ACM Template

Orz

Last build at September 29, 2012

Contents

1 To Do List

测试DC3模板。。
所有带*的内容。。。
可以从原来的模板里面继承一些好东西过来。
set,map,multiset等的搞基用法，以及注意事项。

2 注意事项

10^6数量级慎用后缀数组
TLE的时候要冷静哟。。
7k+的图计数（Wc2012的communication）
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≤来判断更新。。。

3 字符串处理

3.1 *AC自动机

3.1.1 指针

```
const int CHAR=26;
const int TOTLEN=500000;
const int MAXLEN=1000000;
struct Vertex
{
    Vertex *fail,*next[CHAR];
    Vertex(){}
    Vertex(bool flag)//为什么要这样写?
    {
        fail=0;
        memset(next,0,sizeof(next));
    }
};
int size;
Vertex vertex[TOTLEN+1];
void init()
{
    vertex[0]=Vertex(0);
    size=1;
}
void add(Vertex *pos,int cha)
{
    vertex[size]=Vertex(0);
    pos->next[cha]=&vertex[size++];
}
void add(vector<int> s)
{
    int l=s.size();
    Vertex *pos=&vertex[0];
    for (int i=0; i<l; i++)
    {
        if (pos->next[s[i]]==NULL)
            add(pos,s[i]);
        pos=pos->next[s[i]];
    }
}
void bfs()
{
    queue<Vertex *> que;
    Vertex *u=&vertex[0];
    for (int i=0; i<CHAR; i++)
        if (u->next[i]!=NULL)
        {
            que.push(u->next[i]);
            u->next[i]->fail=u;
        }
        else
            u->next[i]=u;
    u->fail=NULL;
    while (!que.empty())
    {
        u=que.front();
        que.pop();
        for (int i=0; i<CHAR; i++)
            if (u->next[i]!=NULL)
            {
                que.push(u->next[i]);
```

```
                u->next[i]->fail=u->fail->next[i];
            }
            else
                u->next[i]=u->fail->next[i];
    }
}
```

3.1.2 非指针

```
void build()
{
    queue<int> Q;
    for (int i = 0;i < 26;i++)
        if (next[root][i] == -1)
            next[root][i] = root;
        else
        {
            fail[next[root][i]] = root;
            Q.push(next[root][i]);
        }
    while (!Q.empty())
    {
        int now = Q.front();
        Q.pop();
        for (int i = 0;i < 26;i++)
            if (next[now][i] == -1)
                next[now][i] = next[fail[now]][i];
            else
            {
                fail[next[now][i]] = next[fail[now]][i];
                Q.push(next[now][i]);
            }
    }
}
```

3.2 后缀数组

3.2.1 DC3

所有下标都是0 n-1，height[0]无意义。

```
//所有相关数组都要开三倍
const int maxn = 300010;
# define F(x) ((x)/3+((x)%3==1?0:tb))
# define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn * 3], wb[maxn * 3], wv[maxn * 3], ws[maxn * 3];
int c0(int *r, int a, int b)
{
    return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
}
int c12(int k, int *r, int a, int b)
{
    if (k == 2) return r[a] < r[b] || r[a] == r[b] && c12(1, r, a + 1, b +
       1);
    else return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b + 1];
}
void sort(int *r, int *a, int *b, int n, int m)
{
    int i;
    for (i = 0; i < n; i++) wv[i] = r[a[i]];
    for (i = 0; i < m; i++) ws[i] = 0;
    for (i = 0; i < n; i++) ws[wv[i]]++;
    for (i = 1; i < m; i++) ws[i] += ws[i - 1];
    for (i = n - 1; i >= 0; i--) b[--ws[wv[i]]] = a[i];
    return;
```

```
}
void dc3(int *r, int *sa, int n, int m)
{
    int i, j, *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0,
        p;
    r[n] = r[n + 1] = 0;
    for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
    sort(r + 2, wa, wb, tbc, m);
    sort(r + 1, wb, wa, tbc, m);
    sort(r, wa, wb, tbc, m);
    for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
        rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
    if (p < tbc) dc3(rn, san, tbc, p);
    else for (i = 0; i < tbc; i++) san[rn[i]] = i;
    for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i] * 3;
    if (n % 3 == 1) wb[ta++] = n - 1;
    sort(r, wb, wa, ta, m);
    for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;
    for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
        sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
    for (; i < ta; p++) sa[p] = wa[i++];
    for (; j < tbc; p++) sa[p] = wb[j++];
}
//str和sa也要三倍
void da(int str[], int sa[], int rank[], int height[], int n, int m)
{
    for (int i = n; i < n * 3; i++)
        str[i] = 0;
    dc3 (str , sa , n + 1 , m);
    int i, j, k;
    for (i = 0; i < n; i++)
    {
        sa[i] = sa[i + 1];
        rank[sa[i]] = i;
    }
    for (i = 0, j = 0, k = 0; i < n; height[rank[i ++]] = k)
        if (rank[i] > 0)
            for (k ? k-- : 0 , j = sa[rank[i] - 1]; i + k < n && j + k < n
                &&
                     str[i + k] == str[j + k]; k ++) ;
}
```

3.2.2 DA

这份似乎就没啥要注意的了。

```
const int maxn = 200010;
int wx[maxn],wy[maxn],*x,*y,wss[maxn],wv[maxn];

bool cmp(int *r,int n,int a,int b,int l)
{
    return a+l<n && b+l<n && r[a]==r[b]&&r[a+l]==r[b+l];
}
void da(int str[],int sa[],int rank[],int height[],int n,int m)
{
    int *s = str;
    int *x=wx,*y=wy,*t,p;
    int i,j;
    for(i=0; i<m; i++)wss[i]=0;
    for(i=0; i<n; i++)wss[x[i]=s[i]]++;
    for(i=1; i<m; i++)wss[i]+=wss[i-1];
    for(i=n-1; i>=0; i--)sa[--wss[x[i]]]=i;
    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]; i+k < n && j+k < n && str[i+k]==str
                    [j+k]; k++);
33  }
```

3.3 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 + 1))
```

```
106             p=SAM_add(p,s[i] - 'a', i + 1);
107         else
108             p = p->next[s[i] - 'a'];
109     }
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,SAM_node[i].id,s[SAM_node[i].pos-SAM_node[
                i].fa->len-1]-'a',SAM_node[i].len-SAM_node[i].fa->len);
143         if (SAM_node[i].pos==SAM_node[i].len)
144         {
145             Sufpos[n-SAM_node[i].pos+1]=i;
146             ST_node[i].pos=n-SAM_node[i].pos+1;
147         }
148     }
149 }
150
151 int rank[MAXN],sa[MAXN+1];
152 int height[MAXN];
153 int L;
154 void ST_dfs(ST_Node *p)
155 {
156     if (p->pos!=-1)
157         sa[L++]=p->pos;
158     for (int i=0;i<CHAR;i++)
159         if (p->next[i])
160             ST_dfs(p->next[i]);
161 }
162 char s[MAXN+1];
163 int main()
164 {
165     gets(s);
166     ST_build(s);
167     L=0;
```

