```

221     cl->R=hd->R;
222     hd->R=cl;
223     cl++;
224     tot++;
225     if(tot>=MAXQ)
226     {
227         while(tot>MINQ)
228         {
229             List *mj=NULL,*mjp,*it=hd->R,*itp=hd;
230             int mp=0;
231             while(it!=hd)
232             {
233                 int id=it->v;
234                 tail=arc[id].tail,head=arc[id].head,c=arc[id].c;
235                 cc=c-nnode[tail].pot+nnode[head].pot;
236                 if(cc>=0)
237                 {
238                     itp->R=it->R;
239                     tot--;
240                 }
241                 else if(cc<mp)
242                     mp=cc,mj=it,mjp=itp;
243                 if(itp->R==it) itp=it;
244                 it=it->R;
245             }
246             if(mj==NULL) break;
247             pivot(mj->v);
248             mjp->R=mj->R;
249             tot--;
250         }
251     }
252 }
253 }
254 while(tot)
255 {
256     List *it=hd->R,*itp=hd;
257     while(it!=hd)
258     {
259         int id=it->v;
260         tail=arc[id].tail,head=arc[id].head;
261         c=arc[id].c;
262         cc=c-nnode[tail].pot+nnode[head].pot;
263         if(cc<0) pivot(id);
264         itp->R=it->R;
265         tot--;
266         it=it->R;
267     }
268 }
269 if(!fd) break;
270 }
271 int sum=0;

```

```
272     for(i=0; i<cnt_arc; i+=2)
273     {
274         sum+=arc[i].c*arc[i^1].u;
275     }
276     return pair<int,int>(arc[sid].u,sum);
277 }
278 void add_arc(int tail,int head,int u,int c)
279 {
280     arc[cnt_arc].tail=tail;
281     arc[cnt_arc].head=head;
282     arc[cnt_arc].u=u;
283     arc[cnt_arc].c=c;
284     cnt_arc++;
285 }
286
287 void init()
288 {
289     cnt_arc = 0;
290 }
291
292 void add_edge(int x,int y,int cap,int cost)
293 {
294     add_arc(x,y,cap,cost);
295     add_arc(y,x,0,-cost);
296 }
297
298 int main()
299 {
300     int n,m,k;
301     while (scanf("%d%d%d",&n,&m,&k) != EOF)
302     {
303         init();
304         add_edge(0,1,k,0);
305         for (int i=0;i<m;i++)
306         {
307             int u,v,a,c;
308             scanf("%d%d%d%d",&u,&v,&a,&c);
309             for (int j=0;j<c;j++)
310                 add_edge(u,v,1,a*(j*2+1));
311         }
312         pair<int,int> ret = cost(0,n,n+1);
313         if (ret.first == k)
314             printf("%d\n",ret.second);
315         else
316             puts("-1");
317     }
318     return 0;
319 }
```

5.5 匈牙利

5.5.1 新版, 隐式图可解

```

1 bool check(int u)
2 {
3     for (int i=head[u]; i!=-1; i=edge[i].next)
4     {
5         int v=edge[i].to;
6         if (matc[v]==u) continue;
7         if (!use[v])
8         {
9             use[v]=1;
10            if (matc[v]==-1 || check(matc[v]))
11            {
12                matc[v]=u;
13                matc[u]=v;
14                return 1;
15            }
16        }
17    }
18    return 0;
19 }
20 int match()
21 {
22     int ret=0;
23     memset(matc,-1,sizeof(matc));
24     for (int u=0; u<N; u++)
25     {
26         if (matc[u]!=-1) continue;
27         memset(use,0,sizeof(use));
28         if (check(u))
29             ret++;
30     }
31     return ret;
32 }

```

5.5.2 邻接矩阵

```

1 bool check(int u)
2 {
3     for (int v=0; v<N; v++)
4         if (am[u][v] && !use[v])
5         {
6             use[v]=1;
7             if (pre[v]==-1 || check(pre[v]))
8             {
9                 pre[v]=u;
10                return 1;
11            }
12        }
13    return 0;
14 }

```

```

15 int match()
16 {
17     int ret=0;
18     memset(pre,-1,sizeof(pre));
19     for (int u=0; u<N; u++)
20     {
21         memset(use,0,sizeof(use));
22         if (check(u))
23             ret++;
24     }
25     return ret;
26 }

```

5.5.3 邻接表

```

1 bool check(int u)
2 {
3     for (int i=head[u]; i!=-1; i=edge[i].next)
4     {
5         int v=edge[i].to;
6         if (!use[v])
7         {
8             use[v]=1;
9             if (pre[v]==-1 || check(pre[v]))
10            {
11                pre[v]=u;
12                return 1;
13            }
14        }
15    }
16    return 0;
17 }
18 int match()
19 {
20     int ret=0;
21     memset(pre,-1,sizeof(pre));
22     for (int u=1; u<=N; u++)
23     {
24         memset(use,0,sizeof(use));
25         if (check(u))
26             ret++;
27     }
28     return ret;
29 }

```

5.6 一般图最大加权匹配

注意 G 初始化

```

1 #define N 229
2 int G[N][N];
3 int cnt_node;

```

```

4  int dist[N];
5  int rec[N],cr,M[N],P[N];
6  bool vst[N];
7  const int inf = 0x3f3f3f3f;
8  bool spfa(int u)
9  {
10     rec[cr++]=u;
11     if(vst[u]) return true;
12     vst[u]=true;
13     int v;
14     for(v=0; v<cnt_node; v++)
15     {
16         if(v!=u&&M[u]!=v&&!vst[v])
17         {
18             int w=M[v];
19             if(dist[w]<dist[u]+G[u][v]-G[v][w])
20             {
21                 dist[w]=dist[u]+G[u][v]-G[v][w];
22                 if(spfa(w))
23                 {
24                     return true;
25                 }
26             }
27         }
28     }
29     cr--;
30     vst[u]=false;
31     return false;
32 }
33 int match()
34 {
35     int i;
36     for(i=0; i<cnt_node; i++) P[i]=i;
37     for(i=0; i<cnt_node; i+=2) M[i]=i+1,M[i+1]=i;
38     int cnt=0;
39     while(1)
40     {
41         memset(dist,0,sizeof(dist));
42         cr=0;
43         int i;
44         bool fd=false;
45         memset(vst,0,sizeof(vst));
46         for(i=0; i<cnt_node; i++)
47         {
48             if(spfa(P[i]))
49             {
50                 fd=true;
51                 int j;
52                 int nx=M[rec[cr-1]];
53                 for(j=cr-2; rec[j]!=rec[cr-1]; j--)
54                 {

```

```

55         M[nx]=rec[j];
56         int tmp=nx;
57         nx=M[rec[j]];
58         M[rec[j]]=tmp;
59     }
60     M[nx]=rec[j];
61     M[rec[j]]=nx;
62     break;
63 }
64 }
65 if(!fd)
66 {
67     cnt++;
68     if(cnt>=3) break;
69     random_shuffle(P,P+cnt_node);
70 }
71 }
72 int sum=0;
73 for(i=0; i<cnt_node; i++)
74 {
75     int v=M[i];
76     if(i<v)
77     {
78         sum+=G[i][v];
79     }
80 }
81 return sum;
82 }

```

5.7 一般图匹配带花树

```

1  const int MaxN = 222;
2  int N;
3  bool Graph[MaxN+1][MaxN+1];
4  int Match[MaxN+1];
5  bool InQueue[MaxN+1], InPath[MaxN+1], InBlossom[MaxN+1];
6  int Head, Tail;
7  int Queue[MaxN+1];
8  int Start, Finish;
9  int NewBase;
10 int Father[MaxN+1], Base[MaxN+1];
11 int Count;
12 void CreateGraph()
13 {
14     int u, v;
15     memset(Graph, false, sizeof(Graph));
16     scanf("%d", &N);
17     while (scanf("%d%d", &u, &v) != EOF)
18         Graph[u][v] = Graph[v][u] = true;
19 }
20 void Push(int u)

```

```

21 {
22     Queue[Tail] = u;
23     Tail++;
24     InQueue[u] = true;
25 }
26 int Pop()
27 {
28     int res = Queue[Head];
29     Head++;
30     return res;
31 }
32 int FindCommonAncestor(int u,int v)
33 {
34     memset(InPath,false,sizeof(InPath));
35     while (true)
36     {
37         u = Base[u];
38         InPath[u] = true;
39         if (u == Start) break;
40         u = Father[Match[u]];
41     }
42     while (true)
43     {
44         v = Base[v];
45         if (InPath[v]) break;
46         v = Father[Match[v]];
47     }
48     return v;
49 }
50 void ResetTrace(int u)
51 {
52     int v;
53     while (Base[u] != NewBase)
54     {
55         v = Match[u];
56         InBlossom[Base[u]] = InBlossom[Base[v]] = true;
57         u = Father[v];
58         if (Base[u] != NewBase) Father[u] = v;
59     }
60 }
61 void BlossomContract(int u,int v)
62 {
63     NewBase = FindCommonAncestor(u,v);
64     memset(InBlossom,false,sizeof(InBlossom));
65     ResetTrace(u);
66     ResetTrace(v);
67     if (Base[u] != NewBase) Father[u] = v;
68     if (Base[v] != NewBase) Father[v] = u;
69     for (int tu = 1; tu <= N; tu++)
70         if (InBlossom[Base[tu]])
71         {

```

```

72     Base[tu] = NewBase;
73     if (!InQueue[tu]) Push(tu);
74 }
75 }
76 void FindAugmentingPath()
77 {
78     memset(InQueue, false, sizeof(InQueue));
79     memset(Father, 0, sizeof(Father));
80     for (int i = 1; i <= N; i++)
81         Base[i] = i;
82     Head = Tail = 1;
83     Push(Start);
84     Finish = 0;
85     while (Head < Tail)
86     {
87         int u = Pop();
88         for (int v = 1; v <= N; v++)
89             if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u] != v))
90             {
91                 if ((v == Start) ||
92                     ((Match[v] > 0) && (Father[Match[v]] > 0)))
93                     BlossomContract(u, v);
94                 else if (Father[v] == 0)
95                 {
96                     Father[v] = u;
97                     if (Match[v] > 0)
98                         Push(Match[v]);
99                     else
100                     {
101                         Finish = v;
102                         return;
103                     }
104                 }
105             }
106     }
107 }
108 void AugmentPath()
109 {
110     int u, v, w;
111     u = Finish;
112     while (u > 0)
113     {
114         v = Father[u];
115         w = Match[v];
116         Match[v] = u;
117         Match[u] = v;
118         u = w;
119     }
120 }
121 void Edmonds()
122 {

```



```

123     memset(Match,0,sizeof(Match));
124     for (int u = 1; u <= N; u++)
125         if (Match[u] == 0)
126         {
127             Start = u;
128             FindAugmentingPath();
129             if (Finish > 0) AugmentPath();
130         }
131 }
132 void PrintMatch()
133 {
134     for (int u = 1; u <= N; u++)
135         if (Match[u] > 0)
136             Count++;
137     printf("%d\n",Count);
138     for (int u = 1; u <= N; u++)
139         if (u < Match[u])
140             printf("%d_ %d\n",u,Match[u]);
141 }
142 int main()
143 {
144     CreateGraph();
145     Edmonds();
146     PrintMatch();
147 }

```

5.8 KM

5.8.1 最大加权匹配

```

1  bool visx[N],visy[N]; //x,y 中的点是否被访问
2  int lx[N],ly[N]; //x,y 中的点的标号
3  int matchy[N]; //y 中各点匹配状态
4  int map[N][N]; //二分图描述 [x][y]
5  bool find(int x)
6  {
7      visx[x]=true;
8      int t;
9      for (int y=0;y<ycnt;y++)
10     {
11         if (!visy[y])
12         {
13             t=lx[x]+ly[y]-map[x][y];
14             if (t==0)
15             {
16                 visy[y]=true;
17                 if (matchy[y]==-1 || find(matchy[y]))
18                 {
19                     matchy[y]=x;
20                     return true;
21                 }
22             }
23         }
24     }
25 }

```

```

23     else if (lack>t) lack=t;
24 }
25 }
26 return false;
27 }
28 void KM()
29 {
30     memset(lx,0,sizeof(lx));
31     memset(ly,0,sizeof(ly));
32     memset(matchy,-1,sizeof(matchy));
33     for (int i=0;i<xcnt;i++)
34         for (int j=0;j<ycnt;j++)
35             if (map[i][j]>lx[i])
36                 lx[i]=map[i][j];
37     for (int x=0;x<xcnt;x++)
38     {
39         while (true)
40         {
41             memset(visx,false,sizeof(visx));
42             memset(visy,false,sizeof(visy));
43             lack=INFI;
44             if (find(x)) break;
45             for (int i=0;i<xcnt;i++)
46             {
47                 if (visx[i]) lx[i]-=lack;
48                 if (visy[i]) ly[i]+=lack;
49             }
50         }
51     }
52     int cost=0;
53     for (int i=0;i<ycnt;i++)
54         cost+=map[matchy[i]][i];
55 }

```

5.8.2 自认为正确的 Kuhn_Munkras

未验证

```

1  #include<cstdio>
2  #include<cstring>
3  #include<algorithm>
4  using namespace std;
5  const int MAXN=100;
6  const int inf=0x3f3f3f3f;
7  bool visitx[MAXN],visity[MAXN];
8  int labx[MAXN],laby[MAXN],matx[MAXN],maty[MAXN],slack[MAXN];
9  int ma[MAXN][MAXN];
10 bool check(int x,int n)
11 {
12     visitx[x]=1;
13     for (int i=0; i<n; i++)
14         if (!visity[i])
15             if (labx[x]+laby[i]==ma[x][i])

```

```

16     {
17         visity[i]=1;
18         if (maty[i]==-1 || check(maty[i],n))
19         {
20             matx[x]=i;
21             maty[i]=x;
22             return 1;
23         }
24     }
25     else
26         slack[i]=min(slack[i],labx[x]+laby[i]-ma[x][i]);
27
28     return 0;
29 }
30 void maintain(int n)
31 {
32     int diff=inf;
33     for (int i=0; i<n; i++)
34         if (!visity[i])
35             diff=min(diff,slack[i]);
36     for (int i=0; i<n; i++)
37     {
38         if (visitx[i])
39             labx[i]-=diff;
40         if (visity[i])
41             laby[i]+=diff;
42         else
43             slack[i]-=diff;
44     }
45 }
46 int Kuhn_Munkras(int n)
47 {
48     for (int i=0; i<n; i++)
49     {
50         labx[i]=-inf;
51         for (int j=0; j<n; j++)
52             labx[i]=max(labx[i],ma[i][j]);
53     }
54     memset(laby,0,4*n);
55     memset(matx,-1,4*n);
56     memset(maty,-1,4*n);
57     for (int i=0; i<n; i++)
58     {
59         memset(visitx,0,n);
60         memset(visity,0,n);
61         memset(slack,63,4*n);
62         while (!check(i,n))
63         {
64             maintain(n);
65             memset(visitx,0,n);
66             memset(visity,0,n);

```

```

67     }
68 }
69 int ret=0;
70 for (int i=0;i<n;i++)
71     ret+=labx[i]+laby[i];
72 return ret;
73 }
74 int main()
75 {
76     int n,m;
77     scanf("%d%d",&m,&n);
78     for (int i=m; i<n; i++)
79         for (int j=0; j<n; j++)
80             ma[i][j]=0;
81     for (int i=0; i<m; i++)
82         for (int j=0; j<n; j++)
83             scanf("%d",&ma[i][j]);
84     printf("%d\n",Kuhn_Munkras(n));
85     printf("%d",matx[0]+1);
86     for (int i=1;i<m;i++)
87         printf("□%d",matx[i]+1);
88     puts("");
89     return 0;
90 }

```

5.9 * 二维平面图的最大流

待整理

```

1  #include <iostream>
2  #include <algorithm>
3  #include <cstdio>
4  #include <cstring>
5  #include <vector>
6  #include <cmath>
7  #include <map>
8  #include <queue>
9  using namespace std;
10
11 const int maxn = 100100;
12 const int inf = 0x3f3f3f3f;
13 struct Point
14 {
15     int x,y,id;
16     double theta;
17     Point() {}
18     Point(int _x,int _y)
19     {
20         x = _x;
21         y = _y;
22     }

```

```

23 Point(Point _s,Point _e,int _id)
24 {
25     id = _id;
26     x = _s.x-_e.x;
27     y = _s.y-_e.y;
28     theta = atan2(y,x);
29 }
30 bool operator < (const Point &b)const
31 {
32     return theta < b.theta;
33 }
34 };
35
36 map<pair<int,int>,int > idmap;
37 struct Edge
38 {
39     int from,to,next,cap,near,mark;
40 };
41 Edge edge[maxn*2];
42 int head[maxn],L;
43 int cntd[maxn];
44 void addedge(int u,int v,int cap)
45 {
46     cntd[u]++;
47     cntd[v]++;
48     idmap[make_pair(u,v)] = L;
49     edge[L].from = u;
50     edge[L].to = v;
51     edge[L].cap = cap;
52     edge[L].next = head[u];
53     edge[L].mark = -1;
54     head[u] = L++;
55 }
56
57 int rtp[maxn];
58 Point p[maxn],tp[maxn];
59 int n,m,S,T;
60 int vid;
61
62 struct Edge2
63 {
64     int to,next,dis;
65 } edge2[maxn*2];
66 int head2[maxn],L2;
67
68 void addedge2(int u,int v,int dis)
69 {
70     edge2[L2].to = v;
71     edge2[L2].dis = dis;
72     edge2[L2].next = head2[u];
73     head2[u] = L2++;

```

```

74 }
75
76 int dist[maxn];
77 bool inq[maxn];
78 int SPFA(int s,int t)
79 {
80     queue<int> Q;
81     memset(inq,false,sizeof(inq));
82     memset(dist,63,sizeof(dist));
83     Q.push(s);
84     dist[s] = 0;
85     while (!Q.empty())
86     {
87         int now = Q.front();
88         Q.pop();
89         for (int i = head2[now]; i != -1; i = edge2[i].next)
90             if (dist[edge2[i].to] > dist[now]+edge2[i].dis)
91             {
92                 dist[edge2[i].to] = dist[now]+edge2[i].dis;
93                 if (inq[edge2[i].to] == false)
94                 {
95                     inq[edge2[i].to] = true;
96                     Q.push(edge2[i].to);
97                 }
98             }
99         inq[now] = false;
100     }
101     return dist[t];
102 }
103
104 int main()
105 {
106     int totcas;
107     scanf("%d",&totcas);
108     for (int cas = 1; cas <= totcas; cas++)
109     {
110         idmap.clear();
111         L = 0;
112         scanf("%d%d",&n,&m);
113         S = T = 0;
114         for (int i = 0; i < n; i++)
115         {
116             head[i] = -1;
117             scanf("%d%d",&p[i].x,&p[i].y);
118             if (p[S].x > p[i].x)
119                 S = i;
120             if (p[T].x < p[i].x)
121                 T = i;
122             cntd[i] = 0;
123         }
124         //源汇中间加入一个特殊节点

