ACM/ICPC TEMPLATE

NKU -> HOT

October 20, 2012

Contents

1 Introduction					
2	2.1 Java Template	2 3 4			
3	3.1 Prim - $O(N^2)$	2 3 4 6 7 8 9			
4	String Algorithm 2 4.1 ELF Hash 2 4.2 Aho-Corasick Automation 2 4.3 Suffix array 2	2			
5	Data Struct 2. 5.1 Binary Indexed Tree	24 25			
6	Computational Geometry 2 6.1 Constants 2 6.2 Compare Function 2 6.3 3D Point and Vector 2 6.4 3D Utility Functions 2 6.4.1 3D dot and 3D vector cross 2 6.4.2 3D line - line min distance 3 6.4.3 Sphere coord 3 6.5 2D Point and Vector 3 6.6 2D Utility Functions 3 6.6.1 2D dot and 2D scalar cross 3	999990000000000000000000000000000000000			
	6.6.2 2D triple point coline	1			

		6.6.3	2D segment - segment overlap	1
		6.6.4	2D angle difference between two vectors	
		6.6.5	2D point - line distance	1
		6.6.6	2D point - circle tangent	2
		6.6.7	2D circle - circle tangent	2
		6.6.8	2D circle - polygon intersection area	3
		6.6.9	2D vector normal	4
		6.6.10	2D vector rotation	4
	6.7	2D Int	ersection	4
		6.7.1	ray - ray intersection	4
		6.7.2	line - circle intersection	4
	6.8	Comp	utational Geometry Topics	5
		6.8.1	2D convex hull	5
		6.8.2	3D convec hull	5
		6.8.3	Max distance point pair	7
		6.8.4	closed point pair among two polygons	7
		6.8.5	closed point pair	9
		6.8.6	closed circle pair	0
		6.8.7	circle hierarchy	2
7	Math	h	4	4
	7.1	Catala	n Number	4

1 Introduction

NKU -> HOT ACM-ICPC template
Thanks all past teammates and contributers!

2 Utility

2.1 Java Template

```
import java.util.*;
2
   import java.io.*;
3
 4
   class Main {
5
6
       void run() {
7
            //Scanner in = new Scanner(System.in);
8
            MyReader in = new MyReader();
9
            String str;
10
11
        public static void main(String args[]) {
12
13
            new Main().run();
14
15
16
        void debug(Object...x) {
17
            System.out.println(Arrays.deepToString(x));
18
        }
19
   }
20
21
   class MyReader {
22
        BufferedReader br = new BufferedReader (
23
                new InputStreamReader (System.in));
24
        StringTokenizer in;
```

```
25
        String next() {
26
            try {
27
                while (in == null || !in.hasMoreTokens()) {
28
                     // Read a new line and split it into tokens
29
                     in = new StringTokenizer(br.readLine());
30
                }
31
                // return next token
                return in.nextToken();
32
33
            } catch (Exception e) {
                // EOF
34
35
                return null;
36
            }
37
38
        // Transform the tokens into other types
39
        int nextInt() {
40
            return Integer.parseInt(next());
41
        }
42
   }
```

2.2 Java Multithread

BECAREFUL: CALL START FOR EACH THREAD!

```
1
   class Test extends Thread {
2
        public static int ans;
3
        public static int end;
4
        public void run() {
5
            int now = 0;
            for (int i = 0; i < 400000000; i++) {
6
7
                now = (now + i) \% 9999997;
8
9
            System.out.println(now);
10
            new SubTask(0,0,100000000).start();
            new SubTask(1,100000000,200000000).start();
11
            new SubTask(2,200000000,300000000).start();
12
13
            new SubTask(3,300000000,40000000).start();
            for (;;) {
14
15
                try {
16
                     sleep(200);
                } catch (Exception e) {}
17
                if (end == 4) break;
18
19
20
            System.out.println(ans);
21
22
        public static void main(String[] args) {
23
            new Test().start();
24
25
   }
26
27
   class SubTask extends Thread {
28
        private int pos;
29
        private int left;
30
        private int right;
31
        final static int mod = 9999997;
32
33
        // init the input data
34
        SubTask(int pos, int left, int right) {
35
            this.pos = pos;
```

```
this.left = left;
36
37
            this.right = right;
38
        }
39
40
        public void run() {
41
            // solve the problem
42
            int ans = 0;
43
            for (int i = left; i < right; i++) {
44
                 ans = (ans + i) % mod;
45
            }
46
            // write the answer back
47
            synchronized (this) {
48
                 Test.ans += ans;
49
                 Test.ans %= mod;
50
                 Test.end ++;
51
            }
52
        }
53
   }
```

2.3 Binary Search

MAKE SURE check(x) is monotone in [L,R)
MAKE SURE check(L) == TRUE AND check(R) == FALSE FIRST!

```
while (l + 1 < r) {
  int mid = (l + r) >> 1;
  if (check(mid)) l = mid;
  else r = mid;
}
return mid;
```

3 Graph Theroy

3.1 Prim - $O(N^2)$

```
#include <iostream>
   #include <cstdio>
3
   using namespace std;
5
   const int MAXN = 100;
   const int EXP = 10;
6
   const int INF = 1000000000;
7
9
   int nn;
10
   int map[MAXN+EXP][MAXN+EXP];
11
12
   int sum;
   bool inSet[MAXN+EXP];
13
14
   int dist[MAXN+EXP];
15
16
   void Prim(){
17
     sum = 0;
18
     for(int i = 1; i <= nn; i++) inSet[i] = 0, dist[i] = INF;
19
     dist[1] = 0;
20
     for(int i = 0; i < nn; i++){
```

```
21
        int min = INF, idx = 0;
22
        for(int j = 1; j \le nn; j++)
23
          if(!inSet[j] \&\& dist[j] < min)
24
            min = dist[j], idx = j;
25
        inSet[idx] = 1;
26
        sum += min;
        for(int j = 1; j \le nn; j++)
27
          if(!inSet[j] && dist[j] > map[idx][j])
28
29
            dist[j] = map[idx][j];
30
      }
31
   }
32
33
   int main(){
      while (scanf("%d\n",&nn) == 1 \&\& nn){
34
        for(int i = 1; i <= nn; i++)
35
36
          for(int j = 1; j <= nn; j++)
            scanf("%d",&map[i][j]);
37
38
        Prim():
39
        printf("%d\n",sum);
40
41
      return 0;
42
   }
```

3.2 Prim- O(MlogN)

```
#include <iostream>
2
   #include <cstdio>
 3
   #include <queue>
   using namespace std;
5
   const int MAXN = 100;
7
   const int MAXM = 10000:
   const int EXP = 10;
8
   const int INF = 1000000000;
10
11
   int nn,mm;
12
13
   int edges;
14
   struct EDGE{
15
      int n;
16
      int v;
     EDGE* nxt;
17
   } pool[MAXM*2+EXP];
18
19
   EDGE Ink[MAXN+EXP];
20
21
   void addEdge(int _f, int _t, int _v){
22
      pool[edges].n = _t;
23
      pool[edges].v = _v;
24
      pool[edges].nxt = lnk[_f].nxt;
25
      lnk[_f].nxt = &pool[edges];
26
      edges++;
27
   }
28
29
   struct NODE{
30
      int n;
31
      int dst;
32
     NODE(int _n = 0, int _dst = 0){
```

```
33
        n = _n;
34
        dst = _dst;
35
     }
36
   };
37
   bool operator <(NODE aa, NODE bb){</pre>
38
      return aa.dst > bb.dst;
39
   }
40
41
   int sum;
42
   bool inSet[MAXN+EXP];
43
   int dist[MAXN+EXP];
44
45
   void Prim_Prio(){
46
     sum = 0;
47
      for(int i = 1; i <= nn; i++) inSet[i] = 0, dist[i] = INF;
48
      dist[1] = 0;
      priority queue <NODE> Q; Q.push(NODE(1,0));
49
50
      while (Q. size ()) {
51
        NODE now = Q.top(); Q.pop();
52
        if(inSet[now.n]) continue;
53
        inSet[now.n] = 1;
54
        sum += now.dst;
55
        for(EDGE* tmp = Ink[now.n].nxt; tmp; tmp = tmp->nxt){
56
          if (!inSet[tmp->n] \&\& tmp->v < dist[tmp->n]) {
            dist[tmp->n] = tmp->v;
57
58
            Q.push(NODE(tmp->n,tmp->v));
59
          }
60
        }
61
62
      }
   }
63
64
65
   int main(){
66
      int cas; scanf("%d",&cas);
67
      while (cas --)
        scanf("%d%d", &nn, &mm);
68
69
        edges = 0;
70
        for(int i = 1; i \le nn; i++) lnk[i].nxt = 0;
71
        for (int i = 1; i \le mm; i++) {
          int aa,bb,vv; scanf("%d%d%d", &aa, &bb, &vv);
72
73
          addEdge(aa, bb, vv);
74
75
        Prim_Prio();
76
        printf("%d\n",sum);
77
      }
78
      return 0;
79
```