```
    ST_dfs(ST_root);
    int n=strlen(s);
    for (int i=0; i<n; i++)
        sa[i]=sa[i+1]-1;
    for (int i=0; i<n; i++)
        rank[sa[i]]=i;
    reverse(s,s+n);
    for (int i=0,j=0,k=0; i<n; height[rank[i++]]=k)
        if (rank[i])
            for (k?k--:0,j=sa[rank[i]-1]; s[i+k]==s[j+k]; k++);
}
```

3.3.1 例题

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;

const int CHAR = 26;
const int MAXN = 100000;

struct SAM_Node
{
    SAM_Node *fa,*next[CHAR];
    int len;
    int id;
    int mat[9];
    SAM_Node() {}
    SAM_Node(int _len)
    {
        fa = 0;
        len = _len;
        memset(mat,0,sizeof(mat));
        memset(next,0,sizeof(next));
    }
};
SAM_Node SAM_node[MAXN*2],*SAM_root,*SAM_last;
int SAM_size;
SAM_Node *newSAM_Node(int len)
{
    SAM_node[SAM_size] = SAM_Node(len);
    SAM_node[SAM_size].id = SAM_size;
    return &SAM_node[SAM_size++];
}
SAM_Node *newSAM_Node(SAM_Node *p)
{
    SAM_node[SAM_size] = *p;
    SAM_node[SAM_size].id = SAM_size;
    return &SAM_node[SAM_size++];
}
void SAM_init()
{
    SAM_size = 0;
    SAM_root = SAM_last = newSAM_Node(0);
}
void SAM_add(int x,int len)
{
    SAM_Node *p = SAM_last,*np = newSAM_Node(p->len+1);
    SAM_last = np;
    for (; p&&!p->next[x]; p=p->fa)
        p->next[x] = np;
    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         for (int j=0; j<n-1; j++)
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]);
```

```
    ans=M=0;
    SAM_build(s[0]);
    for (int j=1; j<n; j++)
    {
        int l=strlen(s[j]),len=0;
        SAM_Node *p=SAM_root;
        for (int i=0; i<l; i++)
        {
            if (p->next[getid(s[j][i])])
            {
                p=p->next[getid(s[j][i])];
                p->mat[j-1]=max(p->mat[j-1],++len);
            }
            else
            {
                while (p && !p->next[getid(s[j][i])])
                    p=p->fa;
                if (!p)
                {
                    p=SAM_root;
                    len=0;
                }
                else
                {
                    len=p->len+1;
                    p=p->next[getid(s[j][i])];
                }
                p->mat[j-1]=max(p->mat[j-1],len);
            }
        }
    }
    memset(head,-1,4*SAM_size);
    for (int i=1; i<SAM_size; i++)
        add_edge(SAM_node[i].fa->id,i);
    dfs(0);
    printf("%d\n",ans);
    return 0;
}
```

LCS2

```
#include <iostream>
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;

const int CHAR = 26;
const int MAXN = 100000;

struct SAM_Node
{
    SAM_Node *fa,*next[CHAR];
    int len;
    int id;
    int mat[9];
    SAM_Node() {}
    SAM_Node(int _len)
    {
        fa = 0;
        len = _len;
        memset(mat,0,sizeof(mat));
        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];
```

```
int M;
int n;
void add_edge(int u,int v)
{
    edge[M].to=v;
    edge[M].next=head[u];
    head[u]=M++;
}
void dfs(int u)
{
    for (int i=head[u]; i!=-1; i=edge[i].next)
    {
        int v=edge[i].to;
        /*for (int j=0; j<n; j++)
            SAM_node[v].mat[j]=max(SAM_node[v].mat[j],SAM_node[u].mat[j]);*/
        dfs(v);
        for (int j=0; j<n; j++)
            SAM_node[u].mat[j]=max(SAM_node[v].mat[j],SAM_node[u].mat[j]);
    }
    int tmp=SAM_node[u].len;
    for (int i=0; i<n; i++)
        tmp=min(tmp,SAM_node[u].mat[i]);
    ans=max(ans,tmp);
}
int main()
{
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    n=0;
    gets(s);
    SAM_build(s);
    while (gets(s))
    {
        int l=strlen(s),len=0;
        SAM_Node *p=SAM_root;
        for (int i=0; i<l; i++)
        {
            if (p->next[getid(s[i])])
            {
                p=p->next[getid(s[i])];
                p->mat[n]=max(p->mat[n],++len);
            }
            else
            {
                while (p && !p->next[getid(s[i])])
                    p=p->fa;
                if (!p)
                {
                    p=SAM_root;
                    len=0;
                }
                else
                {
                    len=p->len+1;
                    p=p->next[getid(s[i])];
                }
                p->mat[n]=max(p->mat[n],len);
            }
            //printf("%d %d %d\n",i,len,p->id);
        }
        n++;
    }
    memset(head,-1,4*SAM_size);
```

```
    for (int i=1; i<SAM_size; i++)
        add_edge(SAM_node[i].fa->id,i);
    dfs(0);
    printf("%d\n",ans);
    return 0;
}
```

3.4 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..
一个后缀最多能匹配B的前缀多长。

```
//自匹配过程
int j;
p [0] = j = -1;
for ( int i = 1; i < lb; i++)
{
    while (j >= 0 && b[j + 1] != b[i]) j = p[j];
    if (b[j + 1] == b[i]) j ++;
    p[i] = j;
}
//下面是匹配过程
j = -1;
for ( int i = 0; i < la; i++)
{
    while (j >= 0 && b[j + 1] != a[i]) j = p[j];
    if (b[j + 1] == a[i]) j ++;
    KMP[i] = j + 1;
}
```

3.5 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。

```
//自匹配过程
int j = 0;
while (j < lb && b[j] == b[j + 1])
    j++;
p[0] = lb, p[1] = j;
int k = 1;
for (int i = 2; i < lb; i++)
{
    int Len = k + p[k] - 1, L = p[i - k];
    if (L < Len - i + 1)
        p[i] = L;
    else
    {
        j = max(0, Len - i + 1);
        while (i + j < lb && b[i + j] == b[j])
            j++;
        p[i] = j, k = i;
    }
}
//下面是匹配过程
j = 0;
while (j < la && j < lb && a[j] == b[j])
    j++;
eKMP[0] = j;
k = 0;
for (int i = 1; i < la; i++)
{
    int Len = k + eKMP[k] - 1, L = p[i - k];
    if (L < Len - i + 1)
        eKMP[i] = L;
```

```
    else
    {
        j = max(0, Len - i + 1);
        while (i + j < la && j < lb && a[i + j] == b[j])
            j++;
        eKMP[i] = j, k = i;
    }
}
```