```

```

125     head[n] = -1;
126     n ++;
127     addedge(S,n-1,inf);
128     addedge(n-1,S,inf);
129     addedge(T,n-1,inf);
130     addedge(n-1,T,inf);
131
132     for (int i = 0; i < m; i++)
133     {
134         int u,v,cap;
135         scanf("%d%d%d",&u,&v,&cap);
136         u--;
137         v--;
138         addedge(u,v,cap);
139         addedge(v,u,cap);
140     }
141
142     for (int i = 0; i < n; i++)
143     {
144         int tot = 0;
145         //源点汇点连到特殊点的方向需要特别考虑一下
146         if (i == S)
147             tp[tot++] = Point(Point(0,0),Point(-1,0),n-1);
148         else if (i == T)
149             tp[tot++] = Point(Point(0,0),Point(1,0),n-1);
150         else if (i == n-1)
151         {
152             tp[tot++] = Point(Point(0,0),Point(1,0),S);
153             tp[tot++] = Point(Point(0,0),Point(-1,0),T);
154         }
155         if (i < n-1)
156         {
157             for (int j = head[i]; j != -1; j = edge[j].next)
158             {
159                 if (i == S && edge[j].to == n-1) continue;
160                 if (i == T && edge[j].to == n-1) continue;
161                 tp[tot++] = Point(p[i],p[edge[j].to],edge[j].to);
162             }
163         }
164         sort(tp,tp+tot);
165         for (int j = 0; j < tot; j++)
166             rtp[tp[j].id] = j;
167         for (int j = head[i]; j != -1; j = edge[j].next)
168             edge[j].near = tp[(rtp[edge[j].to]+1)%tot].id;
169     }
170
171     vid = 0;
172     for (int i = 0; i < L; i++)
173         if (edge[i].mark == -1)
174         {
175             int now = edge[i].from;

```

```

176     int eid = i;
177     int to = edge[i].to;
178     while (true)
179     {
180         edge[eid].mark = vid;
181         eid ^= 1;
182         now = to;
183         to = edge[eid].near;
184         eid = idmap[make_pair(now,to)];
185
186         if (now == edge[i].from) break;
187     }
188     vid++;
189 }
190
191 L2 = 0;
192 for (int i = 0; i < vid; i++)
193     head2[i] = -1;
194 for (int i = 0; i < L; i++)
195     addedge2(edge[i].mark,edge[i^1].mark,edge[i].cap);
196     printf("%d\n",SPFA(edge[0].mark,edge[1].mark));
197 }
198 return 0;
199 }

```

5.10 强联通

```

1 int dfsnum[2000];
2 int low[2000];
3 int stack[2000];
4 int top;
5 int ans;
6 int an;
7 int be[2000];
8 int flag[2000];
9 void dfs(int x)
10 {
11     dfsnum[x] = low[x] = ans++;
12     stack[++top] = x;
13     flag[x] = 1;
14     for (int i = head[x]; i != -1; i = edge[i].next)
15     {
16         int y = edge[i].to;
17         if (dfsnum[y] == -1)
18         {
19             dfs(y);
20             low[x] = min(low[x],low[y]);
21         }
22         else if (flag[y] == 1)
23             low[x] = min(low[x],dfsnum[y]);
24     }

```



```

25  if (dfsnum[x] == low[x])
26  {
27      while (stack[top] != x)
28      {
29          flag[stack[top]] = 0;
30          be[stack[top]] = an;
31          top--;
32      }
33      flag[x] = 0;
34      be[x] = an++;
35      top--;
36  }
37  }

```

调用:

```

1  void SC()
2  {
3      memset(dfsnum,-1,sizeof(dfsnum));
4      memset(flag,0,sizeof(flag));
5      top = 0;
6      an = 0;
7      ans = 0;
8      for (int i = 0; i < n; i++)
9          if (dfsnum[i] == -1)
10             dfs(i);
11  }

```

5.11 最大团以及相关知识

独立集: 独立集是指图的顶点集的一个子集, 该子集的导出子图不含边. 如果一个独立集不是任何一个独立集的子集, 那么称这个独立集是一个极大独立集. 一个图中包含顶点数目最多的独立集称为最大独立集. 最大独立集一定是极大独立集, 但是极大独立集不一定是最大的独立集。

支配集: 与独立集相对应的就是支配集, 支配集也是图顶点集的一个子集, 设 S 是图 G 的一个支配集, 则对于图中的任意一个顶点 u , 要么属于集合 s , 要么与 s 中的顶点相邻. 在 s 中除去任何元素后 s 不再是支配集, 则支配集 s 是极小支配集. 称 G 的所有支配集中顶点个数最少的支配集为最小支配集, 最小支配集中的顶点个数成为支配数。

最小点的覆盖: 最小点的覆盖也是图的顶点集的一个子集, 如果我们选中一个点, 则称这个点将以他为端点的所有边都覆盖了. 将图中所有的边都覆盖所用顶点数最少, 这个集合就是最小的点的覆盖。

最大团: 图 G 的顶点的子集, 设 D 是最大团, 则 D 中任意两点相邻. 若 u, v 是最大团, 则 u, v 有边相连, 其补图 u, v 没有边相连, 所以图 G 的最大团 = 其补图的最大独立集. 给定无向图 $G = (V, E)$, 如果 U 属于 V , 并且对于任意 u, v 包含于 U 有 $\langle u, v \rangle$ 包含于 E , 则称 U 是 G 的完全子图, G 的完全子图 U 是 G 的团, 当且仅当 U 不包含在 G 的更大的完全子图中, G 的最大团是指 G 中所含顶点数目最多的团. 如果 U 属于 V , 并且对于任意 u, v 包含于 U 有 $\langle u, v \rangle$ 不包含于 E , 则称 U 是 G 的空子图, G 的空子图 U 是 G 的独立集, 当且仅当 U 不包含在 G 的更大的独立集, G 的最大团是指 G 中所含顶点数目最多的独立集。

一些性质：最大独立集 + 最小覆盖集 = V ，最大团 = 补图的最大独立集，最小覆盖集 = 最大匹配

```

1  #include <cstdio>
2  bool am[100][100];
3  int ans;
4  int c[100];
5  int U[100][100];
6  int n;
7  bool dfs(int rest,int num)
8  {
9      if (!rest)
10     {
11         if (num>=ans)
12             return 1;
13         else
14             return 0;
15     }
16     int pre=-1;
17     for (int i=0;i<rest && rest-i+num>=ans;i++)
18     {
19         int idx=U[num][i];
20         if (num+c[idx]<ans)
21             return 0;
22         int nrest=0;
23         for (int j=i+1; j<rest; j++)
24             if (am[idx][U[num][j]])
25                 U[num+1][nrest++]=U[num][j];
26         if (dfs(nrest,num+1))
27             return 1;
28     }
29     return 0;
30 }
31 int main()
32 {
33     while (scanf("%d",&n),n)
34     {
35         for (int i=0;i<n;i++)
36             for (int j=0;j<n;j++)
37                 scanf("%d",&am[i][j]);
38         ans=0;
39         for (int i=n-1; i>=0; i--)
40         {
41             int rest=0;
42             for (int j=i+1; j<n; j++)
43                 if (am[i][j])
44                     U[0][rest++]=j;
45             ans+=dfs(rest,0);
46             c[i]=ans;
47         }
48         printf("%d\n",ans);

```

```

49     }
50     return 0;
51 }

```

5.12 双连通分量

标号从 0 起

```

1  #include<cstdio>
2  #include<cstring>
3  #include<stack>
4  #include<queue>
5  #include<algorithm>
6  using namespace std;
7  const int MAXN=100000*2;
8  const int MAXM=200000;
9  struct edges
10 {
11     int to,next;
12     bool cut,visit;
13 } edge[MAXN<<1];
14 int head[MAXN],low[MAXN],dpt[MAXN],L;
15 bool visit[MAXN],cut[MAXN];
16 void init(int n)
17 {
18     L=0;
19     memset(head,-1,4*n);
20     memset(visit,0,n);
21 }
22 void add_edge(int u,int v)
23 {
24     edge[L].cut=edge[L].visit=0;
25     edge[L].to=v;
26     edge[L].next=head[u];
27     head[u]=L++;
28 }
29 int idx;
30 stack<int> st;
31 int bcc[MAXM];
32 void dfs(int u,int fu,int deg)
33 {
34     cut[u]=0;
35     visit[u]=1;
36     low[u]=dpt[u]=deg;
37     int tot=0;
38     for (int i=head[u]; i!=-1; i=edge[i].next)
39     {
40         int v=edge[i].to;
41         if (edge[i].visit)
42             continue;
43         st.push(i/2);
44         edge[i].visit=edge[i^1].visit=1;

```

```

45     if (visit[v])
46     {
47         low[u]=dpt[v]>low[u]?low[u]:dpt[v];
48         continue;
49     }
50     dfs(v,u,deg+1);
51     edge[i].cut=edge[i^1].cut=(low[v]>dpt[u] || edge[i].cut);
52     if (u!=fu) cut[u]=low[v]>=dpt[u]?1:cut[u];
53     if (low[v]>=dpt[u] || u==fu)
54     {
55         while (st.top()!=i/2)
56         {
57             int x=st.top()*2,y=st.top()*2+1;
58             bcc[st.top()]=idx;
59             st.pop();
60         }
61         bcc[i/2]=idx++;
62         st.pop();
63     }
64     low[u]=low[v]>low[u]?low[u]:low[v];
65     tot++;
66 }
67 if (u==fu && tot>1) cut[u]=1;
68 }
69 int main()
70 {
71     int n,m;
72     while (scanf("%d%d",&n,&m)!=EOF)
73     {
74         init(n);
75         for (int i=0; i<m; i++)
76         {
77             int u,v;
78             scanf("%d%d",&u,&v);
79             add_edge(u,v);
80             add_edge(v,u);
81         }
82         idx=0;
83         for (int i=0; i<n; i++)
84             if (!visit[i])
85                 dfs(i,i,0);
86     }
87     return 0;
88 }

```

5.13 割点与桥

```

1  #include<cstdio>
2  #include<cstring>
3  const int MAXN=10000;
4  struct edges

```

```

5 {
6     int to,next;
7     bool cut,visit;
8     int from;
9 } edge[MAXN-1<<1];
10 int head[MAXN],low[MAXN],dfn[MAXN],L;
11 bool visit[MAXN],cut[MAXN];
12 void init(int n)
13 {
14     L=0;
15     memset(head,-1,4*n);
16     memset(cut,0,4*n);
17     memset(visit,0,4*n);
18 }
19 void add_edge(int u,int v)
20 {
21     edge[L].from=u;
22     edge[L].cut=edge[L].visit=0;
23     edge[L].to=v;
24     edge[L].next=head[u];
25     head[u]=L++;
26 }
27 int idx;
28 void dfs(int u,int fu)
29 {
30     visit[u]=1;
31     low[u]=dfn[u]=idx++;
32     int tot=0;
33     for (int i=head[u]; i!=-1; i=edge[i].next)
34     {
35         int v=edge[i].to;
36         if (edge[i].visit)
37             continue;
38         edge[i].visit=edge[i^1].visit=1;
39         if (visit[v])
40         {
41             low[u]=dfn[v]>low[u]?low[u]:dfn[v];
42             continue;
43         }
44         dfs(v,u);
45         edge[i].cut=edge[i^1].cut=low[v]>dfn[u] || edge[i].cut;
46         if (u!=fu) cut[u]=low[v]>=dfn[u]?1:cut[u];
47         low[u]=low[v]>low[u]?low[u]:low[v];
48         tot++;
49     }
50     if (u==fu && tot>1) cut[u]=1;
51 }
52 int main()
53 {
54     int t;
55     scanf("%d",&t);

```

```

56 while (t--)
57 {
58     int n,m;
59     scanf("%d%d",&n,&m);
60     init(n);
61     for (int i=0; i<m; i++)
62     {
63         int u,v;
64         scanf("%d%d",&u,&v);
65         add_edge(--u,--v);
66         add_edge(v,u);
67     }
68     for (int i=0; i<n; i++)
69         if (!visit[i])
70         {
71             idx=0;
72             dfs(i,i);
73         }
74     }
75     return 0;
76 }

```

5.14 LCA

在线 LCA, bfs

```

1  #include<cstdio>
2  #include<cstring>
3  #include<queue>
4  using namespace std;
5  const int NSIZE = 50000;
6  const int DEG = 20;
7  struct trees
8  {
9
10     int fa[DEG];
11     int head,deg;
12 } tree[NSIZE];
13 struct edges
14 {
15     int to , next;
16 } edge[NSIZE];
17 struct states
18 {
19     int u,fu,deg;
20 };
21 int L;
22 void add_edge(int x, int y)
23 {
24     edge[L].to = y;
25     edge[L].next = tree[x].head;
26     tree[x].head = L++;
27 }

```

```

28 int Root;
29 void BFS(int s)
30 {
31     queue<states> que;
32     states st;
33     st.deg=0;
34     st.fu=st.u=s;
35     que.push(st);
36     while(!que.empty())
37     {
38         states st=que.front();
39         que.pop();
40         tree[st.u].deg = st.deg;
41         tree[st.u].fa[0] = st.fu;
42         for (int i=1;i<DEG;i++)
43             tree[st.u].fa[i]=s;
44         for (int tmp=st.fu,num=1;tree[tmp].deg;tmp=tree[st.u].fa[num
            ++])
45             tree[st.u].fa[num]=tree[tmp].fa[num-1];
46         for(int i = tree[st.u].head ; i != -1; i = edge[i].next)
47         {
48             int v = edge[i].to;
49             if (v == st.fu) continue;
50             states nst;
51             nst.u=v;
52             nst.fu=st.u;
53             nst.deg=st.deg+1;
54             que.push(nst);
55         }
56     }
57 }
58 int LCA(int x, int y)
59 {
60     if(tree[x].deg > tree[y].deg) swap(x,y);
61     int hx=tree[x].deg,hy=tree[y].deg;
62     int tx=x,ty=y;
63     for (int det=hy-hx,i=0; det; det>>=1,i++)
64         if (det&1)
65             ty=tree[ty].fa[i];
66     if(tx == ty) return tx;
67     for (int i=DEG-1; i>=0; i--)
68     {
69         if(tree[tx].fa[i] == tree[ty].fa[i])
70             continue;
71         tx = tree[tx].fa[i];
72         ty = tree[ty].fa[i];
73     }
74     return tree[tx].fa[0];
75 }
76 int main()
77 {

```

```

78  int t;
79  scanf("%d",&t);
80  while(t--)
81  {
82      int n;
83      scanf("%d",&n);
84      L = 0;
85      for(int i = 0 ; i < n ; i++)
86          tree[i].head = -1;
87      for(int i = 0 ; i < n-1 ; i++)
88      {
89          int a,b;
90          scanf("%d%d",&a ,&b);
91          add_edge(a-1,b-1);
92          add_edge(b-1,a-1);
93      }
94      Root=0;
95      BFS(Root);
96      int a,b;
97      scanf("%d%d",&a,&b);
98      int lca=LCA(a-1,b-1)+1;
99      printf("%d\n",lca);
100 }
101 return 0;
102 }

```

5.15 最优比例生成树

```

1  #include<stdio.h>
2  #include<string.h>
3  #include<math.h>
4  struct
5  {
6      int x,y;
7      double z;
8  } node[1100];
9  struct
10 {
11     double l,c;
12 } map[1100][1100];
13 int n,l,f[1100],pre[1100];
14 double dis[1100];
15 double mst(double x)
16 {
17     int i,j,tmp;
18     double min,s=0,t=0;
19     memset(f,0,sizeof(f));
20     f[1]=1;
21     for (i=2; i<=n; i++)
22     {
23         dis[i]=map[1][i].c-map[1][i].l*x;

```



```

24     pre[i]=1;
25 }
26 for (i=1; i<n; i++)
27 {
28     min=1e10;
29     for (j=1; j<=n; j++)
30         if (!f[j] && min>dis[j])
31         {
32             min=dis[j];
33             tmp=j;
34         }
35     f[tmp]=1;
36     t+=map[pre[tmp]][tmp].l;
37     s+=map[pre[tmp]][tmp].c;
38     for (j=1; j<=n; j++)
39         if (!f[j] && map[tmp][j].c-map[tmp][j].l*x<dis[j])
40         {
41             dis[j]=map[tmp][j].c-map[tmp][j].l*x;
42             pre[j]=tmp;
43         }
44 }
45 return s/t;
46 }
47 int main()
48 {
49     int i,j;
50     double a,b;
51     scanf("%d",&n);
52     while (n)
53     {
54         for (i=1; i<=n; i++)
55             scanf("%d%d%lf",&node[i].x,&node[i].y,&node[i].z);
56         for (i=1; i<=n; i++)
57             for (j=i+1; j<=n; j++)
58             {
59                 map[j][i].l=map[i][j].l=sqrt(1.0*(node[i].x-node[j].x)*(
                    node[i].x-node[j].x)+(node[i].y-node[j].y)*(node[i].y-
                    node[j].y));
60                 map[j][i].c=map[i][j].c=fabs(node[i].z-node[j].z);
61             }
62         a=0,b=mst(a);
63         while (fabs(b-a)>1e-8)
64         {
65             a=b;
66             b=mst(a);
67         }
68         printf("%.3f\n",b);
69         scanf("%d",&n);
70     }
71 }

```

5.16 生成树计数

根据邻接矩阵构造 Laplacian matrix。

```

1 Matrix laplacian;
2 laplacian.clear();
3 for (int i = 0; i < n; i++)
4     for (int j = 0; j < n; j++)
5         if (i != j && G[i][j])
6         {
7             laplacian.a[i][j] = -1;
8             laplacian.a[i][i]++;
9         }
10 printf("%d\n", laplacian.det(n-1));

```

5.17 全局最小割

```

1 #include <iostream>
2 using namespace std;
3 const int maxn=510;
4 int map[maxn][maxn];
5 int n;
6 void contract(int x,int y)
7 {
8     int i,j;
9     for (i=0; i<n; i++)
10         if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];
11     for (i=y+1; i<n; i++) for (j=0; j<n; j++)
12     {
13         map[i-1][j]=map[i][j];
14         map[j][i-1]=map[j][i];
15     }
16     n--;
17 }
18 int w[maxn],c[maxn];
19 int sx,tx;
20 int mincut()
21 {
22     int i,j,k,t;
23     memset(c,0,sizeof(c));
24     c[0]=1;
25     for (i=0; i<n; i++) w[i]=map[0][i];
26     for (i=1; i+1<n; i++)
27     {
28         t=k=-1;
29         for (j=0; j<n; j++) if (c[j]==0&&w[j]>k)
30             k=w[t=j];
31         c[sx=t]=1;
32         for (j=0; j<n; j++) w[j]+=map[t][j];
33     }
34     for (i=0; i<n; i++) if (c[i]==0) return w[tx=i];
35 }

```

```

36 int main()
37 {
38     int i,j,k,m;
39     while (scanf("%d%d",&n,&m)!=EOF)
40     {
41         memset(map,0,sizeof(map));
42         while (m——)
43         {
44             scanf("%d%d%d",&i,&j,&k);
45             map[i][j]+=k;
46             map[j][i]+=k;
47         }
48         int mint=999999999;
49         while (n>1)
50         {
51             k=mincut();
52             if (k<mint) mint=k;
53             contract(sx,tx);
54         }
55         printf("%d\n",mint);
56     }
57     return 0;
58 }