3.3 Kruskal -O(MlogM)

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

const int MAXN = 100;
const int MAXM = 10000;
```

```
const int EXP = 10;
   const int INF = 1000000000;
10
11
   int nn,mm;
12
13
   struct EDGE{
14
      int f;
15
      int t;
16
      int v;
17
   } pool[MAXM+EXP];
18
19
   bool cmp(EDGE a, EDGE b){
20
   return a.v < b.v;
21
   }
22
23
   int fa[MAXN+EXP];
24
   int find(int x){
25
     int r = x;
26
      while (r != fa[r]) r = fa[r];
27
      while (x != r){
28
        int tmp = fa[x];
29
        fa[x] = r;
30
       x = tmp;
31
32
      return r;
33
   }
34
35
   void uni(int aa, int bb){
36
     int xx = find(aa);
37
      int yy = find(bb);
38
      if(xx != yy) fa[yy] = xx;
39
   }
40
41
   int sum;
42
43
   void Kruskal(){
44
     sum = 0;
45
      sort(pool, pool+mm, cmp);
46
      for(int i = 1; i <= nn; i++) fa[i] = i;
      for (int i = 0; i < mm; i++) {
47
        int aa = find(pool[i].f);
48
49
        int bb = find(pool[i].t);
50
        if(aa == bb) continue;
51
       sum += pool[i].v;
52
        uni(aa, bb);
53
     }
   }
54
55
56
57
   int main(){
58
                  scanf("%d", &cas);
      int cas;
59
      while (cas --)
        scanf("%d%d", &nn, &mm);
60
61
        for (int i = 0; i < mm; i++)
          scanf("%d%d%d", &pool[i].f, &pool[i].t, &pool[i].v);
62
63
        Kruskal();
64
        printf("%d\n",sum);
65
     }
```

```
66 | return 0;
67 |}
```

3.4 Dijkstra - $O(N^2)$

```
1
   #include <iostream>
 2
   #include <cstdio>
 3
   #include <cstring>
   #include <queue>
   using namespace std;
7
   const int MAXN = 1000;
 8
   const int EXP = 10;
9
   const int INF = 1000000000;
10
11
   int nn;
12
   int mm;
13
14
   int map[MAXN][MAXN];
15
16
   int dist[MAXN+EXP];
17
   bool inSet[MAXN+EXP];
18
19
   void init(){
20
      for(int i = 0; i \le nn; i++)
21
        for(int j = 0; j <= nn; j++)
22
          map[i][j] = INF;
23
   }
24
   void Dijk(int s){
25
26
      for(int i = 1; i \le nn; i++){
27
        dist[i] = INF;
28
        inSet[i] = 0;
29
30
      dist[s] = 0;
31
      for(int i = 1; i \le nn; i++){
32
        int min = INF, idx = 0;
33
        for(int j = 1; j <= nn; j++){
34
          if (!inSet[j] && dist[j] < min) {
35
            min = dist[j];
36
            idx = j;
37
          }
38
39
        inSet[idx] = 1;
40
        for(int j = 1; j \le nn; j++){
41
          if (!inSet[j] && dist[idx] + map[idx][j] < dist[j])
42
            dist[j] = dist[idx] + map[idx][j];
43
        }
44
      }
45
   }
46
47
   int main(){
48
      int cas; scanf("%d", &cas);
49
      while (cas --){
        scanf("%d%d", &nn, &mm);
50
51
        init();
52
        for (int i = 1; i \le mm; i++) {
```

```
53
          int aa,bb,dd; scanf("%d%d%d", &aa, &bb, &dd);
          if(map[aa][bb] > dd){
54
55
            map[aa][bb] = map[bb][aa] = dd;
56
57
58
        Dijk(1);
59
        cout << dist[nn] << endl;
60
61
      return 0;
62
   }
```

3.5 Dijkstra - O(MlogN)

```
#include <iostream>
   #include <cstdio>
   #include <cstring>
   #include <queue>
   using namespace std;
 6
   const int MAXN = 50000;
7
8
   const int MAXM = 50000;
9
   const int EXP = 10;
10
   const int INF = 1000000000;
11
12 int edges;
13
   struct EDGE{
14
     int n;
15
     int d;
     EDGE *nxt;
16
17
   } pool[MAXM*2+EXP];
18 EDGE Ink[MAXN+EXP];
19
20
   void addEdge (int _f, int _t, int _d){
     pool[edges].n = _t;
21
     pool[edges].d= _d;
22
23
     pool[edges].nxt = lnk[_f].nxt;
24
     Ink[_f].nxt = &pool[edges];
25
     edges++;
   }
26
27
28
   int nn;
29
   int mm;
30
31
   int dist[MAXN+EXP];
32
   bool inSet[MAXN+EXP];
33
34
   struct NODE{
35
     int n;
36
     int dst;
     NODE(int _n = 0, int _dst = 0){
37
38
       n = _n;
        dst = _dst;
39
40
     }
41
   };
42
   | bool operator <(NODE aa, NODE bb){
43
44 return aa.dst > bb.dst;
```

```
45 | }
46
47
   void Dijk_Prio(int s){
48
      for(int i = 1; i \le nn; i++){
49
        dist[i] = INF;
50
        inSet[i] = 0;
51
      }
52
      priority_queue <NODE> Q;
53
      dist[s] = 0;
54
      Q.push(NODE(s, dist[s]));
55
      while (Q. size ()) {
56
        NODE now = Q.top(); Q.pop();
57
        if(inSet[now.n] == 1) continue;
        inSet[now.n] = 1;
58
        for(EDGE * tmp = lnk[now.n].nxt; tmp; tmp = tmp->nxt){
59
60
          if(!inSet[tmp->n] \& dist[now.n] + tmp->d < dist[tmp->n]){
61
            dist[tmp->n] = dist[now.n] + tmp->d;
62
            Q.push(NODE(tmp->n, dist[tmp->n]));
63
          }
64
        }
65
      }
   }
66
67
68
   int main(){
69
      int cas; scanf("%d", &cas);
      while ( cas --){
70
71
        edges = 0;
        scanf("%d%d", &nn, &mm);
72
73
        for(int i = 1; i \le nn; i++) lnk[i].nxt = 0;
74
        for (int i = 1; i \le mm; i++) {
75
          int aa,bb,dd; scanf("%d%d%d", &aa, &bb, &dd);
76
          addEdge(aa, bb, dd);
77
          addEdge(bb, aa, dd);
78
79
        Dijk_Prio(1);
80
        //cout << dist [?]
81
82
      return 0;
83
```

3.6 Dijkstra with heap

```
#include <cstdio>
2
   #include <cstring>
3
4
   using namespace std;
6
   const int maxN=1010;
7
   const int inf=2000000000;
8
9
   class DJ_heap {
10
   public:
11
      int data[maxN];
12
      int index[maxN];
13
      int pos[maxN];
14
      int tot;
15
     void init (int n, int st) {
```

```
for (int i = 2; i \le n; i++) {
16
          data[i] = inf;
17
18
          int now = (i == st ? 1 : i);
19
          index[i] = now;
20
          pos[now] = i;
21
22
        data[1] = 0;
23
        index[1] = st;
24
        pos[st] = 1;
25
        tot = n;
26
27
      void fix_down(int x) {
28
        for (int son = x + x; son <= tot; x = son, son = x + x) {
29
          if (son < tot && data[son+1] < data[son])</pre>
30
            son++;
31
          if (data[x] > data[son]) {
            int tmp=data[x]; data[x]=data[son]; data[son]=tmp;
32
33
            tmp=index[x]; index[x]=index[son]; index[son]=tmp;
34
            pos[index[x]]=x;
35
            pos[index[son]]=son;
36
          }
37
        }
38
39
      void fix_up(int x) {
        for (int fa = x>>1; x > 1; x = fa, fa = x>>1) {
40
41
          if (data[fa] > data[x]) {
42
            int tmp=data[fa]; data[fa]=data[x]; data[x]=tmp;
43
            tmp=index[fa]; index[fa]=index[x]; index[x]=tmp;
44
            pos[index[x]]=x;
45
            pos[index[fa]]=fa;
46
          }
47
        }
      }
48
49
      void change(int x, int newdata) {
50
        data[pos[x]]=newdata;
51
        fix_up(pos[x]);
52
53
      void pop(int &x,int &dist) {
54
        x=index[1];
55
        dist=data[1];
        index[1]=index[tot];
56
57
        data[1]=data[tot];
58
        pos[x]=0;
59
        pos[index[tot --]]=1;
60
        fix_down(1);
61
      bool empty() {
62
63
        return tot == 0;
64
      }
65
   };
66
67
   int a[1010][2000];
   int b[1010][2000];
69
   int dist[1010];
70
   bool visit [1010];
71
   DJ_heap q;
72
73
```

```
74 | int main() {
75
       int n,m;
76
       scanf("%d%d",&m,&n);
       while (m--) {
77
78
         int f,t,cost;
         scanf("%d%d%d",&f,&t,&cost);
79
80
         a[f][++a[f][0]]=t;
         b[f][++b[f][0]]=cost;
81
82
         a[t][++a[t][0]] = f;
83
         b[t][++b[t][0]] = cost;
84
85
       memset(dist,64,sizeof(dist));
86
       q.init(n,n);
87
       dist[n]=0;
88
       while (!q.empty()&&!visit[1]) {
89
         int v,d;
90
         q.pop(v,d);
91
         for (int i=1; i \le a[v][0]; i++)
92
           if \ (!\ visit[a[v][i]] \ \&\& \ dist[a[v][i]] \ > \ dist[v]+b[v][i]) \ \{
93
              dist[a[v][i]] = dist[v] + b[v][i];
94
              q.change(a[v][i], dist[a[v][i]]);
95
           }
96
         visit[v]=1;
97
       printf("%d\n", dist[1]);
98
99
       return 0;
100
    }
```

3.7 Bellman-Ford