3.6 *Manacher

待整理

```
char s[1000],a[3000];
int p[3000],len,l,pnow,pid,res,resid;

int main()
{
    while (scanf("%s",s) != EOF)
    {
        len = strlen(s);
        l = 0;
        a[l++] = '.';
        a[l++] = ',';
        for (int i = 0;i < len;i++)
        {
            a[l++] = s[i];
            a[l++] = ',';
        }
        pnow = 0;
        res = 0;
        for (int i = 1;i < l;i++)
        {
            if (pnow > i)
                p[i] = min(p[2*pid-i],pnow-i);
            else
                p[i] = 1;
            for (;a[i-p[i]] == a[i+p[i]];p[i]++);
            if (i+p[i] > pnow)
            {
                pnow = i+p[i];
                pid = i;
            }
            if (p[i] > res)
            {
                res = p[i];
                resid = i;
            }
        }
        for (int i = resid-res+2;i < resid+res-1;i += 2)
            printf("%c",a[i]);
        printf("\n");
    }
    return 0;
}
```

3.7 *字符串最小表示法

```
int Gao(char a[],int len)
{
    int i = 0,j = 1,k = 0;
    while (i < len && j < len && k < len)
    {
```

```
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  }
```

4 数学

4.1 模线性方程组

```
//有更新
int m[10],a[10];//模数m 余数a
bool solve(int &m0,int &a0,int m,int a)//模线性方程组
{
    int y,x;
    int g=ex_gcd(m0,m,x,y);
    if (abs(a-a0)%g) return 0;
    x*=(a-a0)/g;
    x%=m/g;
    a0=(x*m0+a0);
    m0*=m/g;
    a0%=m0;
    if (a0<0) a0+=m0;
    return 1;
}
int MLES()
{
    bool flag=1;
    int m0=1,a0=0;
    for (int i=0; i<n; i++)
        if (!solve(m0,a0,m[i],a[i]))
        {
            flag=0;
            break;
        }
    if (flag)
        return a0;
    else
        return -1;
}
```

4.2 扩展GCD

求ax+by=gcd(a,b)的一组解

```
long long ex_gcd(long long a,long long b,long long &x,long long &y)
{
    if (b)
    {
        long long ret = ex_gcd(b,a%b,x,y),tmp = x;
        x = y;
        y = tmp-(a/b)*y;
        return ret;
    }
    else
    {
        x = 1;
        y = 0;
        return a;
    }
}
```

4.3 矩阵

```
struct Matrix
{
    int a[52][52];
    Matrix operator * (const Matrix &b)const
    {
```

```
        Matrix res;
        for (int i = 0; i < 52; i++)
            for (int j = 0; j < 52; j++)
            {
                res.a[i][j] = 0;
                for (int k = 0; k < 52; k++)
                    res.a[i][j] += a[i][k] * b.a[k][j];
            }
        return res;
    }
    Matrix operator ^ (int y)const
    {
        Matrix res, x;
        for (int i = 0; i < 52; i++)
        {
            for (int j = 0; j < 52; j++)
                res.a[i][j] = 0, x.a[i][j] = a[i][j];
            res.a[i][i] = 1;
        }
        for (; y; y >>= 1, x = x * x)
            if (y & 1)
                res = res * x;
        return res;
    }
};
```

4.4 康拓展开

```
const int PermSize = 12;
int factory[PermSize] = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880,
   3628800, 39916800};
int Cantor(int a[])
{
    int i, j, counted;
    int result = 0;
    for (i = 0; i < PermSize; ++i)
    {
        counted = 0;
        for (j = i + 1; j < PermSize; ++j)
            if (a[i] > a[j])
                ++counted;
        result = result + counted * factory[PermSize - i - 1];
    }
    return result;
}

bool h[13];

void UnCantor(int x, int res[])
{
    int i,j,l,t;
    for (i = 1;i <= 12;i++)
        h[i] = false;
    for (i = 1; i <= 12; i++)
    {
        t = x / factory[12 - i];
        x -= t * factory[12 - i];
        for (j = 1, l = 0; l <= t; j++)
            if (!h[j])l++;
        j--;
        h[j] = true;
        res[i - 1] = j;
    }
}
```

5 数据结构

5.1 *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

节点定义。。

```
const int MaxN = 50003;

struct Node
{
    int size,key;

    Node *c[2];
    Node *p;
} mem[MaxN], *cur, *nil;
```

无内存池的几个初始化函数。

```
Node *newNode(int v, Node *p)
{
    cur->c[0] = cur->c[1] = nil, cur->p = p;
    cur->size = 1;
    cur->key = v;
    return cur++;
}

void Init()
{
    cur = mem;
    nil = newNode(0, cur);
    nil->size = 0;
}
```

带内存池的几个函数。

```
int emp[MaxN], totemp;

Node *newNode(int v, Node *p)
{
    cur = mem + emp[--totemp];
    cur->c[0] = cur->c[1] = nil, cur->p = p;
    cur->size = 1;
    cur->key = v;
    return cur;
}

```

```
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 }
```

基本的Splay框架。维护序列用。
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         while (x->p != f)
34         {
35             Node *y = x->p;
36             if (y->p == f)
37                 Rotate(x, x == y->c[0]);
38             else
39             {
40                 int fd = y->p->c[0] == y;
41                 if (y->c[fd] == x)
42                     Rotate(x, fd ^ 1), Rotate(x, fd);
43                 else
```

```
                    Rotate(y, fd), Rotate(x, fd);
            }
        }
        Pushup(x);
        if (f == nil)
            root = x;
    }
    void Select(int k, Node *f)
    {
        Node *x = root;
        Pushdown(x);
        int tmp;
        while ((tmp = x->c[0]->size) != k)
        {
            if (k < tmp)    x = x->c[0];
            else
                x = x->c[1], k -= tmp + 1;
            Pushdown(x);
        }
        Splay(x, f);
    }
    void Select(int l, int r)
    {
        Select(l, nil), Select(r + 2, root);
    }
    Node *Make_tree(int a[], int l, int r, Node *p)
    {
        if (l > r)  return nil;
        int mid = l + r >> 1;
        Node *x = newNode(a[mid], p);
        x->c[0] = Make_tree(a, l, mid - 1, x);
        x->c[1] = Make_tree(a, mid + 1, r, x);
        Pushup(x);
        return x;
    }
    void Insert(int pos, int a[], int n)
    {
        Select(pos, nil), Select(pos + 1, root);
        root->c[1]->c[0] = Make_tree(a, 0, n - 1, root->c[1]);
        Splay(root->c[1]->c[0], nil);
    }
    void Insert(int v)
    {
        Node *x = root, *y = nil;
        while (x != nil)
        {
            y = x;
            y->size++;
            x = x->c[v >= x->key];
        }
        y->c[v >= y->key] = x = newNode(v, y);
        Splay(x, nil);
    }
    void Remove(int l, int r)
    {
        Select(l, r);
        //Recycle(root->c[1]->c[0]);
        root->c[1]->c[0] = nil;
        Splay(root->c[1], nil);
    }
};
```

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);