```

5.18 欧拉路

5.18.1 有向图

```

1 void solve(int x)
2 {
3     int i;
4     if (!match[x])
5     {
6         path[++l]=x;
7         return ;
8     }
9     for (i=1; i<=n; i++)
10         if (b[x][i])
11         {
12             b[x][i]--;
13             match[x]--;
14             solve(i);
15         }
16     path[++l]=x;
17 }

```

5.18.2 无向图

```

1 void solve(int x)
2 {
3     int i;
4     if (!match[x])
5     {

```

```

6     path[++l]=x;
7     return ;
8 }
9 for (i=1; i<=n; i++)
10     if (b[x][i])
11     {
12         b[x][i]--;
13         b[i][x]--;
14         match[x]--;
15         match[i]--;
16         solve(i);
17     }
18     path[++l]=x;
19 }

```

5.18.3 混合图

zju1992

```

1  int in[MAXN+100],out[MAXN+100];
2  int main()
3  {
4      int t;
5      scanf("%d",&t);
6      while (t--)
7      {
8          int n,m;
9          scanf("%d%d",&n,&m);
10         N=n+2;L=-1;
11         for (int i=0;i<N;i++)
12             head[i]=-1;
13         memset(in,0,sizeof(in));
14         memset(out,0,sizeof(out));
15
16         for (int i=0;i<m;i++)
17         {
18             int x,y,z;
19             scanf("%d%d%d",&x,&y,&z);
20             in[y]++;out[x]++;
21             if (!z)
22                 add_edge(x,y,1);
23         }
24         int flag=1;
25         for (int i=1;i<=n;i++)
26         {
27             if (in[i]-out[i]>0)
28                 add_edge(i,n+1,(in[i]-out[i])/2);
29             else
30                 if (out[i]-in[i]>0)
31                     add_edge(0,i,(out[i]-in[i])/2);
32             //printf("%d %d %d\n",i,out[i],in[i]);
33             if ((in[i]+out[i])&1)
34             {

```

```

35         flag=0;
36         break;
37     }
38 }
39 maxflow(0,n+1);
40 for (int i=head[0];i!=-1;i=edge[i].next)
41     if (edge[i].cap>0 && edge[i].cap>edge[i].flow)
42     {
43         flag=0;
44         break;
45     }
46 if (flag)
47     puts("possible");
48 else
49     puts("impossible");
50 }
51 return 0;
52 }

```

5.19 K 短路

```

1  #include<cstdio>
2  #include<cstring>
3  #include<queue>
4  using namespace std;
5  int K;
6  class states
7  {
8  public:
9      int cost,id;
10 };
11 int dist[1000];
12 class cmp
13 {
14 public:
15     bool operator()(const states &i,const states &j)
16     {
17         return i.cost>j.cost;
18     }
19 };
20 class cmp2
21 {
22 public:
23     bool operator()(const states &i,const states &j)
24     {
25         return i.cost+dist[i.id]>j.cost+dist[j.id];
26     }
27 };
28 struct edges
29 {
30     int to,next,cost;

```

```

31 } edger[100000],edge[100000];
32 int headr[1000],head[1000],Lr,L;
33 void dijkstra(int s)
34 {
35     states u;
36     u.id=s;
37     u.cost=0;
38     dist[s]=0;
39     priority_queue<states,vector<states>,cmp> q;
40     q.push(u);
41     while (!q.empty())
42     {
43         u=q.top();
44         q.pop();
45         if (u.cost!=dist[u.id]) continue;
46         for (int i=headr[u.id]; i!=-1; i=edger[i].next)
47         {
48             states v=u;
49             v.id=edger[i].to;
50             if (dist[v.id]>dist[u.id]+edger[i].cost)
51             {
52                 v.cost=dist[v.id]=dist[u.id]+edger[i].cost;
53                 q.push(v);
54             }
55         }
56     }
57 }
58 int num[1000];
59 void init(int n)
60 {
61     Lr=L=0;
62     memset(head,-1,4*n);
63     memset(headr,-1,4*n);
64     memset(dist,63,4*n);
65     memset(num,0,4*n);
66 }
67 void add_edge(int u,int v,int x)
68 {
69     edge[L].to=v;
70     edge[L].cost=x;
71     edge[L].next=head[u];
72     head[u]=L++;
73     edger[Lr].to=u;
74     edger[Lr].cost=x;
75     edger[Lr].next=headr[v];
76     headr[v]=Lr++;
77 }
78 int a_star(int s,int t)
79 {
80     if (dist[s]==0x3f3f3f3f)
81         return -1;

```

```

82     priority_queue<states,vector<states>,cmp2> q;
83     states tmp;
84     tmp.id=s;
85     tmp.cost=0;
86     q.push(tmp);
87     while (!q.empty())
88     {
89         states u=q.top();
90         q.pop();
91         num[u.id]++;
92         if (num[t]==K)
93             return u.cost;
94         for (int i=head[u.id]; i!=-1; i=edge[i].next)
95         {
96             int v=edge[i].to;
97             tmp.id=v;
98             tmp.cost=u.cost+edge[i].cost;
99             q.push(tmp);
100         }
101     }
102     return -1;
103 }
104 int main()
105 {
106     int n,m;
107     scanf("%d%d",&n,&m);
108     init(n);
109     for (int i=0; i<m; i++)
110     {
111         int u,v,x;
112         scanf("%d%d%d",&u,&v,&x);
113         add_edge(u-1,v-1,x);
114     }
115     int s,t;
116     scanf("%d%d%d",&s,&t,&K);
117     if (s==t)
118         K++;
119     dijkstra(t-1);
120     printf("%d\n",a_star(s-1,t-1));
121 }

```

5.20 稳定婚姻

假定有 n 个男生和 n 个女生，理想的拍拖状态就是对于每对情侣 (a,b) ，找不到另一对情侣 (c,d) 使得 c 更喜欢 b ， b 也更喜欢 c ，同理，对 a 来说也没有 (e,f) 使得 a 更喜欢 e 而 e 更喜欢 a ，当然最后会有一些人落单。这样子一个状态可以称为理想拍拖状态，它也有一个专业的名词叫稳定婚姻。

求解这个问题可以用一个专有的算法，延迟认可算法，其核心就是让每个男生按自己喜欢的顺序逐个向女生表白，例如 leokan 向一个女生求爱，这个过程中，若这个女生没有男朋友，那么这个女生就暂时成为 leokan 的女朋友，或这个女生喜欢她现有男朋友的程度没有喜欢 leokan 高，这个女生也暂时成为 leokan 的女朋友，而她原有的男朋友则再找下一个次喜欢的女生来当女朋友。

```
1 #include<string.h>
2 #include<stdio.h>
3 #define N 1050
4 int boy[N][N];
5 int girl[N][N];
6 int ans[N];
7 int cur[N];
8 int n;
9 void getMarry(int g)
10 {
11     for (int i=ans[g]+1;i<n;i++)
12     {
13         int b=girl[g][i]-1;
14         if (cur[b]<0)
15         {
16             ans[g]=i;
17             cur[b]=g;
18             return;
19         }
20         int og=cur[b];
21         if (boy[b][og] > boy[b][g])
22         {
23             cur[b]=g;
24             ans[g]=i;
25             getMarry(og);
26             return;
27         }
28     }
29 };
30 int main()
31 {
32     int t,a;
33     scanf("%d",&t);
34     while(t--)
35     {
36         memset(girl,0,sizeof(girl));
37         memset(boy,0,sizeof(boy));
38         scanf("%d",&n);
39         for (int i=0;i<n;i++)
40             for (int j=0;j<n;j++)
41                 scanf("%d",&girl[i][j]);
42         for (int i=0;i<n;i++)
43             for (int j=0;j<n;j++)
44             {
45                 scanf("%d",&a);
46                 boy[i][a-1]=j;
47             }
48         memset(cur,0xff,sizeof(cur));
49         memset(ans,0xff,sizeof(ans));
50         for (int i=0;i<n;i++)
51             getMarry(i);
```



```

52     for (int i=0;i<n;i++)
53         printf("%d\n",girl[i][ans[i]]);
54     }
55     return 0;
56 }

```

5.21 最小树形图

```

1  const int inf = 19921005;
2  int n,m,u,v,cost,dis[1001][1001],L;
3
4  void init(int n)
5  {
6      L = 0;
7      for (int i = 0; i < n; i++)
8          for (int j = 0; j < n; j++)
9              dis[i][j] = inf;
10 }
11
12 struct Edge
13 {
14     int u,v,cost;
15 };
16
17 Edge e[1001*1001];
18
19 int pre[1001],id[1001],visit[1001],in[1001];
20
21 int zhuliu(int root,int n,int m,Edge e[])
22 {
23     int res = 0,u,v;
24     while (true)
25     {
26         for (int i = 0; i < n; i++)
27             in[i] = inf;
28         for (int i = 0; i < m; i++)
29             if (e[i].u != e[i].v && e[i].cost < in[e[i].v])
30             {
31                 pre[e[i].v] = e[i].u;
32                 in[e[i].v] = e[i].cost;
33             }
34         for (int i = 0; i < n; i++)
35             if (i != root)
36                 if (in[i] == inf) return -1;
37         int tn = 0;
38         memset(id,-1,sizeof(id));
39         memset(visit,-1,sizeof(visit));
40         in[root] = 0;
41         for (int i = 0; i < n; i++)
42         {
43             res += in[i];

```

```

44     v = i;
45     while (visit[v] != i && id[v] == -1 && v != root)
46     {
47         visit[v] = i;
48         v = pre[v];
49     }
50     if(v != root && id[v] == -1)
51     {
52         for(int u = pre[v] ; u != v ; u = pre[u])
53             id[u] = tn;
54         id[v] = tn++;
55     }
56 }
57 if(tn == 0) break;
58 for (int i = 0; i < n; i++)
59     if (id[i] == -1)
60         id[i] = tn++;
61 for (int i = 0; i < m; i++)
62 {
63     int v = e[i].v;
64     e[i].u = id[e[i].u];
65     e[i].v = id[e[i].v];
66     if (e[i].u != e[i].v)
67         e[i++].cost -= in[v];
68     else
69         swap(e[i], e[--m]);
70 }
71 n = tn;
72 root = id[root];
73 }
74 return res;
75 }
76
77 int main()
78 {
79     freopen("in.txt", "r", stdin);
80     while (scanf("%d%d", &n, &m) != EOF)
81     {
82         init(n);
83         for (int i = 0; i < m; i++)
84         {
85             scanf("%d%d%d", &u, &v, &cost);
86             if (u == v) continue;
87             dis[u][v] = min(dis[u][v], cost);
88         }
89         L = 0;
90         for (int i = 0; i < n; i++)
91             for (int j = 0; j < n; j++)
92                 if (dis[i][j] != inf)
93                 {
94                     e[L].u = i;

```

```
95         e[L].v = j;
96         e[L++].cost = dis[i][j];
97     }
98     printf("%d\n",zhuliu(0,n,L,e));
99 }
100 return 0;
101 }
```

6 计算几何

6.1 注意事项

如果用整数小心越界（多次乘法？）
 如果用浮点数判断的时候一定要用 `eps`！

6.2 基本函数

6.2.1 Point 定义

```

1 struct Point
2 {
3     double x, y;
4     Point() {}
5     Point(double _x, double _y)
6     {
7         x = _x, y = _y;
8     }
9     Point operator -(const Point &b) const
10    {
11        return Point(x-b.x, y-b.y);
12    }
13    double operator *(const Point &b) const
14    {
15        return x*b.y-y*b.x;
16    }
17    double operator &(const Point &b) const
18    {
19        return x*b.x+y*b.y;
20    }
21    void transXY(double B)
22    {
23        double tx = x, ty = y;
24        x = tx*cos(B)-ty*sin(B);
25        y = tx*sin(B)+ty*cos(B);
26    }
27 };
  
```

6.2.2 Line 定义

```

1 struct Line
2 {
3     Point s, e;
4     double k;
5     Line() {}
6     Line(Point _s, Point _e)
7     {
8         s = _s, e = _e;
9         k = atan2(e.y-s.y, e.x-s.x);
10    }
  
```

```

11 Point operator &(const Line &b) const
12 {
13     Point res = s;
14     //注意：有些题目可能会有直线相交或者重合情况
15     //可以把返回值改成 pair<Point,int> 来返回两直线的状态。
16     double t = ((s-b.s)*(b.s-b.e))/((s-e)*(b.s-b.e));
17     res.x += (e.x-s.x)*t;
18     res.y += (e.y-s.y)*t;
19     return res;
20 }
21 };

```

6.2.3 距离：点到直线距离

result: 点到直线最近点

```

1 Point NPT(Point P, Line L)
2 {
3     Point result;
4     double a, b, t;
5
6     a = L.e.x-L.s.x;
7     b = L.e.y-L.s.y;
8     t = ((P.x-L.s.x)*a+(P.y-L.s.y)*b)/(a*a+b*b);
9
10    result.x = L.s.x+a*t;
11    result.y = L.s.y+b*t;
12    return dist(P, result);
13 }

```

6.2.4 距离：点到线段距离

res: 点到线段最近点

```

1 Point NearestPointToLineSeg(Point P, Line L)
2 {
3     Point result;
4     double a, b, t;
5
6     a = L.e.x-L.s.x;
7     b = L.e.y-L.s.y;
8     t = ((P.x-L.s.x)*a+(P.y-L.s.y)*b)/(a*a+b*b);
9
10    if (t >= 0 && t <= 1)
11    {
12        result.x = L.s.x+a*t;
13        result.y = L.s.y+b*t;
14    }
15    else
16    {
17        if (dist(P,L.s) < dist(P,L.e))
18            result = L.s;
19        else
20            result = L.e;

```

```

21 | }
22 | return result;
23 | }

```

旧版

```

1 | double CalcDis(Point a,Point s,Point e) //点到线段距离
2 | {
3 |     if (sgn((e-s)*(a-s)) < 0 || sgn((s-e)*(a-e)) < 0)
4 |         return min(dist(a,s),dist(a,e));
5 |     return abs(((s-a)*(e-a))/dist(s-e));
6 | }

```

6.2.5 面积：多边形

点按逆时针排序。

```

1 | double CalcArea(Point p[], int n)
2 | {
3 |     double res = 0;
4 |     for (int i = 0; i < n; i++)
5 |         res += (p[i]*p[(i+1) % n])/2;
6 |     return res;
7 | }

```

6.2.6 判断：线段相交

```

1 | bool inter(Line l1,Line l2)
2 | {
3 |     return
4 |         max(l1.s.x,l1.e.x) >= min(l2.s.x,l2.e.x) &&
5 |         max(l2.s.x,l2.e.x) >= min(l1.s.x,l1.e.x) &&
6 |         max(l1.s.y,l1.e.y) >= min(l2.s.y,l2.e.y) &&
7 |         max(l2.s.y,l2.e.y) >= min(l1.s.y,l1.e.y) &&
8 |         sgn((l2.s-l1.s)*(l1.e-l1.s))*sgn((l2.e-l1.s)*(l1.e-l1.s)) <= 0 &&
9 |         sgn((l1.s-l2.s)*(l2.e-l2.s))*sgn((l1.e-l2.s)*(l2.e-l2.s)) <= 0;
10 | }

```

6.2.7 判断：点在线段上

```

1 | bool OnSeg(Line a,Point b)
2 | {
3 |     return ((a.s-b)*(a.e-b) == 0 &&
4 |         (b.x-a.s.x)*(b.x-a.e.x) <= 0 &&
5 |         (b.y-a.s.y)*(b.y-a.e.y) <= 0);
6 | }

```

6.2.8 判断：点在多边形内

凸包且按逆时针排序

```

1 | bool inPoly(Point a,Point p[],int n)
2 | {
3 |     for (int i = 0;i < n;i++)
4 |         if ((p[i]-a)*(p[(i+1)%n]-a) < 0)

```

```

5     return false;
6     return true;
7 }

```

射线法, 多边形可以是凸的或凹的

poly 的顶点数目要大于等于 3

返回值为:

0 - 点在 poly 内

1 - 点在 poly 边界上

2 - 点在 poly 外

```

1 int inPoly(Point p, Point poly[], int n)
2 {
3     int i, count;
4     Line ray, side;
5
6     count = 0;
7     ray.s = p;
8     ray.e.y = p.y;
9     ray.e.x = -1; //-INF, 注意取值防止越界!
10
11    for (i = 0; i < n; i++)
12    {
13        side.s = poly[i];
14        side.e = poly[(i+1)%n];
15
16        if(OnSeg(p, side))
17            return 1;
18
19        // 如果平行轴则不作考虑sidex
20        if (side.s.y == side.e.y)
21            continue;
22
23        if (OnSeg(side.s, ray))
24        {
25            if (side.s.y > side.e.y) count++;
26        }
27        else if (OnSeg(side.e, ray))
28        {
29            if (side.e.y > side.s.y) count++;
30        }
31        else if (inter(ray, side))
32        {
33            count++;
34        }
35    }
36    return ((count % 2 == 1) ? 0 : 2);
37 }

```

6.2.9 判断: 两凸包相交

需要考虑这几个: 一个凸包的点在另外一个图包内 (包括边界); 一个凸包的某条边与另一个凸包某条边相交; 如果凸包可能退化成点线还需要判断点在线段上和点和点重合。

6.2.10 排序：叉积极角排序

```

1 bool cmp(const Point& a,const Point& b)
2 {
3     if (a.y*b.y <= 0)
4     {
5         if (a.y > 0 || b.y > 0) return a.y < b.y;
6         if (a.y == 0 && b.y == 0) return a.x < b.x;
7     }
8     return a*b > 0;
9 }

```

6.3 三维几何

6.3.1 Point 定义

```

1 struct Point3D
2 {
3     double x,y,z;
4     Point3D() {}
5     Point3D(double _x,double _y,double _z)
6     {
7         x = _x;
8         y = _y;
9         z = _z;
10    }
11    Point3D operator -(const Point3D& b)const
12    {
13        return Point3D(x-b.x,y-b.y,z-b.z);
14    }
15    Point3D operator *(const Point3D& b)const
16    {
17        return Point3D(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
18    }
19    double operator &(const Point3D& b)const
20    {
21        return x*b.x+y*b.y+z*b.z;
22    }
23 };
24 //模
25 double Norm(Point3D p)
26 {
27     return sqrt(p&p);
28 }
29 //绕单位向量 V 旋转  $\theta$  角度
30 Point3D Trans(Point3D pa,Point3D V,double theta)
31 {
32     double s = sin(theta);
33     double c = cos(theta);
34     double x,y,z;
35     x = V.x;
36     y = V.y;

```



```

37     z = V.z;
38     Point3D pp =
39     Point3D(
40         (x*x*(1-c)+c)*pa.x+(x*y*(1-c)-z*s)*pa.y+(x*z*(1-c)+y*s)*pa.z,
41         (y*x*(1-c)+z*s)*pa.x+(y*y*(1-c)+c)*pa.y+(y*z*(1-c)-x*s)*pa.z,
42         (x*z*(1-c)-y*s)*pa.x+(y*z*(1-c)+x*s)*pa.y+(z*z*(1-c)+c)*pa.z);
43     return pp;
44 }