```
#include <iostream>
2
   #include <cstdio>
3
4
   using namespace std;
5
6
   const int MAXN = 1000;
7
   const int MAXM = 2000;
   const int EXP = 10;
9
   const int INF = 1000000000;
10
11
   int mm, nn;
12
13
   int vf[MAXM+EXP], vt[MAXM+EXP], vc[MAXM+EXP]; 记录边
                                                            //
14
15
   int dist[MAXN+EXP];
16
17
   void init(){
18
     scanf("%d%d",&nn,&mm);
19
     for (int i = 0; i < mm; i++) {
        scanf("%d%d%d", vf+i, vt+i, vc+i);
20
21
     }
   }
22
23
24
   void Bellman_Ford(int s){
25
     for(int i = 1; i <= nn; i++)
                                       dist[i] = INF;
26
      dist[s]=0;
27
      for(int i = 0; i < nn-1; i++){
```

```
28
        for (int i = 0; i < mm; i++) {
           if(dist[vf[i]] + vc[i] < dist[vt[i]]){</pre>
29
30
             dist[vt[i]] = dist[vf[i]] + vc[i];
31
32
           if(dist[vt[i]] + vc[i] < dist[vf[i]]){</pre>
33
             dist[vf[i]] = dist[vt[i]] + vc[i];
34
35
        }
36
      }
   }
37
38
39
   int main(){
40
      init();
41
      Bellman_Ford(1);
42
      printf("%d\n", dist[nn]);
43
      return 0;
44
   }
```

3.8 Shortest Path Faster Algorithm

```
#include <iostream>
2
   #include <cstdio>
   #include <cstring>
   #include <queue>
   using namespace std;
7
   const int MAXN = 50000;
   const int MAXM = 50000;
   const int EXP = 10;
10
   const int INF = 1000000000;
11
12
   int edges;
13
   struct EDGE{
14
      int n;
15
      int d;
16
      EDGE *nxt;
17
   }pool[MAXM*2+EXP];
18
   EDGE Ink[MAXN+EXP];
19
20
   void addEdge (int _f, int _t, int _d){
21
      pool[edges].n = _t;
      pool[edges].d= _d;
pool[edges].nxt = lnk[_f].nxt;
22
23
24
      Ink[_f].nxt = &pool[edges];
25
      edges++;
26
   }
27
28
   int nn;
29
   int mm;
30
31
   bool inQ[MAXN+EXP];
   int dist[MAXN+EXP];
32
33
34
   void spfa(int s){
35
      for(int i = 0; i \le nn; i++){
        inQ[i] = 0;
36
        dist[i] = INF;
37
```

```
38
39
      queue<int> Q; Q.push(s);
40
      inQ[s] = 1; dist[s] = 0;
41
      while (Q. size ()) {
42
        int now = Q.front(); Q.pop();
43
        inQ[now] = 0;
44
        for(EDGE* tmp = lnk[now].nxt; tmp; tmp = tmp->nxt){
45
          if(dist[now] + tmp->d < dist[tmp->n]){
46
            dist[tmp->n] = dist[now] + tmp->d;
            if (!inQ[tmp->n]) {
47
48
              Q.push(tmp->n);
              inQ[tmp->n] = 1;
49
50
51
          }
52
        }
53
      }
   }
54
55
56
   int main(){
57
      int cas; scanf("%d", &cas);
58
      while (cas --)
59
        edges = 0;
        scanf("%d%d", &nn, &mm);
60
61
        for(int i = 1; i \le nn; i++) lnk[i].nxt = 0;
        for (int i = 1; i \le mm; i++) {
62
63
          int aa,bb,dd; scanf("%d%d%d", &aa, &bb, &dd);
64
          addEdge(aa, bb, dd);
65
          addEdge(bb, aa, dd);
66
67
        spfa(1);
68
        //cout << dist [?]
69
70
      return 0;
71
```

3.9 Network Flow - ISAP[NON-ORIGINAL]

```
#include <cstring>
1
   #include <cstdio>
3
   #include <queue>
   #include <algorithm>
5
   #include <vector>
7
   using namespace std;
8
   const int MAXN = 210;
9
   const int MAXM=500010;
11
   const int inf = 2E9;
12
13
   typedef struct {int v,next,val;} edge;
14
   struct SAP {
15
     edge e[MAXM];
16
     int p[MAXN], eid;
17
     inline void clear() {memset(p, -1, size of(p)); eid = 0;}
18
      inline void insert1(int from, int to, int val) {
19
        e[eid].v=to;
        e[eid].val=val;
20
```

```
21
        e[eid].next=p[from];
22
        p[from]=eid++;
23
        swap(from, to);
24
        e[eid].v=to;
25
        e[eid].val=0;
26
        e[eid].next=p[from];
27
        p[from]=eid++;
28
29
      inline void insert2(int from, int to, int val) {
30
        e[eid].v=to;
31
        e[eid].val=val;
32
        e[eid].next=p[from];
33
        p[from]=eid++;
34
        swap(from, to);
35
        e[eid].v=to;
36
        e[eid].val=val;
37
        e[eid].next=p[from];
38
        p[from] = eid ++;
39
40
      int n;
41
      int h[MAXN];
42
      int gap[MAXN];
43
      int source, sink;
44
      inline int dfs(int pos,int cost) {
45
        if (pos==sink) {
46
          return cost;
47
48
        int j, minh=n-1, lv=cost, d;
49
        for (j=p[pos]; j!=-1; j=e[j].next) {
50
          int v=e[j].v,val=e[j].val;
51
           if (val > 0) {
52
             if (h[v]+1==h[pos]) {
53
               if (lv <e[j].val) d=lv;</pre>
54
               else d=e[j].val;
55
               d=dfs(v,d);
56
               e[j].val-=d;
               e[j ^1]. val+=d;
57
               lv = d;
58
59
               if (h[source]>=n) return cost-lv;
60
               if (lv == 0) break;
61
62
             if (h[v]<minh)</pre>
                               minh=h[v];
63
          }
64
        }
65
        if (lv==cost) {
66
           —gap[h[pos]];
67
          if (gap[h[pos]]==0) h[source]=n;
68
          h[pos]=minh+1;
69
          ++gap[h[pos]];
70
71
        return cost-lv;
72
      int run() {
73
74
        int ret=0;
75
        memset(gap, 0, size of (gap));
76
        memset(h,0,sizeof(h));
77
        gap[source]=n;
78
        while (h[source]<n) ret+=dfs(source,inf);
```

```
79
        return ret;
80
81
   } solver;
82
83
   int main() {
84
      int N,M;
85
      while (scanf("%d%d",&M,&N)!=EOF) {
86
        solver.source = 1;
87
        solver.sink = N;
88
        solver.n = N;
89
        solver.clear();
90
        while (M--) {
91
          int f,t,w;
92
          scanf("%d%d%d",&f,&t,&w);
93
          solver.insert1(f,t,w);
94
95
        printf("%d\n", solver.run());
96
97
      return 0;
98
```

3.10 Bipartite Graph Matching

```
#include <cstdio>
2
   #include <cstring>
3
4
   bool adj[555][555];
5
   bool visit [555];
6
   int match[555];
7
   int n;
8
9
   bool dfs(int now) {
10
        for (int i = 1; i \le n; i++) {
11
            if (visit[i] == false && adj[now][i]) {
                 visit[i] = true;
12
13
                 int tt = match[i];
14
                 match[i] = now;
15
                 if (tt == -1 \mid | dfs(tt)) return true;
16
                 match[i] = tt;
17
            }
18
19
        return false;
20
   }
21
22
   int main() {
23
        int m;
24
        scanf("%d%d",&n,&m);
25
        for (int i = 0; i < m; i++) {
26
            int f,t; scanf("%d%d",&f,&t);
27
            adi[f][t] = true;
28
        }
29
        int ans = 0;
30
        memset(match, 0 xff, size of (match));
31
        for (int i = 1; i \le n; i++) {
32
            memset(visit,0,sizeof(visit));
33
            if (dfs(i)) ans ++;
34
        }
```

3.11 Minimun Cost Flow [TO BE TESTED!]