```
    void Pushup(Node *x)
    {
        if (x == nil)   return;
        Pushdown(x); Pushdown(x->c[0]); Pushdown(x->c[1]);
        x->size = x->c[0]->size+x->c[1]->size+1;

        x->sum = x->c[0]->sum+x->c[1]->sum+x->key;
        x->lsum = max(x->c[0]->lsum,x->c[0]->sum+x->key+max(0,x->c[1]->lsum)
            );
        x->rsum = max(x->c[1]->rsum,x->c[1]->sum+x->key+max(0,x->c[0]->rsum)
            );
        x->maxsum = max(max(x->c[0]->maxsum,x->c[1]->maxsum),x->key+max(0,x
            ->c[0]->rsum)+max(0,x->c[1]->lsum));
    }
    void Pushdown(Node *x)
    {
        if (x == nil)   return;
        if (x->rev)
        {
            x->rev = 0;
            x->c[0]->rev ^= 1;
            x->c[1]->rev ^= 1;
            swap(x->c[0],x->c[1]);

            swap(x->lsum,x->rsum);
        }
        if (x->same)
        {
            x->same = false;
            x->key = x->lazy;
            x->sum = x->key*x->size;
            x->lsum = x->rsum = x->maxsum = max(x->key,x->sum);
            x->c[0]->same = true, x->c[0]->lazy = x->key;
            x->c[1]->same = true, x->c[1]->lazy = x->key;
        }
    }

int main()
{
    int totcas;
    scanf("%d",&totcas);
    for (int cas = 1;cas <= totcas;cas++)
    {
        Init();
        sp.Init();
        nil->lsum = nil->rsum = nil->maxsum = -Inf;
        sp.Insert(0);
        sp.Insert(0);

        int n,m;
        scanf("%d%d",&n,&m);
        for (int i = 0;i < n;i++)
            scanf("%d",&a[i]);
        sp.Insert(0,a,n);

        for (int i = 0;i < m;i++)
        {
            int pos,tot,c;
            scanf("%s",buf);
            if (strcmp(buf,"MAKE-SAME") == 0)
            {
                scanf("%d%d%d",&pos,&tot,&c);
```

```
                sp.Select(pos-1,pos+tot-2);
                sp.root->c[1]->c[0]->same = true;
                sp.root->c[1]->c[0]->lazy = c;
                sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
            }
            else if (strcmp(buf,"INSERT") == 0)
            {
                scanf("%d%d",&pos,&tot);
                for (int i = 0;i < tot;i++)
                    scanf("%d",&a[i]);
                sp.Insert(pos,a,tot);
            }
            else if (strcmp(buf,"DELETE") == 0)
            {
                scanf("%d%d",&pos,&tot);
                sp.Remove(pos-1,pos+tot-2);
            }
            else if (strcmp(buf,"REVERSE") == 0)
            {
                scanf("%d%d",&pos,&tot);
                sp.Select(pos-1,pos+tot-2);
                sp.root->c[1]->c[0]->rev ^= 1;
                sp.Pushup(sp.root->c[1]), sp.Pushup(sp.root);
            }
            else if (strcmp(buf,"GET-SUM") == 0)
            {
                scanf("%d%d",&pos,&tot);
                sp.Select(pos-1,pos+tot-2);
                printf("%d\n",sp.root->c[1]->c[0]->sum);
            }
            else if (strcmp(buf,"MAX-SUM") == 0)
            {
                sp.Select(0,sp.root->size-3);
                printf("%d\n",sp.root->c[1]->c[0]->maxsum);
            }
        }
    }
    return 0;
}
```

维护多个序列的时候，不需要建立很多Splay。只需要记录某个点在内存池中的绝对位置就可以了。
需要操作它所在的序列时直接Splay到nil。此时Splay的root所在的Splay就是这个序列了。
新建序列的时候需要多加入两个额外节点。如果某个Splay只有两个节点了需要及时回收。
例题：Box（维护括号序列）

```
    \\下面都是专用函数
    \\判断x在不在f里面
    bool Ancestor(Node *x,Node *f)
    {
        if (x == f) return true;
        while (x->p != nil)
        {
            if (x->p == f)  return true;
            x = x->p;
        }
        return false;
    }
    \\把Splay v插入到pos后面，pos=nil时新开一个序列
    void Insert(Node *pos, Node *v)
    {
        int pl;
        if (pos == nil)
        {
```

```
19              Init();
20              Insert(0), Insert(0);
21              pl = 0;
22          }
23          else
24          {
25              Splay(pos, nil);
26              pl = root->c[0]->size;
27          }
28          Select(pl, nil), Select(pl + 1, root);
29          root->c[1]->c[0] = v;
30          v->p = root->c[1];
31          Splay(v, nil);
32      }
33      \\把[l,r]转出来（这里记录的是绝对位置）
34      void Select(Node *l, Node *r)
35      {
36          Splay(l, nil);
37          int pl = root->c[0]->size - 1;
38          Splay(r, nil);
39          int pr = root->c[0]->size - 1;
40          Select(pl, pr);
41      }
42      \\分离[l,r]
43      Node *Split(Node *l, Node *r)
44      {
45          Select(l, r);
46          Node *res = root->c[1]->c[0];
47          root->c[1]->c[0] = res->p = nil;
48          Splay(root->c[1], nil);
49          if (root->size == 2)
50          {
51              Recycle(root);
52              Init();
53          }
54          return res;
55      }
56
57  int main(int argc, char const *argv[])
58  {
59      freopen("P.in", "r", stdin);
60      bool first = true;
61      while (scanf("%d", &n) != EOF)
62      {
63          if (!first) puts("");
64          first = false;
65          Init();
66          for (int i = 0; i < n; i++)
67          {
68              \\建立独立的N个区间，记录绝对位置
69              sp.Init();
70              sp.Insert(0), sp.Insert(0);
71              sp.Insert(0,i+1),sp.Insert(1,i+1);
72              sp.Select(0, 0), l[i] = sp.root->c[1]->c[0];
73              sp.Select(1, 1), r[i] = sp.root->c[1]->c[0];
74          }
75          for (int i = 0; i < n; i++)
76          {
77              int f;
78              scanf("%d", &f);
79              if (f != 0)
80              {
81                  \\把[l[i],r[i]]插入到l[f-1]后面
```