```

6.3.2 经度纬度转换

直角坐标系与极坐标系转换:

$$\begin{cases} x = r \times \sin\theta \times \cos\varphi \\ y = r \times \sin\theta \times \sin\varphi \\ z = r \times \cos\theta \end{cases} \quad \begin{cases} r = \sqrt{x^2 + y^2 + z^2} \\ \varphi = \arctan(\frac{y}{x}) \\ \theta = \arccos(\frac{z}{r}) \end{cases} \quad r \in [0, +\infty), \varphi \in [0, 2\pi], \theta \in [0, \pi]$$

经度维度转换 ($lat1 \in (-\frac{\pi}{2}, \frac{\pi}{2}), lng1 \in (-\pi, \pi)$)

```

1 Point3D getPoint3D(double lat, double lng, double r)
2 {
3     lat += pi/2;
4     lng += pi;
5     return
6     Point3D(r*sin(lat)*cos(lng), r*sin(lat)*sin(lng), r*cos(lat));
7 }

```

6.3.3 判断: 直线相交

```

1 bool LineIntersect(Line3D L1, Line3D L2)
2 {
3     Point3D s = L1.s-L1.e;
4     Point3D e = L2.s-L2.e;
5     Point3D p = s*e;
6     if (ZERO(p)) return false; //是否平行
7     p = (L2.s-L1.e)*(L1.s-L1.e);
8     return ZERO(p&L2.e); //是否共面
9 }

```

6.3.4 判断: 线段相交

需要先判断是否在一个平面上:

```

1 bool inter(Point a, Point b, Point c, Point d)
2 {
3     Point ret = (a-b)*(c-d);
4     Point t1 = (b-a)*(c-a);
5     Point t2 = (b-a)*(d-a);
6     Point t3 = (d-c)*(a-c);
7     Point t4 = (d-c)*(b-c);
8     return sgn(t1&ret)*sgn(t2&ret) < 0 &&
9         sgn(t3&ret)*sgn(t4&ret) < 0;
10 }

```

6.3.5 判断：三维向量是否为 0

```

1 inline bool ZERO(Point3D p)
2 {
3     return (ZERO(p.x) && ZERO(p.y) && ZERO(p.z));
4 }

```

6.3.6 判断：点在直线上

```

1 bool OnLine(Point3D p, Line3D L)
2 {
3     return ZERO((p-L.s)*(L.e-L.s));
4 }

```

6.3.7 判断：点在线段上

```

1 bool OnSeg(Point3D p, Line3D L)
2 {
3     return (ZERO((L.s-p)*(L.e-p)) &&
4             EQ(Norm(p-L.s)+Norm(p-L.e),Norm(L.e-L.s)));
5 }

```

6.3.8 距离：点到直线

```

1 double Distance(Point3D p, Line3D L)
2 {
3     return (Norm((p-L.s)*(L.e-L.s))/Norm(L.e-L.s));
4 }

```

6.3.9 夹角

返回值是 $[0, \pi]$ 之间的弧度

```

1 double Inclination(Line3D L1, Line3D L2)
2 {
3     Point3D u = L1.e - L1.s;
4     Point3D v = L2.e - L2.s;
5     return acos( (u & v) / (Norm(u)*Norm(v)) );
6 }

```

6.4 圆

6.4.1 面积：两圆相交

圆不可包含

```

1 double dis(int x,int y)
2 {
3     return sqrt((double)(x*x+y*y));
4 }
5 double area(int x1,int y1,int x2,int y2,double r1,double r2)
6 {
7     double s=dis(x2-x1,y2-y1);
8     if(r1+r2<s) return 0;
9     else if(r2-r1>s) return PI*r1*r1;
10    else if(r1-r2>s) return PI*r2*r2;

```

```

11  double q1=acos((r1*r1+s*s-r2*r2)/(2*r1*s));
12  double q2=acos((r2*r2+s*s-r1*r1)/(2*r2*s));
13  return (r1*r1*q1+r2*r2*q2-r1*s*sin(q1));
14  }

```

6.4.2 三角形外接圆

```

1  void CircumscribedCircle()
2  {
3      for (int i = 0; i < 3; i++)
4          scanf("%lf%lf",&p[i].x,&p[i].y);
5      tp = Point((p[0].x+p[1].x)/2,(p[0].y+p[1].y)/2);
6      l[0] = Line(tp,Point(tp.x-(p[1].y-p[0].y),tp.y+(p[1].x-p[0].x)));
7      tp = Point((p[0].x+p[2].x)/2,(p[0].y+p[2].y)/2);
8      l[1] = Line(tp,Point(tp.x-(p[2].y-p[0].y),tp.y+(p[2].x-p[0].x)));
9      tp = LineToLine(l[0],l[1]);
10     r = Point(tp,p[0]).Length();
11     printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
12 }

```

6.4.3 三角形内切圆

```

1  void InscribedCircle()
2  {
3      for (int i = 0; i < 3; i++)
4          scanf("%lf%lf",&p[i].x,&p[i].y);
5      if (xmult(Point(p[0],p[1]),Point(p[0],p[2])) < 0)
6          swap(p[1],p[2]);
7      for (int i = 0; i < 3; i++)
8          len[i] = Point(p[i],p[(i+1)%3]).Length();
9      tr = (len[0]+len[1]+len[2])/2;
10     r = sqrt((tr-len[0])*(tr-len[1])*(tr-len[2])/tr);
11     for (int i = 0; i < 2; i++)
12     {
13         v = Point(p[i],p[i+1]);
14         tv = Point(-v.y,v.x);
15         tr = tv.Length();
16         tv = Point(tv.x*r/tr,tv.y*r/tr);
17         tp = Point(p[i].x+tv.x,p[i].y+tv.y);
18         l[i].s = tp;
19         tp = Point(p[i+1].x+tv.x,p[i+1].y+tv.y);
20         l[i].e = tp;
21     }
22     tp = LineToLine(l[0],l[1]);
23     printf("(%.6f,%.6f,%.6f)\n",tp.x,tp.y,r);
24 }

```

6.4.4 点对圆的两个切点

```

1  void calc_qie(Point poi,Point o,double r,Point &result1,Point &
    result2)
2  {

```

```

3   double line = sqrt((poi.x-o.x)*(poi.x-o.x)+(poi.y-o.y)*(poi.y-o.y
   ));
4   double angle = acos(r/line);
5   Point unitvector,lin;
6   lin.x = poi.x-o.x;
7   lin.y = poi.y-o.y;
8   unitvector.x = lin.x/sqrt(lin.x*lin.x+lin.y*lin.y)*r;
9   unitvector.y = lin.y/sqrt(lin.x*lin.x+lin.y*lin.y)*r;
10  result1 = unitvector.Rotate(-angle);
11  result2 = unitvector.Rotate(angle);
12  result1.x += o.x;
13  result1.y += o.y;
14  result2.x += o.x;
15  result2.y += o.y;
16 }

```

6.4.5 两圆公切点

```

1  void Gao()
2  {
3      tn = 0;
4      Point a,b,vab;
5      double tab,tt,dis,theta;
6      for (int i = 0; i < tc; i++)
7          for (int j = 0; j < tc; j++)
8              if (i != j)
9                  {
10                     a = c[i];
11                     b = c[j];
12                     vab = Point(a,b);
13                     tab = atan2(vab.y,vab.x);
14                     dis = sqrt(vab.x*vab.x+vab.y*vab.y);
15                     if (b.r > a.r)
16                         tt = asin((b.r-a.r)/dis);
17                     else
18                         tt = -asin((a.r-b.r)/dis);
19                     theta = tab+pi/2+tt;
20                     tp[tn++] = Point(a.x+a.r*cos(theta),a.y+a.r*sin(theta));
21                     tp[tn++] = Point(b.x+b.r*cos(theta),b.y+b.r*sin(theta));
22                 }
23 }

```

6.4.6 两圆交点

```

1  lab = Point(p[j].x-p[i].x,p[j].y-p[i].y);
2  AB = lab.Length();
3  AC = cr[i];
4  BC = cr[j];
5
6  if (cmp(AB+AC,BC) <= 0) continue; //包含
7  if (cmp(AB+BC,AC) <= 0) continue;
8  if (cmp(AB,AC+BC) > 0) continue; //相离
9

```

```

10 theta = atan2(lab.y,lab.x);
11 fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
12 a0 = theta-fai;
13 if (cmp(a0,-pi) < 0) a0 += 2*pi;
14 a1 = theta+fai;
15 if (cmp(a1,pi) > 0) a1 -= 2*pi;
16 //答案
17 xp[totp++] = Point(p[i].x+cr[i]*cos(a0),p[i].y+cr[i]*sin(a0));
18 xp[totp++] = Point(p[i].x+cr[i]*cos(a1),p[i].y+cr[i]*sin(a1));

```

6.5 三角形相关

费马点：在 $\triangle ABC$ 内求一点 P ，使 $PA + PB + PC$ 之值为最小的点。当三角形有一个内角大于或等于 120° 的时候，费马点就是该内角的顶点
若没有，则费马点就是使得该点至三角形三顶点的连线两两夹角为 120° 的点。

等角共轭点：对于三角形内任意一点 P ，过 A 做直线 L_1 与 AP 关于角 A 的角平分线对称，同样过 B, C 分别做 L_2, L_3 。这三条直线交于 P_1 ，则 P_1 是 P 的等角共轭点。
重心的等角共轭点到三边距离的平方和最小的点。

6.6 矩阵

6.6.1 基本矩阵

按向量 $\overrightarrow{(x,y,z)}$ 平移：

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

按比例 (x,y,z) 缩放：

$$\begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

绕单位向量 $\overrightarrow{(x,y,z)}$ 旋转 $angle$ 角度：

$$\begin{pmatrix} x^2 \times (1-c) + c & x \times y \times (1-c) - z \times s & x \times z \times (1-c) + y \times s & 0 \\ y \times x \times (1-c) + z \times s & y^2 \times (1-c) + c & y \times z \times (1-c) - x \times s & 0 \\ x \times z \times (1-c) - y \times s & y \times z \times (1-c) + x \times s & z^2 \times (1-c) + c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{cases} s = \sin(angle) \\ c = \cos(angle) \end{cases}$$

以上矩阵变换都把点当作列向量，旋转角度的正负由右手定则决定

6.6.2 刘汝佳的几何教室

```

1  const double pi = acos(-1.0);
2
3  int n,m,q;
4  struct Point
5  {
6      double a,b,c,d;
7  };
8  Point p[50000],f[50000];
9
10 double a,b,c,theta,mt[4][4],tmp[4][4],tmt[4][4],rmt[4][8];
11 char com[20];
12
13 void TRANSLATE()
14 {
15     memset(tmt,0,sizeof(tmt));
16     tmt[0][0] = tmt[1][1] = tmt[2][2] = tmt[3][3] = 1;
17     tmt[3][0] = a;
18     tmt[3][1] = b;
19     tmt[3][2] = c;
20     memset(tmp,0,sizeof(tmp));
21     for (int i = 0; i < 4; i++)
22         for (int j = 0; j < 4; j++)
23             for (int k = 0; k < 4; k++)
24                 tmp[i][j] += mt[i][k]*tmt[k][j];
25     for (int i = 0; i < 4; i++)
26         for (int j = 0; j < 4; j++)
27             mt[i][j] = tmp[i][j];
28 }
29
30 void ROTATE()
31 {
32     theta = -theta*pi/180;
33     memset(tmt,0,sizeof(tmt));
34     tmt[3][3] = 1;
35     tmt[0][0] = cos(theta)+(1-cos(theta))*a*a;
36     tmt[1][0] = (1-cos(theta))*a*b+c*sin(theta);
37     tmt[2][0] = (1-cos(theta))*a*c-b*sin(theta);
38     tmt[0][1] = (1-cos(theta))*a*b-c*sin(theta);
39     tmt[1][1] = cos(theta)+(1-cos(theta))*b*b;
40     tmt[2][1] = (1-cos(theta))*b*c+a*sin(theta);
41     tmt[0][2] = (1-cos(theta))*a*c+b*sin(theta);
42     tmt[1][2] = (1-cos(theta))*b*c-a*sin(theta);
43     tmt[2][2] = cos(theta)+(1-cos(theta))*c*c;
44     memset(tmp,0,sizeof(tmp));
45     for (int i = 0; i < 4; i++)
46         for (int j = 0; j < 4; j++)
47             for (int k = 0; k < 4; k++)
48                 tmp[i][j] += mt[i][k]*tmt[k][j];
49     for (int i = 0; i < 4; i++)
50         for (int j = 0; j < 4; j++)
51             mt[i][j] = tmp[i][j];

```

```

52 }
53
54 void SCALE()
55 {
56     memset(tmt,0,sizeof(tmt));
57     tmt[0][0] = a;
58     tmt[1][1] = b;
59     tmt[2][2] = c;
60     tmt[3][3] = 1;
61     memset(tmp,0,sizeof(tmp));
62     for (int i = 0; i < 4; i++)
63         for (int j = 0; j < 4; j++)
64             for (int k = 0; k < 4; k++)
65                 tmp[i][j] += mt[i][k]*tmt[k][j];
66     for (int i = 0; i < 4; i++)
67         for (int j = 0; j < 4; j++)
68             mt[i][j] = tmp[i][j];
69 }
70
71 void solvep(Point p)
72 {
73     memset(tmt,0,sizeof(tmt));
74     tmt[0][0] = p.a;
75     tmt[0][1] = p.b;
76     tmt[0][2] = p.c;
77     tmt[0][3] = 1;
78     memset(tmp,0,sizeof(tmp));
79     for (int i = 0; i < 1; i++)
80         for (int j = 0; j < 4; j++)
81             for (int k = 0; k < 4; k++)
82                 tmp[i][j] += tmt[i][k]*mt[k][j];
83     printf("%.2f_%.2f_%.2f\n",tmp[0][0],tmp[0][1],tmp[0][2]);
84 }
85
86 void solvef(Point f)
87 {
88     memset(tmt,0,sizeof(tmt));
89     tmt[0][0] = f.a;
90     tmt[1][0] = f.b;
91     tmt[2][0] = f.c;
92     tmt[3][0] = 0;
93     memset(tmp,0,sizeof(tmp));
94     for (int i = 0;i < 4;i++)
95         for (int j = 0;j < 1;j++)
96             for (int k = 0;k < 4;k++)
97                 tmp[i][j] += mt[i][k]*tmt[k][j];
98     tmp[3][0] += f.d;
99     double kk = tmp[0][0]*tmp[0][0]+tmp[1][0]*tmp[1][0]+tmp[2][0]*tmp
100         [2][0];
101     kk = sqrt(1/kk);
102     for (int i = 0;i < 4;i++)

```

```

102     printf("%.2f",tmp[i][0]*kk);
103     printf("\n");
104 }
105
106 void solvermt()
107 {
108     memset(rmt,0,sizeof(rmt));
109     for (int i = 0;i < 4;i++)
110         for (int j = 0;j < 4;j++)
111             rmt[i][j] = mt[i][j];
112     rmt[0][4] = rmt[1][5] = rmt[2][6] = rmt[3][7] = 1;
113     for (int i = 0;i < 4;i++)
114     {
115         for (int j = i;j < 4;j++)
116             if (fabs(rmt[j][i]) > 1e-8)
117             {
118                 for (int k = i;k < 8;k++)
119                     swap(rmt[i][k],rmt[j][k]);
120                 break;
121             }
122         double tt = rmt[i][i];
123         for (int j = i;j < 8;j++)
124             rmt[i][j] /= tt;
125         for (int j = 0;j < 4;j++)
126             if (i != j)
127             {
128                 tt = rmt[j][i];
129                 for (int k = i;k < 8;k++)
130                     rmt[j][k] -= rmt[i][k]*tt;
131             }
132     }
133     for (int i = 0;i < 4;i++)
134         for (int j = 0;j < 4;j++)
135             mt[i][j] = rmt[i][4+j];
136 }
137
138 int main()
139 {
140     scanf("%d%d%d",&n,&m,&q);
141     for (int i = 0; i < n; i++)
142         scanf("%lf%lf%lf",&p[i].a,&p[i].b,&p[i].c);
143     for (int i = 0; i < m; i++)
144         scanf("%lf%lf%lf%lf",&f[i].a,&f[i].b,&f[i].c,&f[i].d);
145     memset(mt,0,sizeof(mt));
146     mt[0][0] = mt[1][1] = mt[2][2] = mt[3][3] = 1;
147     for (int i = 0; i < q; i++)
148     {
149         scanf("%s",com);
150         if (strcmp(com,"TRANSLATE") == 0)
151         {
152             scanf("%lf%lf%lf",&a,&b,&c);

```



```

153     TRANSLATE();
154 }
155 else if (strcmp(com,"ROTATE") == 0)
156 {
157     scanf("%lf%lf%lf%lf",&a,&b,&c,&theta);
158     ROTATE();
159 }
160 else if (strcmp(com,"SCALE") == 0)
161 {
162     scanf("%lf%lf%lf",&a,&b,&c);
163     SCALE();
164 }
165 }
166 //处理点
167 for (int i = 0; i < n; i++)
168     solvep(p[i]);
169 //处理面
170 solvermt();
171 for (int i = 0; i < m; i++)
172     solvef(f[i]);
173 return 0;
174 }

```

6.7 重心

```

1 Point CenterOfPolygon(Point poly[],int n)
2 {
3     Point p, p0, p1, p2, p3;
4     double m, m0;
5     p1 = poly[0];
6     p2 = poly[1];
7     p.x = p.y = m = 0;
8     for (int i = 2; i < n; i++)
9     {
10         p3 = poly[i];
11         p0.x = (p1.x + p2.x + p3.x) / 3.0;
12         p0.y = (p1.y + p2.y + p3.y) / 3.0;
13         m0 = p1.x*p2.y+p2.x*p3.y+p3.x*p1.y-p1.y*p2.x-p2.y*p3.x-p3.y*p1.
            x;
14         if (cmp(m + m0,0.0) == 0)
15             m0 += eps;
16         p.x = (m * p.x + m0 * p0.x) / (m + m0);
17         p.y = (m * p.y + m0 * p0.y) / (m + m0);
18         m = m + m0;
19         p2 = p3;
20     }
21     return p;
22 }

```

6.8 KD 树

查找某个点距离最近的点，基本思想是每次分治把点分成两部分，建议按照坐标规模决定是垂直划分还是水平划分，查找时先往分到的那一部分查找，然后根据当前最优答案决定是否

去另一个区间查找。

```

1  bool Div[MaxN];
2  void BuildKD(int deep,int l, int r, Point p[])\ \记得备份一下 P
3  {
4      if (l > r) return;
5      int mid = l + r >> 1;
6      int minX, minY, maxX, maxY;
7      minX = min_element(p + l, p + r + 1, cmpX)->x;
8      minY = min_element(p + l, p + r + 1, cmpY)->y;
9      maxX = max_element(p + l, p + r + 1, cmpX)->x;
10     maxY = max_element(p + l, p + r + 1, cmpY)->y;
11     Div[mid] = (maxX - minX >= maxY - minY);
12     nth_element(p + l, p + mid, p + r + 1, Div[mid] ? cmpX : cmpY);
13     BuildKD(l, mid - 1, p);
14     BuildKD(mid + 1, r, p);
15 }
16
17 long long res;
18 void Find(int l, int r, Point a, Point p[])\ \查找
19 {
20     if (l > r) return;
21     int mid = l + r >> 1;
22     long long dist = dist2(a, p[mid]);
23     if (dist > 0) //如果有重点不能这样判断
24         res = min(res, dist);
25     long long d = Div[mid] ? (a.x - p[mid].x) : (a.y - p[mid].y);
26     int l1, l2, r1, r2;
27     l1 = l, l2 = mid + 1;
28     r1 = mid - 1, r2 = r;
29     if (d > 0)
30         swap(l1, l2), swap(r1, r2);
31     Find(l1, r1, a, p);
32     if (d * d < res)
33         Find(l2, r2, a, p);
34 }