```
1
   #include <iostream>
   #include <queue>
3
   #include <cstring>
    using namespace std;
 7
    int n,m,ans,t,f;
    int maxf[210][210], flow [210][210], cost [210][210];
8
9
   int fa[210], dist[210];
10
   bool inque[210];
11
12
    inline int abs(int a) {return a > 0 ? a : -a ;}
13
    void init() {
      int a[210][2]={0},b[210][2]={0},s=0,sa=0,sb=0;
14
      memset(maxf,0,sizeof(maxf));
15
16
      memset(flow,0,sizeof(flow));
17
      memset(cost, 0, size of(cost));
18
      for (int i=1;i <=n;i++){}
19
        for (int j=1;j <=m;j++) {
20
          char tt;
21
          cin >> tt;
22
          if (tt == 'H') {
23
            a[++sa][0]=i;
24
            a[sa][1]=j;
25
          }
26
          if (tt == 'm') {
27
            b[++sb][0]=i;
28
            b[sb][1]=j;
29
          }
30
        }
31
      }
32
      s=sa;
33
      for (int i = 1; i \le s; i++) {
34
        for (int j = 1; j <= s; j++) {
35
          cost[i][s+j] = abs(a[i][0]-b[j][0])+abs(a[i][1]-b[j][1]);
36
          cost[s+i][i] = cost[i][s+i];
37
          maxf[i][s+j] = 1;
38
        }
39
40
      for (int i = 1; i \le s; i++)
41
        \max[0][i] = \max[s+i][s+s+1] = 1;
42
      n = t = s + s + 1;
43
      f = 0;
44
      ans = 0;
45
   }
46
47
    inline int value(int i,int j) {
48
      return flow[j][i] > 0 ? -cost[i][j] : cost[i][j];
49
   }
50
51 | bool spfamark() {
```

```
52
      memset(fa,0,sizeof(fa));
53
      memset(inque, 0, size of (inque));
54
      memset(dist, 0x3f, sizeof(dist));
55
      queue<int> q;
56
      q.push(f); inque[f] = true; dist[f]=0;
57
      while (!q.empty()) {
58
        int now = q.front(); q.pop(); inque[now] = false;
        for (int i = 0; i \le n; i + +)
59
           if ((maxf[now][i] - flow[now][i] > 0)
    && dist[now] + value(now,i) < dist[i]){</pre>
60
61
             dist[i] = dist[now] + value(now,i);
62
63
             fa[i] = now;
64
             if (!inque[i]) {
65
               inque[i]=1;
66
               q.push(i);
67
             }
68
           }
69
70
      return dist[t] != 0x3f3f3f3f;
71
   }
72
73
74
    int main() {
75
      while (cin >> n >> m && n && m) {
76
        init();
        while (spfamark()) {
77
78
           for(int i = t; i != f; i = fa[i]) {
79
             ans+=value(fa[i],i);
80
             flow[fa[i]][i]++;
81
             flow[i][fa[i]]--;
82
           }
83
84
        cout << ans << endl;
85
86
      return 0;
87
   }
```

3.12 Kuhn-Munkras [NON-ORIGINAL]

refined from http://blog.sina.com.cn/s/blog_6ec5c2d00100vt8d.html

```
class KM_class {
2
    private:
3
      int match[maxm];
4
      int lx [maxn];
5
      int ly [maxm];
6
      bool vis_x [maxn];
7
      bool vis_y [maxm];
8
      int slack;
9
10
   public:
11
      bool DFS(int u) {
12
        vis_x[u] = true;
13
        int tmp;
14
        for (int v = 1; v \le M; v++) {
15
          tmp = Ix[u] + Iy[v] - W[u][v];
          if (tmp == 0) {
16
17
             if (! vis_y[v]) {
```

```
18
               vis_y[v] = true;
19
               if(match[v] == 0 \mid\mid DFS(match[v])) {
20
                 match[v] = u;
                 return true;
21
22
               }
23
            }
24
          } else {
            slack = min(slack,tmp);
25
26
27
28
        return false;
29
30
31
      int KM() {
32
        memset(match, 0, size of (match));
33
        memset(ly,0,sizeof(ly));
34
        for(int u = 1; u \le N; u++) {
35
          lx[u] = W[u][1];
36
          for(int v = 2; v \le M; v++) {
37
            lx[u] = max(lx[u],W[u][v]);
38
          }
39
        }
40
41
        for(int u = 1; u \le N; u++) {
          while(1) {
42
43
            slack = INT_MAX;
44
            memset(vis_x,0,sizeof(vis_x));
45
            memset(vis_y,0,sizeof(vis_y));
46
            if(DFS(u)) break;
47
            for(int i = 1; i \le N; i++)
48
               if(vis_x[i])
49
                 lx[i] -= slack;
            for(int i = 1; i \le M; i++)
50
51
               if(vis_y[i])
52
                 ly[i] += slack;
53
          }
54
        }
55
        int sum = 0;
56
        for (int v = 1; v \le M; v++) sum += W[match[v]][v];
57
        return -sum;
58
59
   } km;
```

3.13 Cut Vetrix and Edge

```
#include <cstdio>
   #include <cstring>
3
   #include <algorithm>
4
5
   using namespace std;
6
7
   bool a[110][110];
8
   int visit [110];
9
   int deep[110];
   int back[110];
   bool cut[110];
11
12 | int n, ans;
```

```
13
14
15
    void dfs(int k,int fa,int d) {
16
        visit[k]=1;
17
        back[k]=deep[k]=d;
18
        int tot=0;
19
        for (int i=1;i<=n;i++) {
20
            if (a[k][i] && i!=fa && visit[i]==1)
21
                 back[k]=min(back[k],deep[i]);
22
             if (a[k][i] && visit[i]==0) {
23
                 dfs(i,k,d+1);
24
                 tot++;
25
                 back[k]=min(back[k],back[i]);
26
                 if ((k==1 &&tot>1) || (k!=1 && back[i]>=deep[k]))
27
                     if (!cut[k]) {
28
                         cut[k]=1;
29
                         ans++;
30
31
               //if back[i]>deep[k] k,i is bridge;
32
            }
33
34
        visit[k]=2;
35
   }
36
    int main() {
37
38
        while (1) {
39
            scanf("%d",&n);
40
            if (n==0)
41
                 break;
42
            memset(a,0,sizeof(a));
43
            memset(back, 0, size of (back));
44
            memset(cut,0,sizeof(cut));
45
            memset(deep, 0, size of (deep));
46
            memset(visit,0,sizeof(visit));
47
            ans=0;
48
            int f;
49
            while (scanf("%d",&f) && f>0)
50
                 while (getchar()!=10) {
51
                     int t;
                     scanf("%d",&t);
52
53
                     a[f][t]=a[t][f]=1;
54
                 }
55
56
            dfs(1,0,0);
57
            printf("%d\n",ans);
58
59
        return 0;
60
```

3.14 Strongly Connected Components

```
#include <cstdio>
#include <cstring>
#include <stack>
#include <vector>

using namespace std;
```

```
7
8
   vector<int> a[10010];
9
   vector<int> b[10010];
10
   stack<int> tt;
11
   int fa[10010];
   int d[10010];
   int size[10010];
13
14
   bool visit [10010];
15
   int n,m;
16
17
   void dfs(int k) {
18
      visit[k]=1;
19
      for (int i=0;i<a[k].size();i++)
20
        if (! visit[a[k][i]])
21
          dfs(a[k][i]);
22
      tt.push(k);
23
   }
24
25
   void dfs(int k, int FA) {
26
      fa[k]=FA;
27
      size[FA]++;
28
      visit[k]=1;
29
      for (int i=0;i<b[k].size();i++)
30
        if (! visit[b[k][i]])
31
          dfs(b[k][i],FA);
32
   }
33
34
    int main() {
      scanf("%d%d",&n,&m);
35
36
      for (int i=0; i < m; i++) {
37
        int f,t;
38
        scanf("%d%d",&f,&t);
39
        a[f].push_back(t);
40
        b[t].push_back(f);
41
42
      memset(visit, 0, size of (visit));
43
      for (int i=1;i<=n;i++)
44
        if (!visit[i])
45
          dfs(i);
46
      memset(visit, 0, size of (visit));
47
      int s=0;
48
      for (;! tt.empty(); tt.pop())
49
        if (!visit[tt.top()])
50
          dfs(tt.top(),++s);
51
      for (int i=1;i<=n;i++) {
        int f=fa[i];
52
53
        for (int j=0;j<b[i].size();j++) {
54
          int t=fa[b[i][j]];
55
          if (f!=t)
56
            d[t]++;
57
        }
58
      }
59
      vector<int> ans;
60
      for (int i=1;i<=s;i++)
61
        if (d[i]==0)
62
          ans.push_back(size[i]);
63
      if (ans.size() == 1)
64
        printf("%d\n", ans[0]);
```

```
65 else
66 puts("0");
67 return 0;
68 }
```

4 String Algorithm

4.1 ELF Hash

```
int elfhash(char *key) {
1
      unsigned int h = 0;
2
3
      while(*key) {
4
       h = (h << 4) + *key++;
5
        unsigned int g=h&0Xf0000000L;
        if (g) h ^= g >> 24;
7
       h &= \sim g;
8
9
      return h%MOD;
10
```

4.2 Aho-Corasick Automation

```
#include <cstdio>
   #include <cstring>
 2
   #include <queue>
 5
   using std::queue;
 6
7
   void toInt(char s[]) {
 8
      for (int i = 0; s[i]; i++) {
 9
        if (s[i]=='A') s[i]='0';
        if (s[i]=='G') s[i]='1';
10
        if (s[i]=='T') s[i]='2';
11
12
        if (s[i]=='C') s[i]='3';
13
14
   }
15
16
   struct trie {
17
      trie *next[4];
      trie *fail;
18
      bool isend;
19
20
   };
21
22
   trie pool[1010];
   trie *head;
24
   trie *root;
25
26
   void insert(char s[]) {
27
      trie *now=root;
      for (;;) {
28
29
        if (s[0]==0) {
30
          now->isend=1;
31
          return;
32
        }
```