```
                Node *pos = sp.Split(l[i], r[i]);
                sp.Insert(l[f - 1], pos);
            }
        }
        scanf("%d", &n);
        for (int i = 0; i < n; i++)
        {
            scanf("%s", com);
            if (com[0] == 'Q')
            {
                int pos;
                scanf("%d", &pos);
                \\求[l[pos-1],r[pos-1]]在哪个序列里面
                sp.Splay(l[pos - 1], nil);
                sp.Select(1, nil);
                printf("%d\n", sp.root->key);
            }
            else
            {
                int u, v;
                scanf("%d%d", &u, &v);
                if (v == 0)
                    sp.Insert(nil, sp.Split(l[u-1], r[u-1]));
                else
                {
                    sp.Select(l[u-1],r[u-1]);
                    if (sp.Ancestor(l[v-1],sp.root->c[1]->c[0]) == false)
                        sp.Insert(l[v - 1], sp.Split(l[u-1], r[u-1]));
                }
            }
        }
    }
    return 0;
}
```

5.2 *动态树

5.2.1 维护点权

被注释的部分是具体题目用到的东西。
支持换根。
Cut操作还没写。

```
const int MaxN = 110000;

struct Node
{
    int size, key;
    bool rev;

//    bool same;
//    int lsum, rsum, sum, maxsum, sa;

    Node *c[2];
    Node *p;
} mem[MaxN], *cur, *nil, *pos[MaxN];

Node *newNode(int v, Node *p)
{
    cur->c[0] = cur->c[1] = nil, cur->p = p;
    cur->size = 1;
    cur->key = v;
    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, x->c[0]->sum + x->key + max(0, x->c
        [1]->lsum));
47  //        x->rsum = max(x->c[1]->rsum, x->c[1]->sum + x->key + max(0, x->c
        [0]->rsum));
48  //        x->maxsum = max(max(x->c[0]->maxsum, x->c[1]->maxsum),
49  //            x->key + max(0, x->c[0]->rsum) + max(0, x->c[1]->lsum));
50
51      }
52      void Pushdown(Node *x)
53      {
54          if (x == nil)   return;
55          if (x->rev)
56          {
57              x->rev = 0;
58              x->c[0]->rev ^= 1;
59              x->c[1]->rev ^= 1;
60              swap(x->c[0], x->c[1]);
61  //注意修改与位置有关的量
62  //              swap(x->lsum,x->rsum);
63          }
64
65  //        if (x->same)
66  //        {
67  //            x->same = false;
68  //            x->key = x->sa;
69  //            x->sum = x->sa * x->size;
70  //            x->lsum = x->rsum = x->maxsum = max(0, x->sum);
71  //            if (x->c[0] != nil)
72  //                x->c[0]->same = true, x->c[0]->sa = x->sa;
73  //            if (x->c[1] != nil)
74  //                x->c[1]->same = true, x->c[1]->sa = x->sa;
75  //        }
76      }
77      bool isRoot(Node *x)
78      {
79          return (x == nil) || (x->p->c[0] != x && x->p->c[1] != x);
80      }
81      void Rotate(Node *x, int f)
```

```
82      {
83          if (isRoot(x))   return;
84          Node *y = x->p;
85          y->c[f ^ 1] = x->c[f], x->p = y->p;
86          if (x->c[f] != nil)
87              x->c[f]->p = y;
88          if (y != nil)
89          {
90              if (y == y->p->c[1])
91                  y->p->c[1] = x;
92              else if (y == y->p->c[0])
93                  y->p->c[0] = x;
94          }
95          x->c[f] = y, y->p = x;
96          Pushup(y);
97      }
98      void Splay(Node *x)
99      {
100         static Node *stack[MaxN];
101         int top = 0;
102         stack[top++] = x;
103         for (Node *y = x; !isRoot(y); y = y->p)
104             stack[top++] = y->p;
105         while (top)
106             Pushdown(stack[--top]);
107
108         while (!isRoot(x))
109         {
110             Node *y = x->p;
111             if (isRoot(y))
112                 Rotate(x, x == y->c[0]);
113             else
114             {
115                 int fd = y->p->c[0] == y;
116                 if (y->c[fd] == x)
117                     Rotate(x, fd ^ 1), Rotate(x, fd);
118                 else
119                     Rotate(y, fd), Rotate(x, fd);
120             }
121         }
122         Pushup(x);
123     }
124     Node *Access(Node *u)
125     {
126         Node *v = nil;
127         while (u != nil)
128         {
129             Splay(u);
130             v->p = u;
131             u->c[1] = v;
132             Pushup(u);
133             u = (v = u)->p;
134             if (u == nil)
135                 return v;
136         }
137     }
138     Node *LCA(Node *u, Node *v)
139     {
140         Access(u);
141         return Access(v);
142     }
143     Node *Link(Node *u, Node *v)
144     {
```

```
145         Access(u);
146         Splay(u);
147         u->rev = true;
148         u->p = v;
149     }
150     void ChangeRoot(Node *u)
151     {
152         Access(u)->rev ^= 1;
153     }
154     Node *GetRoute(Node *u, Node *v)
155     {
156         ChangeRoot(u);
157         return Access(v);
158     }
159 };
160 
161 int n, m;
162 SplayTree sp;
163 
164 int main(int argc, char const *argv[])
165 {
166     while (scanf("%d", &n) != EOF)
167     {
168         Init();
169         for (int i = 0; i < n; i++)
170         {
171             int v;
172             scanf("%d", &v);
173             pos[i] = newNode(v, nil);
174         }
175         for (int i = 0; i < n - 1; i++)
176         {
177             int u, v;
178             scanf("%d%d", &u, &v);
179             u--, v--;
180             sp.Link(pos[u], pos[v]);
181         }
182 
183 //         scanf("%d", &m);
184 //         for (int i = 0; i < m; i++)
185 //         {
186 //             int typ, u, v, c;
187 //             scanf("%d%d%d", &typ, &u, &v);
188 //             u--, v--;
189 //             if (typ == 1)
190 //                 printf("%d\n", sp.GetRoute(pos[u], pos[v])->maxsum);
191 //             else
192 //             {
193 //                 scanf("%d", &c);
194 //                 Node *p = sp.GetRoute(pos[u], pos[v]);
195 //                 p->same = true;
196 //                 p->sa = c;
197 //             }
198 //         }
199     }
200     return 0;
201 }
```

5.3 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;
```

```
}
```