```

6.8.1 例题

查询一个点为中心的给定正方形内所有点并删除 (2012 金华网赛 A)

```

1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <algorithm>
5  #include <cmath>
6  #include <queue>
7  using namespace std;
8
9  const int MaxN = 100000;
10 struct Point

```

```

11 {
12     int x,y,r;
13     int id;
14     bool del;
15 };
16
17 int cmpTyp;
18 bool cmp(const Point& a,const Point& b)
19 {
20     if (cmpTyp == 0)
21         return a.x < b.x;
22     else
23         return a.y < b.y;
24 }
25
26 int cnt[MaxN];
27 bool Div[MaxN];
28 int minX[MaxN],minY[MaxN],maxX[MaxN],maxY[MaxN];
29 void BuildKD(int l,int r,Point p[])
30 {
31     if (l > r) return;
32     int mid = l+r>>1;
33     cmpTyp = 0;
34     minX[mid] = min_element(p+l,p+r+1,cmp)->x;
35     maxX[mid] = max_element(p+l,p+r+1,cmp)->x;
36     cmpTyp = 1;
37     minY[mid] = min_element(p+l,p+r+1,cmp)->y;
38     maxY[mid] = max_element(p+l,p+r+1,cmp)->y;
39
40     cnt[mid] = r-l+1;
41     cmpTyp = Div[mid] = (maxX[mid]-minX[mid] < maxY[mid]-minY[mid]);
42     nth_element(p+l,p+mid,p+r+1,cmp);
43     BuildKD(l,mid-1,p);
44     BuildKD(mid+1,r,p);
45 }
46
47 queue<int> Q;
48 int Find(int l,int r,Point a,Point p[])
49 {
50     if (l > r) return 0;
51     int mid = l+r>>1;
52     if (cnt[mid] == 0) return 0;
53
54     if (maxX[mid] < a.x-a.r ||
55         minX[mid] > a.x+a.r ||
56         maxY[mid] < a.y-a.r ||
57         minY[mid] > a.y+a.r)
58         return 0;
59
60     int totdel = 0;
61

```

```

62     if (p[mid].del == false)
63         if (abs(p[mid].x-a.x) <= a.r && abs(p[mid].y-a.y) <= a.r)
64         {
65             p[mid].del = true;
66             Q.push(p[mid].id);
67             totdel++;
68         }
69
70     totdel += Find(l,mid-1,a,p);
71     totdel += Find(mid+1,r,a,p);
72
73     cnt[mid] -= totdel;
74
75     return totdel;
76 }
77
78 Point p[MaxN],tp[MaxN];
79 int n;
80
81 int main()
82 {
83     int cas = 1;
84     while (true)
85     {
86         scanf("%d",&n);
87         if (n == 0) break;
88
89         for (int i = 0;i < n;i++)
90         {
91             p[i].id = i;
92             int tx,ty;
93             scanf("%d%d",&tx,&ty,&p[i].r);
94             p[i].x = tx-ty;
95             p[i].y = tx+ty;
96             p[i].del = false;
97             tp[i] = p[i];
98         }
99         BuildKD(0,n-1,tp);
100
101         printf("Case_#%d:\n",cas++);
102         int q;
103         scanf("%d",&q);
104         for (int i = 0;i < q;i++)
105         {
106             int id;
107             scanf("%d",&id);
108             int res = 0;
109             id--;
110             Q.push(id);
111             while (!Q.empty())
112             {

```

```

113     int now = Q.front();
114     Q.pop();
115     if (p[now].del == true) continue;
116     p[now].del = true;
117     res += Find(0,n-1,p[now],tp);
118 }
119 printf("%d\n",res);
120 }
121 }
122 return 0;
123 }

```

6.9 半平面交

直线左边代表有效区域。

```

1 bool HPIcmp(Line a, Line b)
2 {
3     if (fabs(a.k - b.k) > eps) return a.k < b.k;
4     return ((a.s - b.s) * (b.e - b.s)) < 0;
5 }
6
7 Line Q[100];
8 void HPI(Line line[], int n, Point res[], int &resn)
9 {
10     int tot = n;
11     sort(line, line + n, HPIcmp);
12     tot = 1;
13     for (int i = 1; i < n; i++)
14         if (fabs(line[i].k - line[i - 1].k) > eps)
15             line[tot++] = line[i];
16     int head = 0, tail = 1;
17     Q[0] = line[0];
18     Q[1] = line[1];
19     resn = 0;
20     for (int i = 2; i < tot; i++)
21     {
22         if (fabs((Q[tail].e - Q[tail].s) * (Q[tail - 1].e - Q[tail - 1].s)) <
23             eps ||
24             fabs((Q[head].e - Q[head].s) * (Q[head + 1].e - Q[head + 1].s)) <
25             eps)
26             return;
27         while (head < tail && (((Q[tail] & Q[tail - 1]) - line[i].s) * (
28             line[i].e - line[i].s)) > eps)
29             tail--;
30         while (head < tail && (((Q[head] & Q[head + 1]) - line[i].s) * (
31             line[i].e - line[i].s)) > eps)
32             head++;
33         Q[++tail] = line[i];
34     }
35     while (head < tail && (((Q[tail] & Q[tail - 1]) - Q[head].s) * (Q[
36         head].e - Q[head].s)) > eps)
37         head++;
38     resn = tail - head;
39 }

```

```

32     tail--;
33     while (head < tail && (((Q[head]&Q[head + 1]) - Q[tail].s) * (Q[
        tail].e-Q[tail].s)) > eps)
34         head++;
35     if (tail <= head + 1) return;
36     for (int i = head; i < tail; i++)
37         res[resn++] = Q[i] & Q[i + 1];
38     if (head < tail + 1)
39         res[resn++] = Q[head] & Q[tail];
40 }

```

6.10 凸包

得到的凸包按照逆时针方向排序。

```

1 //判断是否是共点或者共线用
2 bool conPoint(Point p[],int n)
3 {
4     for (int i = 1;i < n;i++)
5         if (p[i].x != p[0].x || p[i].y != p[0].y)
6             return false;
7     return true;
8 }
9 bool conLine(Point p[],int n)
10 {
11     for (int i = 2;i < n;i++)
12         if ((p[i]-p[0])*(p[1]-p[0]) != 0)
13             return false;
14     return true;
15 }
16
17 bool GScmp(Point a, Point b)
18 {
19     if (fabs(a.x - b.x) < eps)
20         return a.y < b.y - eps;
21     return a.x < b.x - eps;
22 }
23
24 void GS(Point p[],int n,Point res[],int &resn)
25 {
26     resn = 0;
27     int top = 0;
28     sort(p,p+n,GScmp);
29
30     if (conPoint(p,n))
31     {
32         res[resn++] = p[0];
33         return;
34     }
35     if (conLine(p,n))
36     {
37         res[resn++] = p[0];

```

```

38     res[resn++] = p[n-1];
39     return;
40 }
41
42 for (int i = 0; i < n; i++)
43     if (resn < 2 ||
44         (res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)
45         res[resn++] = p[i++];
46     else
47         --resn;
48 top = resn-1;
49 for (int i = n-2; i >= 0; i--)
50     if (resn < top+2 ||
51         (res[resn-1]-res[resn-2])*(p[i]-res[resn-1]) > 0)
52         res[resn++] = p[i--];
53     else
54         --resn;
55 resn--;
56 }

```

6.11 直线与凸包求交点

复杂度 $O(\log n)$ 。
需要先预处理几个东西。

```

1 //二分 [la,lb] 这段区间那条边与 line 相交
2 int Gao(int la,int lb,Line line)
3 {
4     if (la > lb)
5         lb += n;
6     int l = la,r = lb,mid;
7     while (l < r)
8     {
9         mid = l+r+1>>1;
10        if (cmp((line.e-line.s)*(p[la]-line.s),0)*cmp((line.e-line.s)*(
11            p[mid]-line.s),0) >= 0)
12            l = mid;
13        else
14            r = mid-1;
15    }
16    return l%n;
17 }
18 //求 l 与凸包的交点
19 //先调用 Gettheta 预处理出凸包每条边的斜率，然后处理成升序排列
20 double theta[maxn];
21
22 void Gettheta()
23 {
24     for (int i = 0; i < n; i++)
25     {

```

```

26     Point v = p[(i+1)%n]-p[i];
27     theta[i] = atan2(v.y,v.x);
28 }
29 for (int i = 1;i < n;i++)
30     if (theta[i-1] > theta[i]+eps)
31         theta[i] += 2*pi;
32 }
33
34 double Calc(Line l)
35 {
36     double tnow;
37     Point v = l.e-l.s;
38     tnow = atan2(v.y,v.x);
39     if (cmp(tnow,theta[0]) < 0) tnow += 2*pi;
40     int pl = lower_bound(theta,theta+n,tnow)-theta;
41     tnow = atan2(-v.y,-v.x);
42     if (cmp(tnow,theta[0]) < 0) tnow += 2*pi;
43     int pr = lower_bound(theta,theta+n,tnow)-theta;
44     //pl 和 pr 是在 l 方向上距离最远的点对
45     pl = pl%n;
46     pr = pr%n;
47
48     if (cmp(v*(p[pl]-l.s),0)*cmp(v*(p[pr]-l.s),0) >= 0)
49         return 0.0;
50
51     int xa = Gao(pl,pr,l);
52     int xb = Gao(pr,pl,l);
53
54     if (xa > xb) swap(xa,xb);
55     //与 [xa,xa+1] 和 [xb,xb+1] 这两条线段相交
56
57     if (cmp(v*(p[xa+1]-p[xa]),0) == 0) return 0.0;
58     if (cmp(v*(p[xb+1]-p[xb]),0) == 0) return 0.0;
59
60     Point pa,pb;
61     pa = Line(p[xa],p[xa+1])&l;
62     pb = Line(p[xb],p[xb+1])&l;
63     //题目：求直线切凸包得到的两部分的面积
64     double area0 = sum[xb]-sum[xa+1]+(pa*p[xa+1])/2.0+(p[xb]*pb)
        /2.0+(pb*pa)/2.0;
65     double area1 = sum[xa+n]-sum[xb+1]+(pb*p[xb+1])/2.0+(p[xa]*pa)
        /2.0+(pa*pb)/2.0;
66
67     return min(area0,area1);
68 }

```

6.12 点对凸包的两切点

过了 sgu500 的前七组数据，用前需谨慎，虽然我不认为这个有问题。

```

1 double theta[MaxN];
2 void Gettheta(Point p[],int n)

```



```

3 {
4   for (int i = 0; i < n; i++)
5   {
6     Point v = p[(i+1)%n]-p[i];
7     theta[i] = atan2(v.y,v.x);
8   }
9   for (int i = 1; i < n; i++)
10    if (theta[i-1] > theta[i]+eps)
11      theta[i] += 2*pi;
12 }
13 int cmp(double a,double b)
14 {
15   if (fabs(a-b) < eps) return 0;
16   if (a < b) return -1;
17   return 1;
18 }
19 int Gao(int la,int lb,Line line,Point p[],int n)
20 {
21   if (la > lb)
22     lb += n;
23   int l = la,r = lb,mid;
24   while (l < r)
25   {
26     mid = l+r+1>>1;
27     if (cmp((line.e-line.s)*(p[la%n]-line.s),0)*cmp((line.e-line.s)
28       *(p[mid%n]-line.s),0) >= 0)
29       l = mid;
30     else
31       r = mid-1;
32   }
33   return l%n;
34 }
35 int Gao(int la,int lb,int dir,Point s,Point p[],int n)
36 {
37   if (la > lb)
38     lb += n;
39   if (la == lb) return la;
40   int l = la+1,r = lb,mid;
41
42   while (l < r)
43   {
44     mid = l+r+1>>1;
45
46     int ret = cmp((p[mid%n]-s)*(p[(mid-1)%n]-s),0);
47     if (dir*ret < 0)
48       l = mid;
49     else if (dir*ret > 0)
50       r = mid-1;
51     else
52   {

```

```

53     if (dir == 1)
54         l = mid;
55     else
56         r = mid-1;
57 }
58 }
59
60 int ret = cmp((p[l%n]-s)*(p[(l-1)%n]-s),0);
61 if (dir*ret < 0)
62     return l%n;
63 else if (dir*ret > 0)
64     return (l-1)%n;
65 else
66 {
67     if (dir == 1)
68         return l%n;
69     else
70         return (l-1)%n;
71 }
72 }
73 //Gettheta(p,n) first!
74 //返回 S 对于 p[] 的两个切点 p[pl],p[pr]
75 void Calc(Point s,Point p[],int n,int& pl,int& pr)
76 {
77     Line l = Line(s,p[0]);
78     Point v = l.e-l.s;
79     double tnow = atan2(v.y,v.x);
80     if (tnow < theta[0]-eps) tnow += 2*pi;
81     int tpl = lower_bound(theta,theta+n,tnow)-theta;
82     tnow = atan2(-v.y,-v.x);
83     if (tnow < theta[0]-eps) tnow += 2*pi;
84     int tpr = lower_bound(theta,theta+n,tnow)-theta;
85
86     pl = tpl = tpl%n;
87     pr = tpr = tpr%n;
88
89     int px = Gao(pr,pl,l,p,n);
90     //printf("pr = %d -> px = %d\n",tpr,px);
91     //printf("px = %d -> pl = %d\n",px,tpl);
92     //pr -> px
93     //px -> pl
94
95     pl = Gao(tpr,px,1,s,p,n);
96     pr = Gao(px,tpl,-1,s,p,n);
97
98 }

```

6.13 三维凸包

暴力写法

```
1 #define eps 1e-7
```

```

2 #define MAXV 505
3
4 struct pt
5 {
6     double x, y, z;
7     pt() {}
8     pt(double _x, double _y, double _z): x(_x), y(_y), z(_z) {}
9     pt operator - (const pt p1)
10    {
11        return pt(x - p1.x, y - p1.y, z - p1.z);
12    }
13    pt operator * (pt p)
14    {
15        return pt(y*p.z-z*p.y, z*p.x-x*p.z, x*p.y-y*p.x);
16    }
17    double operator ^ (pt p)
18    {
19        return x*p.x+y*p.y+z*p.z;
20    }
21 };
22 struct _3DCH
23 {
24     struct fac
25     {
26         int a, b, c;
27         bool ok;
28     };
29     int n;
30     pt P[MAXV];
31     int cnt;
32     fac F[MAXV*8];
33     int to[MAXV][MAXV];
34     double vlen(pt a)
35     {
36         return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
37     }
38     double area(pt a, pt b, pt c)
39     {
40         return vlen((b-a)*(c-a));
41     }
42     double volume(pt a, pt b, pt c, pt d)
43     {
44         return (b-a)*(c-a)^(d-a);
45     }
46     double ptof(pt &p, fac &f)
47     {
48         pt m = P[f.b]-P[f.a], n = P[f.c]-P[f.a], t = p-P[f.a];
49         return (m * n) ^ t;
50     }
51     void deal(int p, int a, int b)
52     {

```

```

53     int f = to[a][b];
54     fac add;
55     if (F[f].ok)
56     {
57         if (ptof(P[p], F[f]) > eps)
58             dfs(p, f);
59         else
60         {
61             add.a = b, add.b = a, add.c = p, add.ok = 1;
62             to[p][b] = to[a][p] = to[b][a] = cnt;
63             F[cnt++] = add;
64         }
65     }
66 }
67 void dfs(int p, int cur)
68 {
69     F[cur].ok = 0;
70     deal(p, F[cur].b, F[cur].a);
71     deal(p, F[cur].c, F[cur].b);
72     deal(p, F[cur].a, F[cur].c);
73 }
74 bool same(int s, int t)
75 {
76     pt &a = P[F[s].a], &b = P[F[s].b], &c = P[F[s].c];
77     return fabs(volume(a, b, c, P[F[t].a])) < eps && fabs(volume(a,
78         b, c,
79         P[F[t].b])) < eps && fabs(volume(a, b, c, P[F[t].c])) < eps
80         ;
81 }
82 void construct()
83 {
84     cnt = 0;
85     if (n < 4)
86         return;
87     bool sb = 1;
88     for (int i = 1; i < n; i++)
89     {
90         if (vlen(P[0] - P[i]) > eps)
91         {
92             swap(P[1], P[i]);
93             sb = 0;
94             break;
95         }
96     }
97     if (sb) return;
98     sb = 1;
99     for (int i = 2; i < n; i++)
100     {
101         if (vlen((P[0] - P[1]) * (P[1] - P[i])) > eps)
102         {
103             swap(P[2], P[i]);

```

```

102         sb = 0;
103         break;
104     }
105 }
106 if (sb) return;
107 sb = 1;
108 for (int i = 3; i < n; i++)
109 {
110     if (fabs((P[0] - P[1]) * (P[1] - P[2]) ^ (P[0] - P[i])) > eps
111         )
112     {
113         swap(P[3], P[i]);
114         sb = 0;
115         break;
116     }
117 if (sb) return;
118 fac add;
119 for (int i = 0; i < 4; i++)
120 {
121     add.a = (i+1)%4, add.b = (i+2)%4, add.c = (i+3)%4, add.ok =
122         1;
123     if (ptof(P[i], add) > 0)
124         swap(add.b, add.c);
125     to[add.a][add.b] = to[add.b][add.c] = to[add.c][add.a] = cnt;
126     F[cnt++] = add;
127 }
128 for (int i = 4; i < n; i++)
129 {
130     for (int j = 0; j < cnt; j++)
131     {
132         if (F[j].ok && ptof(P[i], F[j]) > eps)
133         {
134             dfs(i, j);
135             break;
136         }
137     }
138 int tmp = cnt;
139 cnt = 0;
140 for (int i = 0; i < tmp; i++)
141 {
142     if (F[i].ok)
143     {
144         F[cnt++] = F[i];
145     }
146 }
147 }
148 //表面积
149 double area()
150 {