```
33
        int tt=s[0]-'0';
34
        if (now->next[tt]==NULL)
35
          now->next[tt]=++head;
36
        now=now->next[tt];
37
        s++;
38
      }
39
   }
40
41
    void buildFaliure() {
      queue<trie*> q;
42
      for (int i=0; i<4; i++)
43
44
        if (root->next[i])
45
          root->next[i]->fail=root;
46
          q.push(root->next[i]);
47
        } else root->next[i]=root;
48
      while (!q.empty()) {
49
        trie *now=q.front(); q.pop();
50
        for (int i=0; i<4; i++) {
51
          trie *u=now->next[i];
52
          if (u) {
53
            q.push(u);
54
            trie *v=now->fail;
            while (v->next[i]==NULL)
55
56
              v=v->fail;
57
            u->fail=v->next[i];
58
59
60
        if (now->fail ->isend) now->isend=1;
61
62
   }
63
64
   int dp[1010][1010];
65
66
    trie* go(trie *now, char ch) {
67
      ch -= '0';
      trie *ans = now;
68
69
      while (ans -> next[ch] == NULL)
70
        ans = ans -> fail;
71
      return ans -> next[ch];
72
   }
73
74
    int main() {
75
      int ii = 1;
76
      for (;;) {
77
        int n;
        scanf("%d",&n);
78
        if (n==0) break;
79
80
        root=head=pool;
81
        memset(dp,0x7f, size of (dp));
82
        memset(pool, 0, size of (pool));
83
84
        static char buf[30];
85
        for (int i=0; i < n; i++) {
          scanf("%s",buf);
86
87
          toInt(buf);
88
          insert(buf);
89
90
        buildFaliure();
```

```
91
 92
         static char word[1010];
 93
         scanf("%s",word);
 94
         toInt(word);
 95
         dp[0][0]=0;
 96
         n=head_pool;
 97
         int len;
 98
         for (len=0;word[len];len++) {
99
           for (int j=0; j < n; j++)
100
             if (dp[i][len]<=len) {
                for (char ch='0';ch<='3';ch++) {
101
                  trie *tmp=go(pool+j,ch);
102
103
                  if (tmp->isend) continue;
104
                  int next=tmp-pool;
105
                  int delta=0; if (ch!=word[len]) delta++;
106
                  if (dp[next][len+1] > dp[j][len] + delta)
107
                    dp[next][len+1] = dp[j][len] + delta;
108
               }
109
             }
110
         int ans=2000000000;
111
112
         for (int j=0; j < n; j++)
113
           if (dp[j][len]<ans) ans=dp[j][len];</pre>
114
         if (ans>len) ans=-1;
         printf("Case %d: %d\n", ii++, ans);
115
116
117
       return 0;
118
    }
```

4.3 Suffix array

```
int wa[maxn],wb[maxn],wv[maxn],ws[maxn];
    int cmp(int *r,int a,int b,int l)
 2
 3
    \{return \ r[a]==r[b]\&\&r[a+l]==r[b+l];\}
 4
    void da(int *r,int *sa,int n,int m) {
 5
      int i, j, p, *x=wa, *y=wb, *t;
 6
      for (i=0; i \le m; i++) ws [i]=0;
 7
      for(i=0; i < n; i++) ws[x[i]=r[i]]++;
 8
      for (i=1; i \le m; i++) ws [i]+=ws[i-1];
 9
      for (i=n-1; i \ge 0; i--) sa[--ws[x[i]]] = i;
10
      for(j=1,p=1;p<n;j*=2,m=p) {
11
         for (p=0, i=n-j; i < n; i++) y[p++]=i;
12
         for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
13
         for (i=0; i < n; i++) wv[i]=x[y[i]];
14
         for (i=0; i \le m; i++) ws [i]=0;
15
         for (i = 0; i < n; i++) ws [wv[i]]++;
16
         for (i=1; i \le m; i++) ws [i]+=ws[i-1];
17
         for (i=n-1; i \ge 0; i--) sa[--ws[wv[i]]] = y[i];
18
         for (t=x, x=y, y=t, p=1, x[sa[0]]=0, i=1; i < n; i++)
19
           x[sa[i]] = cmp(y, sa[i-1], sa[i], j)?p-1:p++;
20
      }
21
    }
```

5 Data Struct

5.1 Binary Indexed Tree

BECAREFUL WHILE I == 0 !!!

```
1
   int sum(int k) {
2
     int ans = 0;
     for (int i = k; i > 0; i = i & -i)
3
4
        ans += a[i];
5
     return ans;
   }
6
7
   void change(int k, int n, int delta) {
     for (int i = k; i \le n; i + = i \& -i)
10
        a[i] += delta;
11
```

5.2 Inversion

```
#include <cstdio>
2
   int a[5000101:
3
   int t[500010];
4
   long long ans;
 7
   void merge(int a[],int sizea,int b[],int sizeb) {
8
        int nowa = 0;
9
        int nowb = 0;
10
        int s = 0;
11
        while (nowa < sizea && nowb < sizeb) {
12
            if (a[nowa]<=b[nowb])</pre>
13
                 t[s++]=a[nowa++];
14
15
            if (a[nowa]>b[nowb]) {
16
                 t[s++] = b[nowb++];
17
                 ans += sizea - nowa;
18
            }
19
        }
20
        while (nowa<sizea)
21
            t[s++]=a[nowa++];
22
        while (nowb<sizeb)
23
            t[s++]=b[nowb++];
24
   }
25
   void sort(int a[],int size) {
27
        if (size < 2)
28
            return;
29
        int lsize = size >> 1;
30
        int rsize = size-lsize;
31
        sort(a, lsize);
32
        sort(a + lsize , rsize);
33
        merge(a, lsize, a+lsize, rsize);
34
        for (int i = 0; i < size; i++)
35
            a[i] = t[i];
   }
36
37
```

```
38
39
    int main() {
40
         while (1) {
41
             int n;
42
             scanf("%d",&n);
43
             if (!n)
44
                  break;
45
             for (int i=0; i < n; i++)
                  scanf("%d",a+i);
46
47
             ans = 0;
48
             sort(a,n);
49
             printf("% | Ild \n", ans);
50
         }
51
    }
```

5.3 BigInt Multiply with FFT

```
#include <cstdio>
   #include <cstring>
 2
 3
   #include <cmath>
 4
 5
   typedef long long Long;
   const int MAXN=32768;
7
   const double pi=acos(-1.0);
   const Long MOD=100000;
9
   const int TEN=5;
10
11
   double ra[MAXN];
12
   double ia [MAXN];
13
   double rb[MAXN];
14
   double ib[MAXN];
15
   double rc[MAXN];
16
   double ic [MAXN];
   char a[MAXN];
17
18
   char b[MAXN];
19
   int slena;
20
   int slenb;
21
   int lena;
   int lenb;
   int n, logn;
24
   Long ans [MAXN];
25
26
   int rev(int x, int bit)
27
   {
28
            int ans=0;
29
            for (int i=0;i<bit;i++)
30
            {
31
                     ans < <= 1;
32
                     if (x&1) ans |=1;
33
                     x >> = 1;
34
35
            return ans;
36
   }
37
   void fft(double ir[],double ii[],int size,int mark)
39
40
            static double R[MAXN];
```

```
41
            static double I[MAXN];
42
            double delta=mark*2*pi;
43
            for (int i=0;i<size;i++)
44
            {
45
                     int tt=rev(i,logn);
46
                     R[tt]=ir[i];
47
                     I[tt]=ii[i];
48
49
            for (int s=1; s <= logn; s++)
50
                     int m=1<<s;
51
                     double rwm=cos(delta/m);
52
53
                     double iwm=sin(delta/m);
54
                     for (int k=0; k< n; k+=m)
55
56
                              double rw=1;
57
                              double iw=0;
58
                              for (int j=0; j < m/2; j++)
59
                              {
60
                                       // t=w*A[k+j+m/2];
61
                                       double rt=rw*R[k+j+m/2]-iw*I[k+j+m/2];
62
                                       double it=rw*I[k+j+m/2]+iw*R[k+j+m/2];
                                       // u=A[k+j];
63
64
                                       double ru=R[k+j];
                                       double iu=I[k+j];
65
66
67
                                       // A[k+j]=u+t;
68
                                       R[k+j]=ru+rt;
69
                                       I[k+j]=iu+it;
70
                                       //A[k+j+m/2]=u-t;
71
72
                                       R[k+j+m/2]=ru-rt;
73
                                       I[k+j+m/2]=iu-it;
74
75
                                       double rnw=rw*rwm-iw*iwm;
76
                                       double inw=rw*iwm+iw*rwm;
77
                                       rw=rnw; iw=inw;
78
                              }
79
                     }
80
            for (int i=0;i<size;i++)
81
82
            {
83
                     ir[i]=R[i];
84
                      ii[i]=I[i];
85
            }
86
   }
87
88
   double POW[10] = {1,1E1,1E2,1E3,1E4,1E5,1E6,1E7,1E8,1E9};
89
90
   int next(char str[])
91
   {
92
            int len=0;
93
            for (str[len]=getchar(); str[len]>='0'; str[len]=getchar())
94
                     len++;
95
             str[len]=0;
96
            return len;
97
   }
98
```