5.4 treap正式版

支持翻转。

```
#include <cstdio>
#include <cstdlib>
#include <algorithm>
using namespace std;

const int MAXN = 100000;
const int MAXM = 100000;
const int inf = 0x7fffffff;
int a[MAXN];
struct Treap
{
    int N;
    Treap()
    {
        N = 0;
        root = NULL;
    }
    void init()
    {
        N = 0;
        root = NULL;
    }
    struct Treap_Node
    {
        Treap_Node *son[2];//left & right
        int value, fix;
        bool lazy;
        int size;
        Treap_Node() {}
        Treap_Node(int _value)
        {
            son[0] = son[1] = NULL;
            value = _value;
            fix = rand() * rand();
            lazy = 0;
            size = 1;
        }
        int sonSize(bool flag)
        {
            if (son[flag] == NULL)
                return 0;
            else
                return son[flag]->size;
        }
    } node[MAXN], *root, *pos[MAXN];
    void up(Treap_Node *p)
    {
        p->size = p->sonSize(0) + p->sonSize(1) + 1;
    }
    void down(Treap_Node *p)
    {
        if (!p->lazy)
            return ;
        for (int i = 0; i < 2; i++)
            if (p->son[i])
                p->son[i]->lazy = !p->son[i]->lazy;
        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 *> ret = split(p->son[1], n - p->
                    sonSize(0) - 1);
100             p->son[1] = ret.first;
101             up(p);
102             return make_pair(p, ret.second);
103         }
104     }
105     int smalls(Treap_Node *p,int value)
106     {
107         if (p==NULL)
108             return 0;
109         if (p->value<=value)
110             return 1+p->sonSize(0)+smalls(p->son[1],value);
111         else
112             return smalls(p->son[0],value);
113     }
114     void insert(int value)
115     {
116         Treap_Node *p = &node[N++];
117         *p = Treap_Node(value);
118         pair<Treap_Node *, Treap_Node *> ret = split(root, smalls(root,
                value));
```

```
119         root = merge(merge(ret.first, p), ret.second);
120     }
121     void remove(int value)
122     {
123         pair<Treap_Node *, Treap_Node *> ret = split(root, smalls(root,
                value) - 1);
124         root = merge(ret.first, split(ret.second, 1).second);
125     }
126     Treap_Node *build(int s, int t)
127     {
128         int idx = t + s >> 1;
129         Treap_Node *p = &node[N++];
130         *p = Treap_Node(a[idx]);
131         pos[a[idx]] = p;
132         if (idx > s)
133             p = merge(build(s, idx - 1), p);
134         if (idx < t)
135             p = merge(p, build(idx + 1, t));
136         up(p);
137         return p;
138     }
139     void build(int n)
140     {
141         root = build(0, n - 1);
142     }
143     void *reverse(int s, int t)
144     {
145         pair<Treap_Node *, Treap_Node *> tmp1, tmp2;
146         tmp1 = split(root, s - 1);
147         tmp2 = split(tmp1.second, t - s + 1);
148         tmp2.first->lazy = !tmp2.first->lazy;
149         root = merge(tmp1.first, merge(tmp2.first, tmp2.second));
150     }
151 };
152 Treap treap;
153 int main()
154 {
155     treap.init();
156     int n;
157     scanf("%d", &n);
158     for (int i = 0; i < n; i++)
159         scanf("%d", &a[i]);
160     treap.build(n);
161 }
```

6 图论

6.1 SAP四版

```
const int MAXEDGE=20400;
const int MAXN=400;
const int inf=0x3fffffff;
struct edges
{
    int cap,to,next,flow;
} edge[MAXEDGE+100];
struct nodes
{
    int head,label,pre,cur;
} node[MAXN+100];
int L,N;
int gap[MAXN+100];
void init(int n)
{
    L=0;
    N=n;
    for (int i=0; i<N; i++)
        node[i].head=-1;
}
void add_edge(int x,int y,int z,int w)
{
    edge[L].cap=z;
    edge[L].flow=0;
    edge[L].to=y;
    edge[L].next=node[x].head;
    node[x].head=L++;
    edge[L].cap=w;
    edge[L].flow=0;
    edge[L].to=x;
    edge[L].next=node[y].head;
    node[y].head=L++;
}
int maxflow(int s,int t)
{
    memset(gap,0,sizeof(gap));
    gap[0]=N;
    int u,ans=0;
    for (int i=0; i<N; i++)
    {
        node[i].cur=node[i].head;
        node[i].label=0;
    }
    u=s;
    node[u].pre=-1;
    while (node[s].label<N)
    {
        if (u==t)
        {
            int min=inf;
            for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
                if (min>edge[i].cap-edge[i].flow)
                    min=edge[i].cap-edge[i].flow;
            for (int i=node[u].pre; i!=-1; i=node[edge[i^1].to].pre)
            {
                edge[i].flow+=min;
                edge[i^1].flow-=min;
            }
            u=s;
```

```
            ans+=min;
            continue;
        }
        bool flag=false;
        int v;
        for (int i=node[u].cur; i!=-1; i=edge[i].next)
        {
            v=edge[i].to;
            if (edge[i].cap-edge[i].flow && node[v].label+1==node[u].label)
            {
                flag=true;
                node[u].cur=node[v].pre=i;
                break;
            }
        }
        if (flag)
        {
            u=v;
            continue;
        }
        node[u].cur=node[u].head;
        int min=N;
        for (int i=node[u].head; i!=-1; i=edge[i].next)
            if (edge[i].cap-edge[i].flow && node[edge[i].to].label<min)
                min=node[edge[i].to].label;
        gap[node[u].label]--;
        if (!gap[node[u].label]) return ans;
        node[u].label=min+1;
        gap[node[u].label]++;
        if (u!=s) u=edge[node[u].pre^1].to;
    }
    return ans;
}
```

6.2 费用流三版

T了可以改成栈。

```
const int MAXM=60000;
const int MAXN=400;
const int inf=0x3fffffff;
int L,N;
int K;
struct edges
{
    int to,next,cap,flow,cost;
} edge[MAXM];
struct nodes
{
    int dis,pre,head;
    bool visit;
} node[MAXN];
void init(int n)
{
    N=n;
    L=0;
    for (int i=0; i<N; i++)
        node[i].head=-1;
}
void add_edge(int x,int y,int cap,int cost)
{
    edge[L].to=y;
    edge[L].cap=cap;
    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=i;
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  }
```

6.3 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)
```

```
    {
        v = Match[u];
        InBlossom[Base[u]] = InBlossom[Base[v]] = true;
        u = Father[v];
        if (Base[u] != NewBase) Father[u] = v;
    }
}
void BlossomContract(int u,int v)
{
    NewBase = FindCommonAncestor(u,v);
    memset(InBlossom,false,sizeof(InBlossom));
    ResetTrace(u);
    ResetTrace(v);
    if (Base[u] != NewBase) Father[u] = v;
    if (Base[v] != NewBase) Father[v] = u;
    for (int tu = 1; tu <= N; tu++)
        if (InBlossom[Base[tu]])
        {
            Base[tu] = NewBase;
            if (!InQueue[tu]) Push(tu);
        }
}
void FindAugmentingPath()
{
    memset(InQueue,false,sizeof(InQueue));
    memset(Father,0,sizeof(Father));
    for (int i = 1; i <= N; i++)
        Base[i] = i;
    Head = Tail = 1;
    Push(Start);
    Finish = 0;
    while (Head < Tail)
    {
        int u = Pop();
        for (int v = 1; v <= N; v++)
            if (Graph[u][v] && (Base[u] != Base[v]) && (Match[u] != v))
            {
                if ((v == Start) || ((Match[v] > 0) && (Father[Match[v]] >
                   0)))
                    BlossomContract(u,v);
                else if (Father[v] == 0)
                {
                    Father[v] = u;
                    if (Match[v] > 0)
                        Push(Match[v]);
                    else
                    {
                        Finish = v;
                        return;
                    }
                }
            }
    }
}
void AugmentPath()
{
    int u,v,w;
    u = Finish;
    while (u > 0)
    {
        v = Father[u];
        w = Match[v];
        Match[v] = u;
```