```

```

151     double ret = 0.0;
152     for (int i = 0; i < cnt; i++)
153     {
154         ret += area(P[F[i].a], P[F[i].b], P[F[i].c]);
155     }
156     return ret / 2.0;
157 }
158 //体积
159 double volume()
160 {
161     pt O(0, 0, 0);
162     double ret = 0.0;
163     for (int i = 0; i < cnt; i++)
164     {
165         ret += volume(O, P[F[i].a], P[F[i].b], P[F[i].c]);
166     }
167     return fabs(ret / 6.0);
168 }
169 //表面三角形数
170 int facetCnt_tri()
171 {
172     return cnt;
173 }
174 //表面多边形数
175 int facetCnt()
176 {
177     int ans = 0;
178     for (int i = 0; i < cnt; i++)
179     {
180         bool nb = 1;
181         for (int j = 0; j < i; j++)
182         {
183             if (same(i, j))
184             {
185                 nb = 0;
186                 break;
187             }
188         }
189         ans += nb;
190     }
191     return ans;
192 }
193
194 pt Fc[MAXV*8];
195 double V[MAXV*8];
196 pt Center()//重心
197 {
198     pt O(0,0,0);
199     for (int i = 0; i < cnt; i++)
200     {
201         Fc[i].x = (O.x+P[F[i].a].x+P[F[i].b].x+P[F[i].c].x)/4.0;

```

```

202     Fc[i].y = (O.y+P[F[i].a].y+P[F[i].b].y+P[F[i].c].y)/4.0;
203     Fc[i].z = (O.z+P[F[i].a].z+P[F[i].b].z+P[F[i].c].z)/4.0;
204     V[i] = volume(O,P[F[i].a],P[F[i].b],P[F[i].c]);
205 }
206 pt res = Fc[0],tmp;
207 double m = V[0];
208 for (int i = 1; i < cnt; i++)
209 {
210     if (fabs(m+V[i]) < eps)
211         V[i] += eps;
212     tmp.x = (m*res.x+V[i]*Fc[i].x)/(m+V[i]);
213     tmp.y = (m*res.y+V[i]*Fc[i].y)/(m+V[i]);
214     tmp.z = (m*res.z+V[i]*Fc[i].z)/(m+V[i]);
215     m += V[i];
216     res = tmp;
217 }
218 return res;
219 }
220 };
221
222 _3DCH hull;
223
224 int main()
225 {
226     while (scanf("%d",&hull.n) != EOF)
227     {
228         for (int i = 0; i < hull.n; i++)
229             scanf("%lf%lf%lf",&hull.P[i].x,&hull.P[i].y,&hull.P[i].z);
230         hull.construct();
231     }
232     return 0;
233 }

```

6.14 旋转卡壳

“对踵”

6.14.1 单个凸包

```

1 void solve(Point p[],int n)
2 {
3     Point v;
4     int cur = 1;
5     for (int i = 0; i < n; i++)
6     {
7         v = p[i]-p[(i+1)%n];
8         while (v*(p[(cur+1)%n]-p[cur]) < 0)
9             cur = (cur+1)%n;
10        //p[cur] -> p[i]
11        //p[cur] -> p[i+1]

```

```

12     //p[cur] -> (p[i],p[i+1])
13 }
14 }

```

6.14.2 两个凸包

注意初始点的选取，代码只是个示例。

有时候答案需要取 $\text{solve}(p_0, n, p_1, m)$ 和 $\text{solve}(p_1, m, p_0, n)$ 的最优值。

何老鱼说我是错的。。

```

1 void solve(Point p0[],int n,Point p1[],int m)
2 {
3     Point v;
4     int cur = 0;
5     for (int i = 0; i < n; i++)
6     {
7         v = p0[i]-p0[(i+1)%n];
8         while (v*(p1[(cur+1)%m]-p1[cur]) < 0)
9             cur = (cur+1)%m;
10        //p1[cur] -> p0[i]
11        //p1[cur] -> p0[i+1]
12        //p1[cur] -> (p0[i],p0[i+1])
13    }
14 }

```

6.14.3 外接矩形

```

1 void solve()
2 {
3     resa = resb = 1e100;
4     double dis1,dis2;
5     Point xp[4];
6     Line l[4];
7     int a,b,c,d;
8     int sa,sb,sc,sd;
9     a = b = c = d = 0;
10    sa = sb = sc = sd = 0;
11    Point va,vb,vc,vd;
12    for (a = 0; a < n; a++)
13    {
14        va = Point(p[a],p[(a+1)%n]);
15        vc = Point(-va.x,-va.y);
16        vb = Point(-va.y,va.x);
17        vd = Point(-vb.x,-vb.y);
18        if (sb < sa)
19        {
20            b = a;
21            sb = sa;
22        }
23        while (xmult(vb,Point(p[b],p[(b+1)%n])) < 0)
24        {
25            b = (b+1)%n;

```



```

26     sb++;
27 }
28 if (sc < sb)
29 {
30     c = b;
31     sc = sb;
32 }
33 while (xmult(vc,Point(p[c],p[(c+1)%n])) < 0)
34 {
35     c = (c+1)%n;
36     sc++;
37 }
38 if (sd < sc)
39 {
40     d = c;
41     sd = sc;
42 }
43 while (xmult(vd,Point(p[d],p[(d+1)%n])) < 0)
44 {
45     d = (d+1)%n;
46     sd++;
47 }
48
49 //卡在 p[a],p[b],p[c],p[d] 上
50 sa++;
51 }
52 }

```

6.15 三角形内点个数

6.15.1 无三点共线

```

1 Point p[1000], tp[2000], base;
2
3 bool cmp(const Point &a, const Point &b)
4 {
5     return a.theta < b.theta;
6 }
7
8 int cnt[1000][1000];
9 int cntleft[1000][1000];
10 int n, m;
11
12 int calc(int a, int b, int c)
13 {
14     Point p1 = p[b] - p[a], p2 = p[c] - p[a];
15     if (atan2(p1.y, p1.x) > atan2(p2.y, p2.x))
16         swap(b, c);
17     if ((p[b] - p[a]) * (p[c] - p[a]) > 0)
18         return cnt[a][c] - cnt[a][b] - 1;
19     else
20         return n - 3 - (cnt[a][c] - cnt[a][b] - 1);

```

```

21 }
22
23 int main(int argc, char const *argv[])
24 {
25     int totcas;
26     scanf("%d", &totcas);
27     for (int cas = 1; cas <= totcas; ++cas)
28     {
29         scanf("%d", &n);
30         for (int i = 0; i < n; ++i)
31         {
32             scanf("%lld%lld", &p[i].x, &p[i].y);
33             p[i].id = i;
34         }
35         for (int i = 0; i < n; ++i)
36         {
37             m = 0;
38             base = p[i];
39             for (int j = 0; j < n; ++j)
40                 if (i != j)
41                 {
42                     tp[m] = p[j];
43                     Point v = tp[m] - base;
44                     tp[m++].theta = atan2(v.y, v.x);
45                 }
46
47             sort(tp, tp + m, cmp);
48             for (int j = 0; j < m; ++j)
49                 tp[m + j] = tp[j];
50
51             //calc cnt
52             for (int j = 0; j < m; ++j)
53                 cnt[i][tp[j].id] = j;
54
55             //calc cntleft
56             for (int j = 0, k = 0, tot = 0; j < m; ++j)
57             {
58                 while (k == j || (k < j + m && (tp[j] - base) * (tp[k] -
59                     base) > 0))
60                     k++, tot++;
61                 cntleft[i][tp[j].id] = —tot;
62             }
63         }
64
65         printf("Case_ %d: \n", cas);
66         int q;
67         scanf("%d", &q);
68         for (int i = 0; i < q; ++i)
69         {
70             int x, y, z;
71             scanf("%d%d%d", &x, &y, &z);

```

```

71     if ((p[z] - p[x]) * (p[y] - p[x]) > 0)
72         swap(y, z);
73     int res = cntleft[x][z] + cntleft[z][y] + cntleft[y][x];
74     res += calc(x, y, z) + calc(y, z, x) + calc(z, x, y);
75     res -= 2 * (n - 3);
76     printf("%d\n", res);
77 }
78 }
79 return 0;
80 }

```

6.15.2 有三点共线且点有类别之分

```

1  int n,n0,n1,m;
2  Point p[3000], tp[3000], base;
3
4  bool cmp(const Point &a, const Point &b)
5  {
6      if ((a-base)*(b-base) == 0)
7      {
8          return (a-base).getMol() < (b-base).getMol();
9      }
10     return a.theta < b.theta;
11 }
12
13 int cnt[100][100];
14 int cntleft[100][100];
15
16 int calc(int a,int b,int c)
17 {
18     Point p1 = p[b]-p[a],p2 = p[c]-p[a];
19     if (atan2(1.0*p1.y,1.0*p1.x) > atan2(1.0*p2.y,1.0*p2.x))
20         swap(b,c);
21     int res = cnt[a][c]-cnt[a][b];
22     if ((p[b]-p[a])*(p[c]-p[a]) > 0)
23         return res;
24     else
25         return n1-res;
26 }
27
28 int main()
29 {
30     int cas = 0;
31     while (scanf("%d%d",&n0,&n1) != EOF)
32     {
33         n = n1+n0;
34         for (int i = 0; i < n; i++)
35         {
36             scanf("%I64d%I64d",&p[i].x,&p[i].y);
37             p[i].id = i;
38         }
39         for (int i = 0; i < n0; ++i)

```

```

40     {
41         m = 0;
42         base = p[i];
43         for (int j = 0; j < n; ++j)
44             if (i != j)
45                 {
46                     tp[m] = p[j];
47                     Point v = tp[m] - base;
48                     tp[m++].theta = atan2(1.0*v.y, 1.0*v.x);
49                 }
50
51         sort(tp, tp + m, cmp);
52         for (int j = 0; j < m; ++j)
53             tp[m + j] = tp[j];
54
55         for (int j = 0, tot = 0; j < m; ++j)
56         {
57             if (tp[j].id < n0)
58                 cnt[i][tp[j].id] = tot;
59             else
60                 tot++;
61         }
62
63         for (int j = 0, k = 0, tot = 0; j < m; ++j)
64         {
65             while (k == j || (k < j + m && (tp[j] - base) * (tp[k] -
66                 base) > 0))
67                 {
68                     if (tp[k].id >= n0)
69                         tot++;
70                     k++;
71                 }
72             if (tp[j].id >= n0)
73                 tot--;
74             else
75                 cntleft[i][tp[j].id] = tot;
76         }
77
78         int ans = 0;
79         for (int i = 0; i < n0; i++)
80             for (int j = i+1; j < n0; j++)
81                 for (int k = j+1; k < n0; k++)
82                     {
83                         int x = i, y = j, z = k;
84
85                         if ((p[z] - p[x]) * (p[y] - p[x]) > 0)
86                             swap(y, z);
87                         int res = cntleft[x][z] + cntleft[z][y] + cntleft[y][x];
88
89                         res += calc(x, y, z) + calc(y, z, x) + calc(z, x, y);

```

```

90
91         res -= 2 * n1;
92
93         //printf("%d %d %d %d\n",x,y,z,res);
94
95         if (res%2 == 1)
96             ans++;
97     }
98     printf("Case_%d:_%d\n",++cas,ans);
99 }
100 return 0;
101 }

```

6.16 最近点对

6.16.1 类快排算法

```

1 double calc_dis(Point &a ,Point &b) {
2     return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
3 }
4 //别忘了排序
5 bool operator<(const Point &a ,const Point &b) {
6     if(a.y != b.y) return a.x < b.x;
7     return a.x < b.x;
8 }
9 double Gao(int l ,int r ,Point pnts[]) {
10     double ret = inf;
11     if(l == r) return ret;
12     if(l+1 ==r) {
13         ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
14         return ret;
15     }
16     if(l+2 ==r) {
17         ret = min(calc_dis(pnts[l],pnts[l+1]) ,ret);
18         ret = min(calc_dis(pnts[l],pnts[l+2]) ,ret);
19         ret = min(calc_dis(pnts[l+1],pnts[l+2]) ,ret);
20         return ret;
21     }
22
23     int mid = l+r>>1;
24     ret = min (ret ,Gao(l ,mid,pnts));
25     ret = min (ret , Gao(mid+1, r,pnts));
26
27     for(int c = l ; c<=r; c++)
28         for(int d = c+1; d <=c+7 && d<=r; d++) {
29             ret = min(ret , calc_dis(pnts[c],pnts[d]));
30         }
31     return ret;
32 }

```

6.16.2 随机增量法

```

1 #include <iostream>

```

```

2  #include <cstdio>
3  #include <cstring>
4  #include <map>
5  #include <vector>
6  #include <cmath>
7  #include <algorithm>
8  #define Point pair<double,double>
9  using namespace std;
10
11 const int step[9][2] =
    {{-1,-1},{-1,0},{-1,1},{0,-1},{0,0},{0,1},{1,-1},{1,0},{1,1}};
12 int n,x,y,nx,ny;
13 map<pair<int,int>,vector<Point > > g;
14 vector<Point > tmp;
15 Point p[20000];
16 double tx,ty,ans,nowans;
17 vector<Point >::iterator it,op,ed;
18 pair<int,int> gird;
19 bool flag;
20
21 double Dis(Point p0,Point p1)
22 {
23     return sqrt((p0.first-p1.first)*(p0.first-p1.first)+
24               (p0.second-p1.second)*(p0.second-p1.second));
25 }
26
27 double CalcDis(Point p0,Point p1,Point p2)
28 {
29     return Dis(p0,p1)+Dis(p0,p2)+Dis(p1,p2);
30 }
31
32 void build(int n,double w)
33 {
34     g.clear();
35     for (int i = 0;i < n;i++)
36         g[make_pair((int)floor(p[i].first/w),(int)floor(p[i].second/w))
37           ].push_back(p[i]);
38
39 int main()
40 {
41     int t;
42     scanf("%d",&t);
43     for (int ft = 1;ft <= t;ft++)
44     {
45         scanf("%d",&n);
46         for (int i = 0;i < n;i++)
47         {
48             scanf("%lf%lf",&tx,&ty);
49             p[i] = make_pair(tx,ty);
50         }

```

```

51     random_shuffle(p,p+n);
52     ans = CalcDis(p[0],p[1],p[2]);
53     build(3,ans/2.0);
54     for (int i = 3;i < n;i++)
55     {
56         x = (int)floor(2.0*p[i].first/ans);
57         y = (int)floor(2.0*p[i].second/ans);
58         tmp.clear();
59         for (int k = 0;k < 9;k++)
60         {
61             nx = x+step[k][0];
62             ny = y+step[k][1];
63             gird = make_pair(nx,ny);
64             if (g.find(gird) != g.end())
65             {
66                 op = g[gird].begin();
67                 ed = g[gird].end();
68                 for (it = op;it != ed;it++)
69                     tmp.push_back(*it);
70             }
71         }
72         flag = false;
73         for (int j = 0;j < tmp.size();j++)
74             for (int k = j+1;k < tmp.size();k++)
75             {
76                 nowans = CalcDis(p[i],tmp[j],tmp[k]);
77                 if (nowans < ans)
78                 {
79                     ans = nowans;
80                     flag = true;
81                 }
82             }
83         if (flag == true)
84             build(i+1,ans/2.0);
85         else
86             g[make_pair((int)floor(2.0*p[i].first/ans),(int)floor(2.0*p
87                 [i].second/ans))].push_back(p[i]);
88     }
89     printf("%.3f\n",ans);
90 }

```

6.17 多圆面积并

6.17.1 去重

有时候可能需要去掉不需要的圆

```

1  for (int i = 0; i < n; i++)
2  {
3      scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
4      del[i] = false;
5  }

```

```

6  for (int i = 0; i < n; i++)
7      if (del[i] == false)
8      {
9          if (c[i].r == 0.0) del[i] = true;
10         for (int j = 0; j < n; j++)
11             if (i != j)
12                 if (del[j] == false)
13                     if (cmp(Point(c[i].c,c[j].c).Len()+c[i].r,c[j].r) <= 0)
14                         del[i] = true;
15     }
16 tn = n;
17 n = 0;
18 for (int i = 0; i < tn; i++)
19     if (del[i] == false)
20         c[n++] = c[i];

```

6.17.2 圆并

$ans[i]$ 表示被覆盖 i 次的面积

```

1  const double pi = acos(-1.0);
2  const double eps = 1e-8;
3  struct Point
4  {
5      double x,y;
6      Point(){}
7      Point(double _x,double _y)
8      {
9          x = _x;
10         y = _y;
11     }
12     double Length()
13     {
14         return sqrt(x*x+y*y);
15     }
16 };
17 struct Circle
18 {
19     Point c;
20     double r;
21 };
22 struct Event
23 {
24     double tim;
25     int typ;
26     Event(){}
27     Event(double _tim,int _typ)
28     {
29         tim = _tim;
30         typ = _typ;
31     }
32 };

```



```

33
34 int cmp(const double& a,const double& b)
35 {
36     if (fabs(a-b) < eps) return 0;
37     if (a < b) return -1;
38     return 1;
39 }
40
41 bool Eventcmp(const Event& a,const Event& b)
42 {
43     return cmp(a.tim,b.tim) < 0;
44 }
45
46 double Area(double theta,double r)
47 {
48     return 0.5*r*r*(theta-sin(theta));
49 }
50
51 double xmult(Point a,Point b)
52 {
53     return a.x*b.y-a.y*b.x;
54 }
55
56 int n,cur,tote;
57 Circle c[1000];
58 double ans[1001],pre[1001],AB,AC,BC,theta,fai,a0,a1;
59 Event e[4000];
60 Point lab;
61
62 int main()
63 {
64     while (scanf("%d",&n) != EOF)
65     {
66         for (int i = 0;i < n;i++)
67             scanf("%lf%lf%lf",&c[i].c.x,&c[i].c.y,&c[i].r);
68         for (int i = 1;i <= n;i++)
69             ans[i] = 0.0;
70         for (int i = 0;i < n;i++)
71         {
72             tote = 0;
73             e[tote++] = Event(-pi,1);
74             e[tote++] = Event(pi,-1);
75             for (int j = 0;j < n;j++)
76                 if (j != i)
77                 {
78                     lab = Point(c[j].c.x-c[i].c.x,c[j].c.y-c[i].c.y);
79                     AB = lab.Length();
80                     AC = c[i].r;
81                     BC = c[j].r;
82                     if (cmp(AB+AC,BC) <= 0)
83                     {

```

```

84         e[tote++] = Event(-pi,1);
85         e[tote++] = Event(pi,-1);
86         continue;
87     }
88     if (cmp(AB+BC,AC) <= 0) continue;
89     if (cmp(AB,AC+BC) > 0) continue;
90     theta = atan2(lab.y,lab.x);
91     fai = acos((AC*AC+AB*AB-BC*BC)/(2.0*AC*AB));
92     a0 = theta-fai;
93     if (cmp(a0,-pi) < 0) a0 += 2*pi;
94     a1 = theta+fai;
95     if (cmp(a1,pi) > 0) a1 -= 2*pi;
96     if (cmp(a0,a1) > 0)
97     {
98         e[tote++] = Event(a0,1);
99         e[tote++] = Event(pi,-1);
100        e[tote++] = Event(-pi,1);
101        e[tote++] = Event(a1,-1);
102    }
103    else
104    {
105        e[tote++] = Event(a0,1);
106        e[tote++] = Event(a1,-1);
107    }
108    }
109    sort(e,e+tote,Eventcmp);
110    cur = 0;
111    for (int j = 0;j < tote;j++)
112    {
113        if (cur != 0 && cmp(e[j].tim,pre[cur]) != 0)
114        {
115            ans[cur] += Area(e[j].tim-pre[cur],c[i].r);
116            ans[cur] += xmult(Point(c[i].c.x+c[i].r*cos(pre[cur]),c[i].c.y+c[i].r*sin(pre[cur])),
117                               Point(c[i].c.x+c[i].r*cos(e[j].tim),c[i].c.y+c[i].r*sin(e[j].tim)))/2.0;
118        }
119        cur += e[j].typ;
120        pre[cur] = e[j].tim;
121    }
122    }
123    for (int i = 1;i < n;i++)
124        ans[i] -= ans[i+1];
125    for (int i = 1;i <= n;i++)
126        printf("[%d] = %.3f\n",i,ans[i]);
127    }
128    return 0;
129 }