```
99
     int main()
100
101
              int nn=0;
102
              scanf("%d",&nn); getchar();
103
              while (nn--)
104
              {
105
                       memset(ra, 0, n < < 3);
106
                       memset(ia, 0, n < < 3);
                       memset(rb, 0, n < < 3);
107
108
                       memset(ib, 0, n << 3);
109
                       memset(ans, 0, n << 3);
110
111
                       slena=next(a);
112
                       int cnt=0; lena=0;
113
                       for (int j=slena-1; j>=0; j--)
114
115
                                ra[lena]=ra[lena]+(a[i]-'0')*POW[cnt++];
116
                                if (cnt==TEN) {lena++; cnt=0;}
117
118
                       if (ra[lena] > 0.1)
                                                  lena++;
119
120
                       slenb=next(b);
                       cnt=0; lenb=0;
121
122
                       for (int j=slenb-1; j>=0; j--)
123
124
                                rb[lenb]=rb[lenb]+(b[j]-'0')*POW[cnt++];
125
                                if (cnt==TEN) {lenb++; cnt=0;}
126
127
                       if (rb[lenb] > 0.1)
                                                  lenb++;
128
129
                       n=1; log n = 0;
130
                       while (n<lena || n<lenb) {n+=n;logn++;}
131
                       n+=n; logn++;
132
133
                       fft(ra,ia,n,1);
134
                       fft(rb, ib, n, 1);
135
                       for (int i=0; i < n; i++)
136
                       {
137
                                rc[i]=ra[i]*rb[i]-ia[i]*ib[i];
138
                                ic[i]=ra[i]*ib[i]+rb[i]*ia[i];
139
                       fft(rc,ic,n,-1);
140
141
                       for (int i=0; i < n; i++)
142
                                ans[i]=(Long)(rc[i]/n+0.5);
143
                       for (int i=0; i < n-1; i++)
144
145
                                ans [i+1]+=ans [i]/MOD;
146
                                ans[i]%=MOD;
147
148
                       bool print=0;
149
                       for (int i=n-1; i \ge 0; i--)
150
151
                                if (!print && (ans[i]>0 || i==0))
152
153
                                          print=1;
                                          printf("% | Ild ", ans[i]);
154
155
                                } else
156
                                if (print)
```

6 Computational Geometry

6.1 Constants

```
const double eps=1e-12;
const double pi=acos(-1.0f);
const double INF=1.0/0.0f;
const double NEGINF=-1.0/0.0f;
```

6.2 Compare Function

```
inline bool equ0(const double& x){return fabs(x)<eps;}
inline bool equ(const double& x,const double& y){return fabs(x-y)<eps
;}
inline bool ls(const double& x,const double& y){return x+eps<y;}
inline bool gr(const double& x,const double& y){return x-eps>y;}
inline bool greq(const double& x,const double& y){return x+eps>=y;}
inline bool lseq(const double& x,const double& y){return x-eps<=y;}</pre>
```

6.3 3D Point and Vector

```
1
      struct Point{
 2
        double x,y,z;
        explicit Point(const double& a=0,const double& b=0,const double& c
 3
           =0):x(a),y(b),z(c){}
        Point(const Point& p):x(p.x),y(p.y),z(p.z){}
 4
 5
        Point operator -() const{return Point(-x,-y,-z);}
 6
        Point operator - (const Point& a) const{return Point(x-a.x,y-a.y,z-a.
           z);}
 7
        Point operator+(const Point& a) const{return Point(x+a.x,y+a.y,z+a.
        Point operator*(const double& s) const{return Point(s*x,s*y,s*z);}
 8
        double sqlen() const{return x*x+y*y+z*z;}
9
10
        double len() const{return sqrt(sqlen());}
11
        void norm() {double l=len(); x/=l; y/=l; z/=l;}
12
      };
13
      typedef Point Vec;
```

6.4 3D Utility Functions

6.4.1 3D dot and 3D vector cross

```
inline double dot(const Vec& u, const Vec& v)
{return u.x*v.x + u.y*v.y + u.z*v.z;}
inline Vec cross(const Vec& u, const Vec& v){
  return Vec(u.y*v.z-u.z*v.y,u.z*v.x-u.x*v.z,u.x*v.y-u.y*v.x);
}
```

6.4.2 3D line - line min distance

POJ 2852

```
1
      inline bool line mindist(const Point& p0, const Vec& d0,
      const Point& p1, const Vec& d1,
2
3
      double* t, double* k){
4
        double a=dot(d0,d0), b=-dot(d0,d1), c=-dot(d1,d1), d=dot(d0,p1-p0),
5
        e = dot(d1, p1-p0);
        double det=a*c+b*b;
6
        if(equ0(det)) return false;
7
8
        t = (d*c-b*e)/det;
9
        *k=(a*e+b*d)/det;
10
        return true;
11
      }
```

6.4.3 Sphere coord

```
inline double deg2grad(const double& d){return d*pi/180.0;}
inline Vec sphere_coord(const double& ele,const double& az){
  double x,y,z,r;
  z=sin(ele);
  r=cos(ele);x=r*cos(az);y=r*sin(az);
  return Vec(x,y,z);
}
```

6.5 2D Point and Vector

```
1
     struct Point{
2
       double x,y;
3
        explicit Point(const double& a=0,const double& b=0):x(a),y(b){}
4
       Point(const Point& p):x(p.x),y(p.y){}
       Point& operator=(const Point& p){x=p.x;y=p.y;return *this;}
5
       Point operator -() const{return Point(-x,-y);}
6
7
       Point operator—(const Point& a) const{return Point(x-a.x,y-a.y);}
8
       Point operator+(const Point& a) const{return Point(x+a.x,y+a.y);}
9
       Point operator*(const double& s) const{return Point(s*x,s*y);}
10
       double sqlen() const{return x*x+y*y;}
11
       double len() const{return sqrt(sqlen());}
       void norm(){double l=len();x/=l;y/=l;}
12
13
     typedef Point Vec;
14
```

6.6 2D Utility Functions

6.6.1 2D dot and 2D scalar cross

```
inline double dot(const Vec& u,const Vec& v){return u.x*v.x + u.y*v.y
;}
inline double cross(const Vec& u,const Vec& v){return u.x*v.y - u.y*v
.x;}
```

6.6.2 2D triple point coline

```
inline bool coline(const Point& a,const Point& b,const Point& c){
   Vec u=b-a,v=c-a;
   return equ0(cross(u,v));
}
```

6.6.3 2D segment - segment overlap

```
1
      inline bool seg_overlap(const Point& a1, const Point& a2, const Point&
         b1, const Point& b2){
2
        if(coline(a1,a2,b1) && coline(a1,a2,b2)){
          double k1, k2;
3
4
          Vec axis=a2-a1;
5
          Vec temp=b1-a1;
6
          if(equ0(axis.y)){
7
            k1=temp.x/axis.x;
8
            temp=b2-a1;
9
            k2=temp.x/axis.x;
10
          }
11
          else {
            k1=temp.y/axis.y;
12
13
            temp=b2-a1;
14
            k2=temp.y/axis.y;
15
16
          if((gr(k1,1.0f) \&\& gr(k2,1.0f)) \mid | (ls(k1,0.0f) \&\& ls(k2,0.0f)))
              return false;
17
          else return true;
18
19
        else return false;
20
      }
```

6.6.4 2D angle difference between two vectors

```
//CCW for positive
inline double angle_diff(const Vec& from, const Vec& to){
   Vec nfrom=from; nfrom.norm();
   Vec nto=to; nto.norm();
   double d=dot(nfrom, nto), c=cross(nfrom, nto), a=acos(d);
   if(ls(c,0.0f)) return 2.0f*pi-a; else return a;
}
```

6.6.5 2D point - line distance

```
inline double dist2line(const Point& p,const Point& o,const Vec& d){
   Vec u=p-o,nd=d;nd.norm();
   double proj=dot(u,nd);
   return sqrt(u.sqlen()-proj*proj);
}
```

6.6.6 2D point - circle tangent

POJ 1375

```
1
     inline int point_circle_tan(const Point& p,const Point& C,const
         double& r,Point* tps){
2
       Vec v=C-p; double cdist=v.len();
        if(ls(cdist,r)) return 0;
3
       else if(equ(cdist,r)){tps[0]=p;return 1;}
4
5
       double l=sqrt(cdist*cdist-r*r);v.norm();v=v*l;
6
       double a=asin(r/cdist);
7
       Vec u=vec_rotate(v,a);tps[0]=p+u;
8
       u=vec\_rotate(v,-a);tps[1]=p+u;
9
       return 2;
10
     };
```

6.6.7 2D circle - circle tangent

require circles no touch, no intersect POJ 2416