```
116         Match[u] = v;
117         u = w;
118     }
119 }
120 void Edmonds()
121 {
122     memset(Match,0,sizeof(Match));
123     for (int u = 1; u <= N; u++)
124         if (Match[u] == 0)
125         {
126             Start = u;
127             FindAugmentingPath();
128             if (Finish > 0) AugmentPath();
129         }
130 }
131 void PrintMatch()
132 {
133     for (int u = 1; u <= N; u++)
134         if (Match[u] > 0)
135             Count++;
136     printf("%d\n",Count);
137     for (int u = 1; u <= N; u++)
138         if (u < Match[u])
139             printf("%d␣%d\n",u,Match[u]);
140 }
141 int main()
142 {
143     CreateGraph();
144     Edmonds();
145     PrintMatch();
146 }
```

6.4 *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 }
```

7 计算几何

太乱了尼玛。。

7.1 基本函数

7.1.1 Point定义

```
struct Point
{
    double x, y;
    Point() {}
    Point(double _x, double _y)
    {
        x = _x, y = _y;
    }
    Point operator -(const Point &b)const
    {
        return Point(x - b.x, y - b.y);
    }
    double operator *(const Point &b)const
    {
        return x * b.y - y * b.x;
    }
    double operator &(const Point &b)const
    {
        return x * b.x + y * b.y;
    }
};
```

7.1.2 Line定义

```
struct Line
{
    Point s, e;
    double k;
    Line() {}
    Line(Point _s, Point _e)
    {
        s = _s, e = _e;
        k = atan2(e.y - s.y, e.x - s.x);
    }
    Point operator &(const Line &b)const
    {
        Point res = s;
        //注意：有些题目可能会有直线相交或者重合情况
        //可以把返回值改成pair<Point,int>来返回两直线的状态。
        double t = ((s - b.s) * (b.s - b.e)) / ((s - e) * (b.s - b.e));
        res.x += (e.x - s.x) * t;
        res.y += (e.y - s.y) * t;
        return res;
    }
};
```

7.1.3 距离：两点距离

```
double dist2(Point a, Point b)
{
    return (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y);
}
```

7.1.4 距离：点到线段距离

res：点到线段最近点

```
double dist2(Point p1, Point p2, Point p)
{
    Point res;
    double a, b, t;
    a = p2.x - p1.x;
    b = p2.y - p1.y;
    t = ((p.x - p1.x) * a + (p.y - p1.y) * b) / (a * a + b * b);
    if (t >= 0 && t <= 1)
    {
        res.x = p1.x + a * t;
        res.y = p1.y + b * t;
    }
    else
    {
        if (dist2(p, p1) < dist2(p, p2))
            res = p1;
        else
            res = p2;
    }
    return dist2(p, res);
}
```

7.1.5 面积：多边形

点按逆时针排序。

```
double CalcArea(Point p[], int n)
{
    double res = 0;
    for (int i = 0; i < n; i++)
        res += (p[i] * p[(i + 1) % n]) / 2;
    return res;
}
```

7.2 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。

```
bool Div[MaxN];
void BuildKD(int deep,int l, int r, Point p[])\\记得备份一下P
{
    if (l > r) return;
    int mid = l + r >> 1;
    int minX, minY, maxX, maxY;
    minX = min_element(p + l, p + r + 1, cmpX)->x;
    minY = min_element(p + l, p + r + 1, cmpY)->y;
    maxX = max_element(p + l, p + r + 1, cmpX)->x;
    maxY = max_element(p + l, p + r + 1, cmpY)->y;
    Div[mid] = (maxX - minX >= maxY - minY);
    nth_element(p + l, p + mid, p + r + 1, Div[mid] ? cmpX : cmpY);
    BuildKD(l, mid - 1, p);
    BuildKD(mid + 1, r, p);
}

long long res;
void Find(int l, int r, Point a, Point p[])\\查找
{
    if (l > r)  return;
    int mid = l + r >> 1;
    long long dist = dist2(a, p[mid]);
    if (dist > 0)//如果有重点不能这样判断
        res = min(res, dist);
```

```
    long long d = Div[mid] ? (a.x - p[mid].x) : (a.y - p[mid].y);
    int l1, l2, r1, r2;
    l1 = l, l2 = mid + 1;
    r1 = mid - 1, r2 = r;
    if (d > 0)
        swap(l1, l2), swap(r1, r2);
    Find(l1, r1, a, p);
    if (d * d < res)
        Find(l2, r2, a, p);
}
```

7.2.1 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）

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <queue>
using namespace std;

const int MaxN = 100000;
struct Point
{
    int x,y,r;
    int id;
    bool del;
};

int cmpTyp;
bool cmp(const Point& a,const Point& b)
{
    if (cmpTyp == 0)
        return a.x < b.x;
    else
        return a.y < b.y;
}

int cnt[MaxN];
bool Div[MaxN];
int minX[MaxN],minY[MaxN],maxX[MaxN],maxY[MaxN];
void BuildKD(int l,int r,Point p[])
{
    if (l > r)  return;
    int mid = l+r>>1;
    cmpTyp = 0;
    minX[mid] = min_element(p+l,p+r+1,cmp)->x;
    maxX[mid] = max_element(p+l,p+r+1,cmp)->x;
    cmpTyp = 1;
    minY[mid] = min_element(p+l,p+r+1,cmp)->y;
    maxY[mid] = max_element(p+l,p+r+1,cmp)->y;

    cnt[mid] = r-l+1;
    cmpTyp = Div[mid] = (maxX[mid]-minX[mid] < maxY[mid]-minY[mid]);
    nth_element(p+l,p+mid,p+r+1,cmp);
    BuildKD(l,mid-1,p);
    BuildKD(mid+1,r,p);
}

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%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);
```

```
            while (!Q.empty())
            {
                int now = Q.front();
                Q.pop();
                if (p[now].del == true) continue;
                p[now].del = true;
                res += Find(0,n-1,p[now],tp);
            }
            printf("%d\n",res);
        }
    }
    return 0;
}
```

7.3 半平面交

直线左边代表有效区域。

```
bool HPIcmp(Line a, Line b)
{
    if (fabs(a.k - b.k) > eps)    return a.k < b.k;
    return ((a.s - b.s) * (b.e-b.s)) < 0;
}