```

6.18 一个圆与多边形面积交

```

1 bool InCircle(Point a,double r)
2 {
3     return cmp(a.x*a.x+a.y*a.y,r*r) <= 0;
4     //这里判断的时候 EPS 一定不要太小!!
5 }
6
7 double CalcArea(Point a,Point b,double r)
8 {
9     Point p[4];
10    int tot = 0;
11    p[tot++] = a;
12
13    Point tv = Point(a,b);
14    Line tmp = Line(Point(0,0),Point(tv.y,-tv.x));
15    Point near = LineToLine(Line(a,b),tmp);
16    if (cmp(near.x*near.x+near.y*near.y,r*r) <= 0)
17    {
18        double A,B,C;
19        A = near.x*near.x+near.y*near.y;
20        C = r;
21        B = C*C-A;
22        double tvl = tv.x*tv.x+tv.y*tv.y;
23        double tmp = sqrt(B/tvl); //这样做只用一次开根
24        p[tot] = Point(near.x+tmp*tv.x,near.y+tmp*tv.y);
25        if (OnSeg(Line(a,b),p[tot]) == true) tot++;
26        p[tot] = Point(near.x-tmp*tv.x,near.y-tmp*tv.y);
27        if (OnSeg(Line(a,b),p[tot]) == true) tot++;
28    }
29    if (tot == 3)
30    {
31        if (cmp(Point(p[0],p[1]).Length(),Point(p[0],p[2]).Length()) >
32            0)
33            swap(p[1],p[2]);
34    }
35    p[tot++] = b;
36
37    double res = 0.0,theta,a0,a1,sgn;
38    for (int i = 0;i < tot-1;i++)
39    {
40        if (InCircle(p[i],r) == true && InCircle(p[i+1],r) == true)
41        {
42            res += 0.5*xmult(p[i],p[i+1]);
43        }
44        else
45        {
46            a0 = atan2(p[i+1].y,p[i+1].x);
47            a1 = atan2(p[i].y,p[i].x);
48            if (a0 < a1) a0 += 2*pi;
49            theta = a0-a1;
50            if (cmp(theta,pi) >= 0) theta = 2*pi-theta;
51            sgn = xmult(p[i],p[i+1])/2.0;

```

```

51     if (cmp(sgn,0) < 0) theta = -theta;
52     res += 0.5*r*r*theta;
53 }
54 }
55 return res;
56 }

```

调用

```

1 area2 = 0.0;
2 for (int i = 0; i < resn; i++) //遍历每条边, 按照逆时针
3     area2 += CalcArea(p[i], p[(i+1)%resn], r);

```

6.19 精度问题

6.19.1 浮点数为啥会有精度问题

浮点数 (以 C/C++ 为准), 一般用的较多的是 float、double。

	占字节数	数值范围	十进制精度位数
float	4	$-3.4e-38 \sim 3.4e38$	6 ~ 7
double	8	$-1.7e-308 \sim 1.7e308$	14 ~ 15

如果内存不是很紧张或者精度要求不是很低, 一般选用 double。14 位的精度 (是有效数字位, 不是小数点后的位数) 通常够用了。注意, 问题来了, 数据精度位数达到了 14 位, 但有些浮点运算的结果精度并达不到这么高, 可能准确的结果只有 10 ~ 12 位左右。那低几位呢? 自然就是不可预料的数字了。这给我们带来这样的问题: 即使是理论上相同的值, 由于是经过不同的运算过程得到的, 他们在低几位有可能 (一般来说都是) 是不同的。这种现象看似没太大的影响, 却会一种运算产生致命的影响: `==`。恩, 就是判断相等。注意, C/C++ 中浮点数的 `==` 需要完全一样才能返回 true。

6.19.2 eps

eps 缩写自 epsilon, 表示一个小量, 但这个量又要确保远大于浮点运算结果的不确定量。eps 最常见的取值是 $1e-8$ 左右。引入 eps 后, 我们判断两浮点数 a、b 相等的方式如下:

```

1 int sgn(double a){return a < -eps ? -1 : a < eps ? 0 : 1;}

```

这样, 我们才能把相差非常近的浮点数判为相等; 同时把确实相差较大 (差值大于 eps) 的数判为不相等。

养成好习惯, 尽量不要再对浮点数做 `==` 判断。

6.19.3 eps 带来的函数越界

如果 `sqrt(a)`, `asin(a)`, `acos(a)` 中的 a 是你自己算出来并传进来的, 那就得小心了。

如果 a 本来应该是 0 的, 由于浮点误差, 可能实际是一个绝对值很小的负数 (比如 $-1e-12$), 这样 `sqrt(a)` 应得 0 的, 直接因 a 不在定义域而出错。

类似地, 如果 a 本来应该是 ± 1 , 则 `asin(a)`、`acos(a)` 也有可能出错。

因此, 对于此种函数, 必需事先对 a 进行校正。

6.19.4 输出陷阱 I

现在考虑一种情况，题目要求输出保留两位小数。有个 case 的正确答案的精确值是 0.005，按理应该输出 0.01，但你的结果可能是 0.005000000001(恭喜)，也有可能是 0.004999999999(悲剧)，如果按照 `printf("%.2lf", a)` 输出，那你的遭遇将和括号里的字相同。解决办法是，如果 a 为正，则输出 $a + \text{eps}$ ，否则输出 $a - \text{eps}$

6.19.5 输出陷阱 II

ICPC 题目输出有个不成文的规定 (有时也成文)，不要输出：-0.000

那我们首先要弄清，什么时候按 `printf("%.3lf", a)` 输出会出现这个结果。

直接给出结果好了： $a \in (-0.000499999\ldots, -0.000\ldots1)$

所以，如果你发现 a 落在这个范围内，请直接输出 0.000。更保险的做法是用 `sprintf` 直接判断输出结果是不是 -0.000 再予处理。

6.19.6 范围越界

请注意，虽然 `double` 可以表示的数的范围很大，却不是无穷大，上面说过最大是 $1e308$ 。所以有些时候你得小心了，比如做连乘的时候，必要的时候要换成对数的和。

6.19.7 关于 set

经观察，`set` 不是通过 `==` 来判断相等的，是通过 `<` 来进行的，具体说来，只要 $a < b$ 和 $b < a$ 都不成立，就认为 a 和 b 相等，可以发现，如果将小于定义成：

```
1 | bool operator < (const Dat dat) const {return val < dat.val - eps;}
```

就可以解决问题了。(基本类型不能重载运算符，所以封装了下)

6.19.8 输入值波动过大

这种情况不常见，不过可以帮助你更熟悉 `eps`。假如一道题输入说，给一个浮点数 a ， $1e-20 < a < 1e20$ 。那你还敢用 $1e-8$ 做 `eps` 么？合理的做法是把 `eps` 按照输入规模缩放到合适大小。

6.19.9 一些建议

容易产生较大浮点误差的函数有 `asin`、`acos`。欢迎尽量使用 `atan2`。

另外，如果数据明确说明是整数，而且范围不大的话，使用 `int` 或者 `long long` 代替 `double` 都是极佳选择，因为就不存在浮点误差了

7 搜索

7.1 Dancing Links

7.1.1 估价函数

```

1 int h()
2 {
3     bool vis[100];
4     memset(vis,false,sizeof(vis));
5     int i,j,k,res=0,mi,col;
6     while(1)
7     {
8         mi=inf;
9         for(i=R[head]; i!=head&&i<=2*n; i=R[i])
10            if(mi>nk[i]&&!vis[i])
11            {
12                mi=nk[i];
13                col=i;
14            }
15        if(mi==inf)
16            break;
17        res++;
18        vis[col]=true;
19        for(j=D[col]; j!=col; j=D[j])
20            for(k=R[j]; k!=j; k=R[k])
21            {
22                if(C[k]>2*n)
23                    continue;
24                vis[C[k]]=true;
25            }
26    }
27    return res;
28 }
```

7.1.2 DLX

```

1 void remove1(int col)
2 {
3     int i,j;
4     L[R[col]]=L[col];
5     R[L[col]]=R[col];
6     for(i=D[col];i!=col;i=D[i])
7     {
8         L[R[i]]=L[i];
9         R[L[i]]=R[i];
10    }
11 }
12 void remove2(int col)
13 {
14     int i,j;
15     L[R[col]]=L[col];
```

```

16  R[L[col]]=R[col];
17  for(i=D[col];i!=col;i=D[i])
18  {
19      for(j=R[i];j!=i;j=R[j])
20      {
21          U[D[j]]=U[j];
22          D[U[j]]=D[j];
23          --nk[C[j]];
24      }
25  }
26 }
27 void resume1(int col)
28 {
29     int i,j;
30     for(i=U[col];i!=col;i=U[i])
31     {
32         L[R[i]]=i;
33         R[L[i]]=i;
34     }
35     L[R[col]]=col;
36     R[L[col]]=col;
37 }
38 void resume2(int col)
39 {
40     int i,j;
41     for(i=U[col];i!=col;i=U[i])
42     {
43         for(j=L[i];j!=i;j=L[j])
44         {
45             ++nk[C[j]];
46             U[D[j]]=j;
47             D[U[j]]=j;
48         }
49     }
50     L[R[col]]=col;
51     R[L[col]]=col;
52 }
53 int h()
54 {
55     bool vis[100];
56     memset(vis,false,sizeof(vis));
57     int i,j,k,res=0,mi,col;
58     while(1)
59     {
60         mi=inf;
61         for(i=R[head];i!=head&&i<=2*n;i=R[i])
62             if(mi>nk[i]&&!vis[i])
63             {
64                 mi=nk[i];
65                 col=i;
66             }

```

```

67     if(mi==inf)
68         break;
69     res++;vis[col]=true;
70     for(j=D[col];j!=col;j=D[j])
71         for(k=R[j];k!=j;k=R[k])
72         {
73             if(C[k]>2*n)
74                 continue;
75             vis[C[k]]=true;
76         }
77     }
78     return res;
79 }
80 bool DLX(int d,int deep)
81 {
82     if(d+h(>deep) return false;
83     if(R[head]==head || R[head]>2*n)
84         return true;
85     if(d>=deep)
86         return false;
87     int col,ma=inf;
88     int i,j;
89     for(i=R[head];i!=head&&i<=2*n;i=R[i])
90         if(nk[i]<ma)
91         {
92             col=i;
93             ma=nk[i];
94         }
95     remove1(col);
96     for(i=D[col];i!=col;i=D[i])
97     {
98         int flag=1;
99         for(j=R[i];j=R[j])
100         {
101             if(j==R[i]&&!flag)
102                 break;
103             U[D[j]]=U[j];
104             D[U[j]]=D[j];
105             if(C[j]>2*n)
106                 remove2(C[j]);
107             else
108                 remove1(C[j]);
109             flag=0;
110         }
111         if(DLX(d+1,deep))
112             return true;
113         flag=1;
114         for(j=L[i];j=L[j])
115         {
116             if(j==L[i]&&!flag)
117                 break;

```



```
118     if(C[j]>2*n)
119         resume2(C[j]);
120     else
121         resume1(C[j]);
122     U[D[j]]=j;
123     D[U[j]]=j;
124     flag=0;
125 }
126 }
127 resume1(col);
128 return false;
129 }
```

8 动态规划

8.1 斜率优化

```

1  #include<cstdio>
2  #include<algorithm>
3  using namespace std;
4  int a[1000],sum[1001],dp[1000][1000];
5  int deque[1000];
6  const int inf=0x7fffffff;
7  int N,s,t;
8  int calc(int i,int l,int j)//决策值计算
9  {
10     return dp[j][l-1]-(sum[i]-sum[j])*(sum[N]-sum[i]);
11 }
12 bool check(int i,int l)//尾端判断
13 {
14     int k1=deque[t-1],k2=deque[t-2];
15     return (long long)(dp[k1][l]-dp[k2][l])*(sum[i]-sum[k1])>(long
        long)(dp[i][l]-dp[k1][l])*(sum[k1]-sum[k2]);
16 }
17 int main()
18 {
19     int n,m;
20     while (scanf("%d%d",&n,&m),n)
21     {
22         for (int i=0; i<n; i++)
23             scanf("%d",&a[i]);
24         N=n;
25         sum[0]=0;
26         for (int i=0; i<n; i++)
27             sum[i+1]=sum[i]+a[i];
28         dp[0][0]=0;
29         for (int i=0; i<n; i++)
30             for (int j=i+1; j<n; j++)
31                 dp[0][0]+=a[i]*a[j];
32         for (int i=1; i<n; i++)
33             dp[i][0]=inf;
34         for (int i=1; i<n; i++)
35         {
36             dp[i][1]=inf;
37             for (int j=0; j<i; j++)
38                 dp[i][1]=min(dp[i][1],calc(i,1,j));
39         }
40         for (int l=2; l<=m; l++)
41         {
42             s=t=0;//双端队列清空
43             for (int i=l; i<n; i++)
44             {
45                 while (t-s>1 && check(i-1,l-1)) t--;
46                 deque[t++]=i-1;//决策加入

```

```

47     while (t-s>1 && calc(i,l,deque[s])>calc(i,l,deque[s+1])) s
        ++;
48     dp[i][l]=calc(i,l,deque[s]);
49 }
50 }
51 int ans=0x7fffffff;
52 for (int i=m; i<n; i++)
53     ans=min(ans,dp[i][m]);
54 printf("%d\n",ans);
55 }
56 return 0;
57 }

```

8.2 RMQ 二版

```

1 void init()
2 {
3     int i,j;
4     int n=N,k=1,l=0;
5     for (i=0; i<n; i++)
6     {
7         f[i][0]=ele[i].num;
8         if (i+1>k*2)
9         {
10            k*=2;
11            l++;
12        }
13        lent[i+1]=l;
14    }
15    for (j=1; (1<<j)-1<n; j++)
16        for (i=0; i+(1<<j)-1<n; i++)
17            f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
18 }
19 int fint(int x,int y)
20 {
21     int k=lent[y-x+1];
22     return max(f[x][k],f[y-(1<<k)+1][k]);
23 }

```

8.3 二维 LIS

```

1 #include<cstdio>
2 #include<map>
3 using namespace std;
4 map<int,int> mp[100001];
5 bool check(int idx,int x,int y)
6 {
7     if (!idx) return 1;
8     if (mp[idx].begin()->first>=x) return 0;
9     map<int,int> ::iterator it=mp[idx].lower_bound(x);
10    it--;
11    if (it->second<y) return 1;

```

```

12     else return 0;
13 }
14 int main()
15 {
16     int n;
17     scanf("%d",&n);
18     int l=0,r=0;
19     for (int i=0;i<n;i++)
20     {
21         int x,y;
22         scanf("%d%d",&x,&y);
23         int tl=l,tr=r;
24         while (tl<tr)
25         {
26             int mid=(tl+tr+1)/2;
27             if (check(mid,x,y))
28                 tl=mid;
29             else
30                 tr=mid-1;
31         }
32         if (tl==r) r++;
33         int idx=tl+1;
34         map<int,int> ::iterator itl=mp[idx].lower_bound(x),itr=itl;
35         while (itr!=mp[idx].end() && itr->second>y) itr++;
36         if (mp[idx].find(x)!=mp[idx].end())
37             y=min(y,mp[idx][x]);
38         if (itl!=itr) mp[idx].erase(itl,itr);
39         if (mp[idx].find(x)==mp[idx].end() || mp[idx][x]>y)
40             mp[idx][x]=y;
41     }
42     printf("%d\n",r);
43     return 0;
44 }

```

8.4 插头 DP

Tower Defence 独立插头 + 构造解

构造解的时候保存的是在 hash_map 的 ele 数组的下标位置
 没想清楚千万别去写

```

1  int bit[12];
2
3  inline int getbit(long long sta,int pos)
4  {
5      return sta/bit[pos]%bit[1];
6  }
7
8  inline long long setbit(long long sta,int pos,int val)
9  {
10     return sta/bit[pos+1]*bit[pos+1]+val*bit[pos]+sta%bit[pos];
11 }
12

```

```

13 int n,m,mp[30][10];
14 char buf[30][10];
15 hash_map dp[2];
16 bool flag;
17 int key,val,upd,l,u,res,msk,cov,now,pr,resnow,resmsk,pru;
18 int w[15],s[15],top;
19 int pre[210][10007],preuse[210][10007];
20
21 void decode(int msk,int& key,int& cov)
22 {
23     int tmp;
24     key = cov = 0;
25     for (int i = 0; i < m+1; i++)
26     {
27         tmp = getbit(msk,i);
28         if (tmp > 0)
29         {
30             key = setbit(key,i,tmp-1);
31             cov = setbit(cov,i,1);
32         }
33     }
34 }
35
36 int encode(int key,int cov)
37 {
38     int res = 0,tmp;
39     for (int i = 0; i < m+1; i++)
40     {
41         tmp = getbit(cov,i);
42         if (tmp > 0)
43         {
44             tmp = getbit(key,i);
45             res = setbit(res,i,tmp+1);
46         }
47     }
48     return res;
49 }
50
51 void update(int a,int key,int cov,int val)
52 {
53     int msk = encode(key,cov);
54     int pos;
55     if (dp[a][msk] < val)
56     {
57         dp[a][msk] = val;
58         pos = dp[a].fint(msk);
59         pre[now][pos] = pr;
60         preuse[now][pos] = pru;
61     }
62 }
63

```

```

64 int count3(int sta)
65 {
66     int res = 0;
67     for (int i = 0; i < m+1; i++)
68         if (getbit(sta,i) == 3)
69             res++;
70     return res;
71 }
72
73 void expand(int sta)
74 {
75     top = 0;
76     for (int i = 0; i < m+1; i++)
77         if (getbit(sta,i) == 1)
78             s[top++] = i;
79         else if (getbit(sta,i) == 2)
80         {
81             w[s[top-1]] = i;
82             w[i] = s[top-1];
83             top--;
84         }
85 }
86
87 int main()
88 {
89     //freopen("TD.in","r",stdin);
90     //freopen("TDM.out","w",stdout);
91     bit[0] = 1;
92     for (int i = 1; i < 12; i++)    bit[i] = bit[i-1]*5;
93     int t;
94     scanf("%d",&t);
95     dp[0].init();
96     dp[1].init();
97     for (int ft = 1; ft <= t; ft++)
98     {
99         scanf("%d%d",&n,&m);
100         res = 0;
101         memset(mp,0,sizeof(mp));
102         memset(pre,0,sizeof(pre));
103         memset(preuse,0,sizeof(preuse));
104         for (int i = 0; i < n; i++)
105         {
106             scanf("%s",buf[i]);
107             for (int j = 0; j < m; j++)
108                 if (buf[i][j] == '.')
109                     mp[i][j] = 1;
110                 else if (buf[i][j] != 'B')
111                     mp[i][j] = 2;
112         }
113         dp[0].clear();
114         dp[1].clear();