```
1
      struct Seg{
2
        Point p0, p1;
3
        Seg(const Point& m=Point(), const Point& n=Point()):p0(m),p1(n){
4
5
      };
6
7
      inline void tan_2circle(const Point& CO, const double& RO,
8
      const Point& C1, const double& R1,
9
      Seg* segs){
10
        Point c0=C0, c1=C1; double r0=R0, r1=R1;
11
        if (gr(R1,R0)) {swap(c0,c1);swap(r0,r1);}
12
        Vec v=c1-c0; double cdist=v.len();
13
        Point p=c0+v*(r0/(r1+r0));
14
15
        Point t[2];
16
        point_circle_tan(p,c0,r0,t);
17
        Vec u=p-t[0]; Point w;
18
       w=t[0]+u*((r1+r0)/r0);
19
        segs[0]=Seg(t[0],w);
20
        u=p-t[1];w=t[1]+u*((r1+r0)/r0);
21
        segs[1]=Seg(t[1],w);
22
23
        if (equ(r0,r1)){
24
          u=v;u.norm();u=u*r0;
25
          Vec offset=vec_rotate(u, pi*0.5);
26
          segs[2]=Seg(c0+offset,c1+offset);
27
          offset=vec_rotate(u,-pi*0.5);
28
          segs[3]=Seg(c0+offset,c1+offset);
```

```
29
        }
30
        else {
31
          p=c0+v*(r0/(r0-r1));
32
          Point k[2];
33
          point_circle_tan(p,c1,r1,t); point_circle_tan(p,c0,r0,k);
34
          segs[2]=Seg(t[0],k[0]);
35
          segs[3]=Seg(t[1],k[1]);
36
        }
37
      }
```

6.6.8 2D circle - polygon intersection area

POJ 3675 schindlerlee, thanks

```
1
      //CCW
 2
      inline double poly_area(const vector < Point > & p){
 3
        Point prev=p[0], cur;
 4
        double area=0;
 5
        for(int i=1;i<p.size();++i){
          cur=p[i];
 6
 7
          area+=0.5 * cross(prev,cur);
 8
          prev=cur;
 9
10
        return area;
11
      }
12
13
      double R;
14
      Point C;
15
      vector < Point > P;
16
      int main(){
        while (scanf("% If ",&R) == 1) {
17
18
        P. clear();
19
        int N;
20
        scanf("%d",&N);
21
        for (int i=0; i< N; ++ i) {
22
          double x,y;scanf("%|f%|f",&x,&y);
23
          P.push_back(Point(x,y));
24
25
        C=Point(0.0f,0.0f);
        P.push_back(P[0]);
26
27
        double area = 0.0 f;
28
        double t,k;
29
        for (int i = 0; i < P. size () -1; ++i) {
30
          Point s0=P[i], s1=P[i+1];
31
           if (line_circle_intersect(s0,s1-s0,C,R,&t,&k)){
32
             Point p0=s0+(s1-s0)*t;
33
             Point p1=s0+(s1-s0)*k;
34
             if(|seq(k,0.0f) || greq(t,1.0f)){
35
               area+=0.5 * R*R * angle_diff(s0,s1);
36
37
             else if(lseq(t,0.0f) && greq(k,1.0f)){
               area+=0.5 * cross(s0,s1);
38
39
             else if(lseq(t,0.0f) && greq(k,0.0f) && lseq(k,1.0f)){
40
41
               area +=0.5 * cross(s0, p1);
               area+=0.5 * R*R * angle_diff(p1,s1);
42
43
             }
```

```
44
            else if(greq(k,1.0f) && greq(t,0.0f) && lseq(t,1.0f)){
              area+=0.5 * R*R * angle_diff(s0,p0);
45
46
              area+=0.5 * cross(p0,s1);
47
            }
48
            else{
              area+=0.5 * R*R * angle_diff(s0,p0);
49
              area+=0.5 * cross(p0,p1);
50
              area+=0.5 * R*R * angle_diff(p1,s1);
51
52
53
54
          else area+=0.5f * R*R * angle_diff(s0,s1);
55
56
        printf("%.2f\n", fabs(area));
      }//while case
57
58
      return 0;
59
```

6.6.9 2D vector normal

return two normalized vectors perp to the v (CCW,CW)

```
inline pair < Vec, Vec > vec_normal(const Vec& v) {
    Vec u=v;u.norm();
    Vec CCW=Vec(-u.y,u.x), CW=Vec(u.y,-u.x);
    return make_pair(CCW,CW);
}
```

6.6.10 2D vector rotation

CCW for positive angle

```
inline Vec vec_rotate(const Vec& v,const double& a){
   double s=sin(a),c=cos(a);
   return Vec(c*v.x-s*v.y,s*v.x+c*v.y);
}
inline Vec vec_rotate_left(const Vec& v){return Vec(-v.y,v.x);}
inline Vec vec_rotate_right(const Vec& v){return Vec(v.y,-v.x);}
```

6.7 2D Intersection

6.7.1 ray - ray intersection

```
1
      inline bool ray intersect (const Point& p, const Vec& u,
2
      const Point& q, const Vec& v,
3
      double* t, double* k) {
4
        double a=u.x,b=-v.x,c=u.y,d=-v.y;
        double e=q.x-p.x, f=q.y-p.y;
5
        double det=a*d-b*c;
6
7
        if(equ0(det)) return false;
8
        else {
9
          t = (e^*d - b^*f) / det;
10
          *k=(a*f-e*c)/det;
11
          return true;
12
        }
13
      }
```

6.7.2 line - circle intersection

```
1
      inline bool line_circle_intersect(const Point& o, const Vec& d,
2
      const Point& C, const double& R,
3
      double* t,double* k){
        Vec delt=o-C;
4
5
        double a=d.sqlen(),b=2*dot(delt,d),c=delt.sqlen()-R*R;
        double x=b*b-4*a*c;
6
7
        if(ls(x,0.0f)) return false;
8
        x = sgrt(x); *t = (-b-x)/(2*a); *k = (-b+x)/(2*a);
9
        return true;
10
     }
```

6.8 Computational Geometry Topics

6.8.1 2D convex hull

O(NlogN)
monotone chain
vertex points only: modify the 'left' to strict, otherwise colinear points included.
CCW order
POJ 2187

```
const int MAXP=50000;
 1
 2
      Point P[MAXP];
 3
      int N;
 4
 5
      int H[MAXP], lower[MAXP];
 6
      bool vis[MAXP];
 7
      int top;
 8
      int monotone chain(){
9
        memset(vis,0,sizeof(vis));
        int utop=0,ltop=0;
10
        for (int i=N-1; i>=0; --i) {
11
12
           if (utop < 2) H[utop + +] = i;
13
           else{
             while(utop>1 && cross(P[H[utop-1]]-P[H[utop-2]],P[i]-P[H[utop
14
                 -1]]) <= 0) --utop;
15
            H[utop++]=i;
16
          }
17
        for(int i=0;i<utop;++i) vis[H[i]]=true;</pre>
18
19
        for (int i = 0; i < N; ++i) {
20
           if (ltop <2) lower[ltop++]=i;</pre>
21
           else{
22
             while(ltop>1 && cross(P[lower[ltop-1]]-P[lower[ltop-2]],P[i]-P[
                 lower[ltop - 1]]) <= 0)
23
             —Itop;
24
             lower[ltop++]=i;
25
26
27
        for(int i=0;i<ltop;++i) if(!vis[lower[i]]) H[utop++]=lower[i];</pre>
28
        return utop;
29
      }
```

6.8.2 3D convec hull

O(N2) incremental method POJ 3528 schindlerlee, thanks.

```
1
      //CCW
 2
      struct Face{
 3
        Point p0, p1, p2;
 4
        Face(const Point& a=Point(),const Point& b=Point(),const Point& c=
            Point()):
 5
        p0(a),p1(b),p2(c){
 6
 7
      };
 8
      inline Vec face normal(const Face& f){
9
        Vec u=f.p1-f.p0;
10
        Vec v=f.p2-f.p0;
11
        Vec n=cross(u,v);
12
        n.norm();
13
        return n;
14
15
      inline bool point_above_face(const Point& p,const Face& f){
16
        Vec u=p-f.p0;
17
        Vec n=face_normal(f);
18
        return gr(dot(u,n),0.0f);
19
      }
20
21
22
      int edge[MAXP][MAXP];
23
      int main(){
24
        int N;
25
        vector < Point > Points;
26
        vector < Face > CH;
27
        while (scanf("%d",&N) == 1){
28
        for (int i = 0; i < N; ++i) {
29
          double x,y,z;
          scanf("%|f%|f%|f",&x,&y,&z);
30
          Points.push_back(Point(x,y,z,i));
31
32
33
        //special case
34
35
        CH. push_back(Face(Points[0], Points[1], Points[2]));
36
        CH. push_back(Face(Points[2], Points[1], Points[0]));
37
38
        for(int i=3;i<Points.size();++i){</pre>
39
          Point cur=Points[i];
40
          for(int j=0;j<CH.size();++j){
41
            Face f=CH[i];
42
             if(point above face(cur, f)){
               edge[f.p0.id][f.p1.id]=1;
43
               edge[f.p1.id][f.p2.id]=1;
44
45
               edge[f.p2.id][f.p0.id]=1;
46
47
            else{
48
               edge [f.p0.id][f.p1.id]=-1;
49
               edge [f.p1.id][f.p2.id]=-1;
               edge[f.p2.id][f.p0.id]=-1;
50
            }
51
```