Line Q[100];
void HPI(Line line[], int n, Point res[], int &resn)
{
    int tot = n;
    sort(line, line + n, HPIcmp);
    tot = 1;
    for (int i = 1; i < n; i++)
        if (fabs(line[i].k - line[i - 1].k) > eps)
            line[tot++] = line[i];
    int head = 0, tail = 1;
    Q[0] = line[0];
    Q[1] = line[1];
    resn = 0;
    for (int i = 2; i < tot; i++)
    {
        if (fabs((Q[tail].e-Q[tail].s) * (Q[tail - 1].e-Q[tail - 1].s)) <
           eps ||
                fabs((Q[head].e-Q[head].s) * (Q[head + 1].e-Q[head + 1].s))
                   < eps)
            return;
        while (head < tail && (((Q[tail]&Q[tail - 1]) - line[i].s) * (line[i
           ].e-line[i].s)) > eps)
            tail--;
        while (head < tail && (((Q[head]&Q[head + 1]) - line[i].s) * (line[i
           ].e-line[i].s)) > eps)
            head++;
        Q[++tail] = line[i];
    }
    while (head < tail && (((Q[tail]&Q[tail - 1]) - Q[head].s) * (Q[head].e-
       Q[head].s)) > eps)
        tail--;
    while (head < tail && (((Q[head]&Q[head + 1]) - Q[tail].s) * (Q[tail].e-
       Q[tail].s)) > eps)
        head++;
    if (tail <= head + 1) return;
    for (int i = head; i < tail; i++)
        res[resn++] = Q[i] & Q[i + 1];
    if (head < tail + 1)
        res[resn++] = Q[head] & Q[tail];
```

```
}
```

7.4 凸包

得到的凸包按照逆时针方向排序。

```
bool GScmp(Point a, Point b)
{
    if (fabs(a.x - b.x) < eps)
        return a.y < b.y - eps;
    return a.x < b.x - eps;
}

void GS(Point p[], int n, Point res[], int &resn)
{
    resn = 0;
    int top = 0;
    sort(p, p + n, GScmp);
    for (int i = 0; i < n;)
        if (resn < 2 || (res[resn - 1] - res[resn - 2]) * (p[i] - res[resn - 1]) > eps)
            res[resn++] = p[i++];
        else
            --resn;
    top = resn - 1;
    for (int i = n - 2; i >= 0;)
        if (resn < top + 2 || (res[resn - 1] - res[resn - 2]) * (p[i] - res[resn - 1]) > eps)
            res[resn++] = p[i--];
        else
            --resn;
    resn--;
    if (resn < 3)   resn = 0;
}
```

7.5 直线与凸包求交点

复杂度$O(\log n)$。
需要先预处理几个东西。

```
//二分[la,lb]这段区间那条边与line相交
int Gao(int la,int lb,Line line)
{
    if (la > lb)
        lb += n;
    int l = la,r = lb,mid;
    while (l < r)
    {
        mid = l+r+1>>1;
        if (cmp((line.e-line.s)*(p[la]-line.s),0)*cmp((line.e-line.s)*(p[mid]-line.s),0) >= 0)
            l = mid;
        else
            r = mid-1;
    }
    return l%n;
}
//求l与凸包的交点

//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
double theta[maxn];

```

```
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 }
```

7.6 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, b, c
                ,
78                  P[F[t].b])) < eps && fabs(volume(a, b, c, P[F[t].c])) < eps;
79      }
80      void construct()
81      {
82          cnt = 0;
83          if (n < 4)
84              return;
85          bool sb = 1;
86          for (int i = 1; i < n; i++)
87          {
88              if (vlen(P[0] - P[i]) > eps)
89              {
90                  swap(P[1], P[i]);
91                  sb = 0;
92                  break;
93              }
94          }
95          if (sb)return;
96          sb = 1;
97          for (int i = 2; i < n; i++)
98          {
99              if (vlen((P[0] - P[1]) * (P[1] - P[i])) > eps)
100             {
101                 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                 swap(P[3], P[i]);
113                 sb = 0;
114                 break;
115             }
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 = 1;
122             if (ptof(P[i], add) > 0)
123                 swap(add.b, add.c);
124             to[add.a][add.b] = to[add.b][add.c] = to[add.c][add.a] = cnt;
125             F[cnt++] = add;
126         }
127         for (int i = 4; i < n; i++)
128         {
129             for (int j = 0; j < cnt; j++)
130             {
131                 if (F[j].ok && ptof(P[i], F[j]) > eps)
132                 {
133                     dfs(i, j);
134                     break;
```

```
                }
            }
        }
        int tmp = cnt;
        cnt = 0;
        for (int i = 0; i < tmp; i++)
        {
            if (F[i].ok)
            {
                F[cnt++] = F[i];
            }
        }
    }
//表面积
    double area()
    {
        double ret = 0.0;
        for (int i = 0; i < cnt; i++)
        {
            ret += area(P[F[i].a], P[F[i].b], P[F[i].c]);
        }
        return ret / 2.0;
    }
//体积
    double volume()
    {
        pt O(0, 0, 0);
        double ret = 0.0;
        for (int i = 0; i < cnt; i++)
        {
            ret += volume(O, P[F[i].a], P[F[i].b], P[F[i].c]);
        }
        return fabs(ret / 6.0);
    }
//表面三角形数
    int facetCnt_tri()
    {
        return cnt;
    }
//表面多边形数
    int facetCnt()
    {
        int ans = 0;
        for (int i = 0; i < cnt; i++)
        {
            bool nb = 1;
            for (int j = 0; j < i; j++)
            {
                if (same(i, j))
                {
                    nb = 0;
                    break;
                }
            }
            ans += nb;
        }
        return ans;
    }

    pt Fc[MAXV*8];
    double V[MAXV*8];
    pt Center()//重心
    {
```

```
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 }
```

8 杂物

8.1 高精度数

支持乘以整数和加法。

```
struct BigInt
{
    const static int mod = 100000000;
    int a[600],len;
    BigInt (){}
    BigInt (int v)
    {
        len = 0;
        do
        {
            a[len++] = v%mod;
            v /= mod;
        }while(v);
    }
    BigInt operator *(const int& b) const
    {
        BigInt res;
        res.len = len;
        for (int i = 0; i <= len; ++i)
            res.a[i] = 0;
        for (int i = 0; i < len; ++i)
        {
            res.a[i] += a[i]*b;
            res.a[i+1] += res.a[i]/mod;
            res.a[i] %= mod;
        }
        if (res.a[len] > 0)  res.len++;
        return res;
    }
    BigInt operator +(const BigInt& b) const
    {
        BigInt res;
        res.len = max(len,b.len);
        for (int i = 0; i <= res.len; ++i)
            res.a[i] = 0;
        for (int i = 0; i < res.len; ++i)
        {
            res.a[i] += ((i < len)?a[i]:0)+((i < b.len)?b.a[i]:0);
            res.a[i+1] += res.a[i]/mod;
            res.a[i] %= mod;
        }
        if (res.a[res.len] > 0)  res.len++;
        return res;
    }
    void output()
    {
        printf("%d",a[len-1]);
        for (int i = len-2; i >= 0; --i)
            printf("%08d",a[i]);
        printf("\n");
    }
};
```

8.2 整数外挂

```
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 }
```