```

```

115     flag = 0;
116     dp[flag][0] = 0;
117     int res = 0;
118     now = 0;
119     for (int i = 0; i < n; i++)
120     {
121         for (int j = 0; j < m; j++)
122         {
123             dp[!flag].clear();
124             for (int k = 0; k < dp[flag].N; k++)
125             {
126                 msk = dp[flag].ele[k].key;
127                 pr = k;
128                 val = dp[flag].ele[k].val;
129                 decode(msk, key, cov);
130                 l = getbit(key, j);
131                 u = getbit(key, j+1);
132                 if (mp[i][j] == 0) //是障碍
133                 {
134                     if (l == 0 && u == 0)
135                     {
136                         pru = 0;
137                         update(!flag, key, setbit(setbit(cov, j, 0), j+1, 0), val);
138                     }
139                 }
140                 else
141                 {
142                     if (mp[i][j] == 1 && l == 0 && u == 0) //不要插头
143                     {
144                         pru = 1;
145                         update(!flag, key, setbit(setbit(cov, j, 0), j+1, 0), val);
146                     }
147                     if (getbit(cov, j) == 1 && l == 0) continue; //不可以在
                        这里搞插头
148                     if (getbit(cov, j+1) == 1 && u == 0) continue;
149                     cov = setbit(setbit(cov, j, 1), j+1, 1); //更新覆盖情况
150                     upd = setbit(setbit(key, j, 0), j+1, 0);
151                     pru = 2;
152                     if (mp[i][j] == 2)
153                     {
154                         if (l == 0 && u == 0)
155                         {
156                             if (count3(key) < 2) //可以新建独立插头
157                             {
158                                 if (mp[i][j+1] != 0)
159                                 update(!flag, setbit(setbit(key, j, 0), j+1, 3), cov,
                                    val+1);
160                                 if (mp[i+1][j] != 0)
161                                 update(!flag, setbit(setbit(key, j, 3), j+1, 0), cov,
                                    val+1);
162                             }

```

```

163     }
164     else if (l == 0 || u == 0)
165     {
166         if (l+u < 3 && count3(key) < 2)//可以用一个独立插头来
            结束这条路径
167         {
168             expand(key);
169             if (l > 0)
170                 update(!flag, setbit(upd, w[j], 3), cov, val+1);
171             else
172                 update(!flag, setbit(upd, w[j+1], 3), cov, val+1);
173         }
174         else if (l+u == 3 && upd == 0)//路径的一端
175         {
176             if (res < val+1)
177             {
178                 res = val+1;
179                 resnow = now-1;
180                 resmsk = k;
181             }
182         }
183     }
184 }
185 else if (l == 0 && u == 0)
186 {
187     if (mp[i][j+1] != 0 && mp[i+1][j] != 0)//可以新建插头
188         update(!flag, setbit(setbit(key, j, 1), j+1, 2), cov, val
            +1);
189 }
190 else if (l == 0 || u == 0)
191 {
192     if (mp[i][j+1] != 0)//可以延续插头
193         update(!flag, setbit(upd, j+1, l+u), cov, val+1);
194     if (mp[i+1][j] != 0)//可以延续插头
195         update(!flag, setbit(upd, j, l+u), cov, val+1);
196 }
197 else if (l == u)
198 {
199     if (l < 3) //合并两个相同的括号
200     {
201         expand(key);
202         if (l == 1)
203             update(!flag, setbit(upd, w[j+1], 1), cov, val+1);
204         else
205             update(!flag, setbit(upd, w[j], 2), cov, val+1);
206     }
207     else if (upd == 0)//合并两个独立插头
208     {
209         if (res < val+1)
210         {
211             res = val+1;

```



```

212         resnow = now-1;
213         resmsk = k;
214     }
215 }
216 }
217 else if (l == 3 || u == 3) //合并独立插头与括号
218 {
219     expand(key);
220     if (l == 3)
221         update(!flag, setbit(upd, w[j+1], 3), cov, val+1);
222     else
223         update(!flag, setbit(upd, w[j], 3), cov, val+1);
224 }
225 else if (l == 2 || u == 1) //合并)(
226     update(!flag, upd, cov, val+1);
227 }
228 }
229 flag = !flag;
230 now++;
231 }
232 if (i+1 == n)    break;
233
234 dp[!flag].clear();
235 for (int k = 0; k < dp[flag].N; k++)
236 {
237     msk = dp[flag].ele[k].key;
238     pr = k;
239     val = dp[flag].ele[k].val;
240     pru = 0;
241     decode(msk, key, cov);
242     update(!flag, key*bit[1], cov*bit[1], val);
243 }
244 now++;
245 flag = !flag;
246 }
247
248 printf("Case_%d:_%d\n", ft, res);
249 for (int i = resnow; i >= 0; i--)
250 {
251     if (preuse[i][resmsk] == 1)
252         buf[i/(m+1)][i%(m+1)] = 'W';
253     resmsk = pre[i][resmsk];
254 }
255 for (int i = 0; i < n; i++)
256     printf("%s\n", buf[i]);
257 printf("\n");
258 }
259 return 0;
260 }

```

9 杂物

9.1 高精度数

支持乘以整数和加法。

```

1 struct BigInt
2 {
3     const static int mod = 1000000000;
4     int a[600], len;
5     BigInt (){}
6     BigInt (int v)
7     {
8         len = 0;
9         do
10        {
11            a[len++] = v%mod;
12            v /= mod;
13        }while(v);
14    }
15     BigInt operator *(const int& b) const
16     {
17         BigInt res;
18         res.len = len;
19         for (int i = 0; i <= len; ++i)
20             res.a[i] = 0;
21         for (int i = 0; i < len; ++i)
22         {
23             res.a[i] += a[i]*b;
24             res.a[i+1] += res.a[i]/mod;
25             res.a[i] %= mod;
26         }
27         if (res.a[len] > 0) res.len++;
28         return res;
29     }
30     BigInt operator +(const BigInt& b) const
31     {
32         BigInt res;
33         res.len = max(len, b.len);
34         for (int i = 0; i <= res.len; ++i)
35             res.a[i] = 0;
36         for (int i = 0; i < res.len; ++i)
37         {
38             res.a[i] += ((i < len)?a[i]:0)+((i < b.len)?b.a[i]:0);
39             res.a[i+1] += res.a[i]/mod;
40             res.a[i] %= mod;
41         }
42         if (res.a[res.len] > 0) res.len++;
43         return res;
44     }
45     void output()

```

```

46     {
47         printf("%d",a[len-1]);
48         for (int i = len-2; i >= 0; --i)
49             printf("%08d",a[i]);
50         printf("\n");
51     }
52 };

```

9.2 整数外挂

```

1  int wg;
2  char ch;
3  bool ng;
4
5  inline int readint()
6  {
7      ch = getchar();
8      while (ch != '-' && (ch < '0' || ch > '9')) ch = getchar();
9      if (ch == '-')
10     {
11         ng = true;
12         ch = getchar();
13     }
14     else
15         ng = false;
16     wg = ch-'0';
17     ch = getchar();
18     while (ch >= '0' && ch <= '9')
19     {
20         wg = wg*10+ch-'0';
21         ch = getchar();
22     }
23     if (ng == true) wg = -wg;
24     return wg;
25 }

```

9.3 Java

9.3.1 文件操作

```

1  import java.io.*;
2  import java.util.*;
3  import java.math.*;
4  import java.text.*;
5
6  public class Main
7  {
8
9      public static void main(String args[]) throws
10         FileNotFoundException, IOException
11     {
12         Scanner sc = new Scanner(new FileReader("a.in"));

```

```

12     PrintWriter pw = new PrintWriter(new FileWriter("a.out"));
13     int n,m;
14     n=sc.nextInt();//读入下一个INT
15     m=sc.nextInt();
16
17     for(ci=1; ci<=c; ++ci)
18     {
19         pw.println("Case_"+ci+":_easy_for_output");
20     }
21
22     pw.close();//关闭流并释放, 这个很重要, 否则是没有输出的
23     sc.close();//关闭流并释放
24 }
25 }

```

9.3.2 优先队列

```

1 PriorityQueue queue = new PriorityQueue( 1, new Comparator()
2 {
3     public int compare( Point a, Point b )
4     {
5         if( a.x < b.x || a.x == b.x && a.y < b.y )
6             return -1;
7         else if( a.x == b.x && a.y == b.y )
8             return 0;
9         else
10            return 1;
11     }
12 });

```

9.3.3 Map

```

1 Map map = new HashMap();
2 map.put("sa","dd");
3 String str = map.get("sa").toString;
4
5 for(Object obj : map.keySet()){
6     Object value = map.get(obj );
7 }

```

9.3.4 sort

```

1 static class cmp implements Comparator
2 {
3     public int compare(Object o1,Object o2)
4     {
5         BigInteger b1=(BigInteger)o1;
6         BigInteger b2=(BigInteger)o2;
7         return b1.compareTo(b2);
8     }
9 }
10 public static void main(String[] args) throws IOException
11 {

```

```

12 Scanner cin = new Scanner(System.in);
13 int n;
14 n=cin.nextInt();
15 BigInteger[] seg = new BigInteger[n];
16 for (int i=0;i<n;i++)
17     seg[i]=cin.nextBigInteger();
18 Arrays.sort(seg,new cmp());
19 }

```

9.4 hashmap

```

1 struct hash_map
2 {
3     const static int mod=10007;
4     int head[mod];
5     struct hash_tables
6     {
7         int key;
8         int val;
9         int next;
10    } ele[10007];
11    int N;
12    int getHash(int x)
13    {
14        return x%mod; //小心负数
15    }
16    void init()
17    {
18        memset(head,255,sizeof(head));
19        N=0;
20    }
21    void clear()
22    {
23        for (int i = 0; i < N; i++)
24            head[getHash(ele[i].key)] = -1;
25        N = 0;
26    }
27    int fint(int x)
28    {
29        for (int i=head[getHash(x)]; i!=-1; i=ele[i].next)
30            if (ele[i].key==x) return i;
31        return -1;
32    }
33    void insert(int x)
34    {
35        int tmp=getHash(x);
36        ele[N].key=x;
37        ele[N].val=0;
38        ele[N].next=head[tmp];
39        head[tmp]=N++;
40    }
41    int& operator [] (int x)

```

```

42     {
43         int tmp=fint(x);
44         if (tmp==−1)
45         {
46             insert(x);
47             return ele[N−1].val;
48         }
49         else
50             return ele[tmp].val;
51     }
52 };

```

9.5 C++&STL 常用函数

9.5.1 lower_bound/upper_bound

不解释

```

1  iterator lower_bound(const key_type &key )
2  \\返回一个迭代器, 指向键值 >= key 的第一个元素。
3  iterator upper_bound(const key_type &key )
4  \\返回一个迭代器, 指向键值 > key 的第一个元素。
5
6  #include <iostream>
7  #include <algorithm>
8  #include <vector>
9  using namespace std;
10
11 int main () {
12     int myints[] = {10,20,30,30,20,10,10,20};
13     vector<int> v(myints,myints+8);
14     // 10 20 30 30 20 10 10 20
15     vector<int>::iterator low,up;
16
17     sort (v.begin(), v.end());
18     // 10 10 10 20 20 20 30 30
19
20     low=lower_bound (v.begin(), v.end(), 20);
21     // 10 10 10 20 20 20 30 30
22     //           ^
23     up= upper_bound (v.begin(), v.end(), 20);
24     // 10 10 10 20 20 20 30 30
25     //           ^
26
27     cout << "lower_bound_at_position" << int(low− v.begin()) << endl
28         ;
29     cout << "upper_bound_at_position" << int(up − v.begin()) << endl
30         ;
31     return 0;
32 }

```

Output:

```
1 | lower_bound at position 3
2 | upper_bound at position 6
```

9.5.2 rotate

把数组后一半搬到前面

```
1 | template <class ForwardIterator>
2 |     void rotate ( ForwardIterator first, ForwardIterator middle,
3 |                  ForwardIterator last );
```

9.5.3 nth_element

```
1 | template <class RandomAccessIterator>
2 |     void nth_element ( RandomAccessIterator first,
3 |                      RandomAccessIterator nth,
4 |                      RandomAccessIterator last );
5 | template <class RandomAccessIterator, class Compare>
6 |     void nth_element ( RandomAccessIterator first,
7 |                      RandomAccessIterator nth,
8 |                      RandomAccessIterator last, Compare comp );
```

9.5.4 bitset

取用

```
1 | bitset<4> mybits;
2 |
3 | mybits[1]=1;           // 0010
4 | mybits[2]=mybits[1];   // 0110
```

翻转

```
1 | bitset<4> mybits (string("0001"));
2 |
3 | cout << mybits.flip(2) << endl;      // 0101
4 | cout << mybits.flip() << endl;       // 1010
```

运算

```
1 | bitset<4> first (string("1001"));
2 | bitset<4> second (string("0011"));
3 |
4 | cout << (first^=second) << endl;      // 1010 (XOR,assign)
5 | cout << (first&=second) << endl;      // 0010 (AND,assign)
6 | cout << (first|=second) << endl;      // 0011 (OR,assign)
7 |
8 | cout << (first<<=2) << endl;           // 1100 (SHL,assign)
```

```

9  cout << (first>>=1) << endl;           // 0110 (SHR,assign)
10
11 cout << (~second) << endl;             // 1100 (NOT)
12 cout << (second<<1) << endl;           // 0110 (SHL)
13 cout << (second>>1) << endl;           // 0001 (SHR)
14
15 cout << (first==second) << endl;        // false (0110==0011)
16 cout << (first!=second) << endl;        // true  (0110!=0011)
17
18 cout << (first&second) << endl;         // 0010
19 cout << (first|second) << endl;         // 0111
20 cout << (first^second) << endl;         // 0101

```

9.5.5 multimap

遍历

```

1  multimap<char,int> mymm;
2  multimap<char,int>::iterator it;
3  char c;
4
5  mymm.insert(pair<char,int>('x',50));
6  mymm.insert(pair<char,int>('y',100));
7  mymm.insert(pair<char,int>('y',150));
8  mymm.insert(pair<char,int>('y',200));
9  mymm.insert(pair<char,int>('z',250));
10 mymm.insert(pair<char,int>('z',300));
11
12 for (c='x'; c<='z'; c++)
13 {
14     cout << "There are " << (int)mymm.count(c);
15     cout << " elements with key " << c << ":";
16     for (it=mymm.equal_range(c).first; it!=mymm.equal_range(c).second
17           ; ++it)
18         cout << " " << (*it).second;
19     cout << endl;
20 }
21 /*
22 Output:
23 There are 1 elements with key x: 50
24 There are 3 elements with key y: 100 150 200
25 There are 2 elements with key z: 250 300
26 */

```

二分查找

```

1  multimap<char,int> mymultimap;
2  multimap<char,int>::iterator it,itlow,itup;
3
4  mymultimap.insert(pair<char,int>('a',10));

```



```

5 mymultimap.insert(pair<char,int>('b',121));
6 mymultimap.insert(pair<char,int>('c',1001));
7 mymultimap.insert(pair<char,int>('c',2002));
8 mymultimap.insert(pair<char,int>('d',11011));
9 mymultimap.insert(pair<char,int>('e',44));
10
11 itlow=mymultimap.lower_bound ('b'); // itlow points to b
12 itup=mymultimap.upper_bound ('d'); // itup points to e (not d)
13
14 // print range [itlow,itup):
15 for ( it=itlow ; it != itup; it++ )
16     cout << (*it).first << "=>" << (*it).second << endl;
17
18 /*
19 Output:
20
21 b => 121
22 c => 1001
23 c => 2002
24 d => 11011
25 */

```

删除

```

1 multimap<char,int> mymultimap;
2 multimap<char,int>::iterator it;
3
4 // insert some values:
5 mymultimap.insert(pair<char,int>('a',10));
6 mymultimap.insert(pair<char,int>('b',20));
7 mymultimap.insert(pair<char,int>('b',30));
8 mymultimap.insert(pair<char,int>('c',40));
9 mymultimap.insert(pair<char,int>('d',50));
10 mymultimap.insert(pair<char,int>('d',60));
11 mymultimap.insert(pair<char,int>('e',70));
12 mymultimap.insert(pair<char,int>('f',80));
13
14 it=mymultimap.find('b');
15 mymultimap.erase (it);
16 // erasing by iterator (1 element)
17
18 mymultimap.erase ('d');
19 // erasing by key (2 elements)
20
21 it=mymultimap.find ('e');
22 mymultimap.erase ( it, mymultimap.end() );
23 // erasing by range
24
25 // show content:
26 for ( it=mymultimap.begin() ; it != mymultimap.end(); it++ )
27     cout << (*it).first << "=>" << (*it).second << endl;
28

```

```

29  /*
30  Output:
31
32  a => 10
33  b => 30
34  c => 40
35  */

```

9.6 位运算

9.6.1 基本操作

注意括号

功能	示例	位运算
去掉最后一位	(101101 → 10110)	x shr 1
在最后加一个 0	(101101 → 1011010)	x shl 1
在最后加一个 1	(101101 → 1011011)	x shl 1+1
把最后一位变成 1	(101100 → 101101)	x or 1
把最后一位变成 0	(101101 → 101100)	x or 1-1
最后一位取反	(101101 → 101100)	x xor 1
把右数第 k 位变成 1	(101001 → 101101, $k = 3$)	x or (1 shl (k-1))
把右数第 k 位变成 0	(101101 → 101001, $k = 3$)	x and not (1 shl (k-1))
右数第 k 位取反	(101001 → 101101, $k = 3$)	x xor (1 shl (k-1))
取末三位	(1101101 → 101)	x and 7
取末 k 位	(1101101 → 1101, $k = 5$)	x and (1 shl k-1)
取右数第 k 位	(1101101 → 1, $k = 4$)	x shr (k-1) and 1
把末 k 位变成 1	(101001 → 101111, $k = 4$)	x or (1 shl k-1)
末 k 位取反	(101001 → 100110, $k = 4$)	x xor (1 shl k-1)
把右边连续的 1 变成 0	(100101111 → 100100000)	x and (x+1)
把右起第一个 0 变成 1	(100101111 → 100111111)	x or (x+1)
把右边连续的 0 变成 1	(11011000 → 11011111)	x or (x-1)
取右边连续的 1	(100101111 → 1111)	(x xor (x+1)) shr 1
去掉右起第一个 1 的左边	(100101000 → 1000)	x and (x xor (x-1))

9.6.2 枚举长为 n 含 k 个 1 的 01 串

```

1  int n = 5, k = 3;
2  for (int s = (1 << k)-1, u = 1 << n; s < u;)
3  {
4      for (int i = 0; i < n; i++)
5          printf("%d", (((s >> (n-1-i)) & 1) == 1));
6      printf("\n");
7
8      int b = s & -s;
9      s = (s+b) | (((s^(s+b)) >> 2)/b);
10 }

```

9.7 其它

9.7.1 对跑脚本

```
1 while true; do
2     ./gen > input
3     ./sol < input > output.sol
4     ./bf < input > output.bf
5
6     diff output.sol output.bf
7     if [ $? -ne 0 ] ; then break; fi
8 done
```