```
52
          }
53
          vector < Face > T;
54
          for(int j=0;j<CH.size();++j){
55
            Face f=CH[j];
56
            if (edge[f.p0.id][f.p1.id]==1){
57
               if(edge[f.p1.id][f.p0.id]==-1) T.push_back(Face(f.p0,f.p1,cur
58
               if(edge[f.p2.id][f.p1.id]==-1) T.push_back(Face(f.p1,f.p2,cur
59
               if(edge[f.p0.id][f.p2.id]==-1) T.push_back(Face(f.p2,f.p0,cur
                  ));
60
            }
61
            else{
62
              T.push_back(f);
63
64
          }
65
          swap(CH,T);
66
67
        double area=0.0f;
68
        for(int i=0;i<CH.size();++i){
69
          Face f=CH[i];
70
          Vec crs=cross(f.p1-f.p0,f.p2-f.p0);
          area+=0.5f * crs.len();
71
72
        printf("%.3f\n", area);
73
74
75
      return 0;
76
      }
```

6.8.3 Max distance point pair

O(NlogN) applied on the convex hull CCW ordered points rotating calipers no explicit angle calculation POJ 2187; TJU 2847

```
inline int Next(int x){return (x+1)%top;}
1
2
   int max_dist;
3
   void rc(){
4
      int s;
5
      int p=0,q=Next(p);
6
      while(area(P[H[p]],P[H[Next(p)]],P[H[Next(q)]]) >
7
      area(P[H[p]],P[H[Next(p)]],P[H[q]])){
8
        q=Next(q);
9
      }
10
      s=p;
11
12
        Vec d=P[H[p]] - P[H[q]]; int temp=d.sqlen();
13
        max_dist=max( max_dist , temp);
14
        int np=p,nq=q;
15
        if (area(P[H[p]], P[H[Next(p)]], P[H[Next(q)]]) >
16
        area(P[H[p]], P[H[Next(p)]], P[H[q]])) nq=Next(q);
17
        if (area(P[H[q]], P[H[Next(q)]], P[H[Next(p)]]) >
        area(P[H[q]],P[H[Next(q)]],P[H[p]])) np=Next(p);
18
        if(nq!=q) q=Next(q);
19
20
        else p=Next(p);
21
      } while(q!=Next(s));
```

22 | }

6.8.4 closed point pair among two polygons

```
O(NlogN)
rotating calipers
no explicit angle calculation
POJ 3608 utility functions
compare functor for monotone chain convec hull
```

```
struct comp{
bool operator()(const Point& a,const Point& b)const{
  if(equ(a.x,b.x)) return ls(a.y,b.y); else return ls(a.x,b.x);
}

};
```

utility functions to step to the next vertex. used in rotating calipers

```
1 inline int nextp(int i){return (i+1)%M;}
2 inline int nextq(int i){return (i+1)%N;}
```

check if a point can perp project on a segment

```
inline bool point_over_seg(const Point& p,const Point& a,const Point& b
    ){
    Vec u=b-a; double l=u.len();
    u.norm();
    double proj=dot(u,p-a);
    double t=proj/l;
    return ls(t,1.0f) && gr(t,0.0f);
}
```

check if two segments' perp projection overlap

```
inline bool seg_over_seg(const Point& a,const Point& b,
1
2
                           const Point& c, const Point& d){
3
     Vec u=b—a:
4
     Vec r0=vec_rotate_left(u),r1=vec_rotate_left(u);
5
     double t,k;
6
     true;
7
     if (ray\_intersect(b,r1,c,d-c,&t,&k)) if (ls(k,1) \&\& gr(k,0)) return
        true;
8
     Vec v=d-c:
9
     r0=vec_rotate_left(v);r1=vec_rotate_left(v);
10
     if (ray\_intersect(c, r0, a, b-a, &t, &k)) if (ls(k, 1) && gr(k, 0)) return
11
     if(ray\_intersect(d,r1,a,b-a,&t,&k)) if(ls(k,1) \& gr(k,0)) return
        true;
12
     return false;
```

13 | }

return two parallel segments' distance

rotating calipers

```
1
    void rc(){
2
      int p,q;
3
      int sp;
4
      int cnt=0;
5
      double f;
6
      f = 10001;
7
      for (int i=0; i < M; ++ i) if (P[i]. x < f) { f=P[i]. x; p=i; }
8
      f = -10001:
9
      for(int i=0; i< N; ++i) if(Q[i].x>f){f=Q[i].x;q=i;}
10
      while (gr(area(P[p], P[nextp(p)], Q[nextq(q)]), area(P[p], P[nextp(p)], Q[q
          ]))) q=nextq(q);
11
      sp=p;
12
      if(point_over_seg(Q[q],P[p],P[nextp(p)]))
13
14
        mindist=min(mindist, dist2line(Q[q],P[p],P[nextp(p)]-P[p]));
15
      else mindist=min(mindist,(P[p]-Q[q]).len());
16
      do{
17
        int np=p,nq=q;
18
        if (gr (area (P[p], P[nextp(p)], Q[nextq(q)]), area (P[p], P[nextp(p)], Q[q
            ]))) nq=nextq(q);
19
        if (gr (area (Q[q],Q[nextq(q)],P[nextp(p)]), area (Q[q],Q[nextq(q)],P[p
            ]))) np=nextp(p);
20
        if(nq==q && np!=p){
21
          if (point_over_seg(Q[q],P[p],P[nextp(p)]))
22
            mindist=min(mindist, dist2line(Q[q],P[p],P[nextp(p)]-P[p]));
23
          else mindist=min(mindist,(Q[q]-P[nextp(p)]).len());
24
          p=nextp(p);
25
26
        else if (np==p \&\& nq!=q) {
          if (point_over_seg(P[p],Q[q],Q[nextq(q)]))
27
28
            mindist=min(mindist, dist2line(P[p],Q[q],Q[nextq(q)]-Q[q]));
29
          else mindist=min(mindist,(P[p]-Q[nextq(q)]).len());
30
          q=nextq(q);
31
32
        else if (np==p \&\& nq==q) {
33
          if (seg_over_seg(P[p], P[nextp(p)], Q[q], Q[nextq(q)]))
34
            mindist=min(mindist, dist_2seg(P[p], P[nextp(p)], Q[q], Q[nextq(q)
                ]));
35
          else {
36
            // mindist=min(mindist,(P[nextp(p)]-Q[nextq(q)]).len());
37
            mindist=min(mindist,(P[nextp(p)]-Q[q]).len());
38
            // mindist=min(mindist,(Q[nextq(q)]-P[p]).len());
39
          }
40
          p=nextp(p);
41
42
        if (p==sp) ++cnt;
43
      } while (!( cnt>1&&p==nextp(sp)));
44
   }
```

6.8.5 closed point pair O(NlogN) sweep line POJ 3714; ZOJ 2107 compx, compy, compare functors used in sorted point sequence and BST respectively

```
const int MAXN=200002;
2
3
   Point P[MAXN];
4
   int N;
5
   struct compx{
6
     bool operator()(const Point& a, const Point& b)const{
7
        if(a.x==b.x) return ls(a.y,b.y);
8
        else return ls(a.x,b.x);
9
     }
10
   };
11
   struct compy{
     bool operator()(const Point& a, const Point& b)const{
12
13
        if(a.y==b.y) return ls(a.x,b.x);
        else return ls(a.y,b.y);
14
15
16
   };
```

sweep line algorithm

```
double mindist;
2
3
   Point eventp;
4
   typedef set < Point, compy> Box;
5
   typedef Box::iterator BoxIter;
   void sweepline(){
7
      sort(P,P+2*N,compx());
8
      mindist=INF;
     int I=0;
9
10
     Box box;
11
      for (int i=0; i<2*N;++i) {
12
        eventp=P[i];
13
        double x=eventp.x-mindist, negy=eventp.y-mindist, posy=eventp.y+
           mindist;
14
15
        while(|<i && |s(P[|].x,x)){
16
          box.erase(P[I]);
17
          ++|:
18
        }
19
20
        BoxIter a=box.lower_bound(Point(x,negy));
21
        BoxIter b=box.lower_bound(Point(x,posy));
        for(BoxIter iter=a;iter!=b;iter++){
22
23
          Vec d=*iter - eventp;
24
          if(iter ->color != eventp.color) mindist=min(mindist,d.len());
25
26
        box.insert(eventp);
27
     }
28
   }
```

```
6.8.6 closed circle pair

O(NlogN)
sweep line
HDU 3124
data structure and compare functors
```

```
struct Circle {
2
      Point c; double r;
3
      Circle(const Point& a=Point(), const double& b=0):c(a),r(b){
4
5
   };
6
   const int MAXN=50000;
7
   Circle C[MAXN];
8
   int N;
9
   Point P[MAXN];
10
11
   struct compx{
12
     bool operator()(const Point& a, const Point& b)const{
13
        if(equ(a.x,b.x)) return ls(a.y,b.y);
14
        else return ls(a.x,b.x);
15
   };
16
17
   struct compy{
18
      bool operator()(const Point& a,const Point& b)const{
19
        if(equ(a.y,b.y)) return ls(a.x,b.x);
20
        else return ls(a.y,b.y);
21
     }
22
   };
```

sweep line algorithm, use circle's top and bottom vertex as event point

```
double mindist;
2
3
   Point eventp:
   typedef set < Point, compy> Box;
   typedef Box::iterator BoxIter;
   void sweepline(){
6
7
      sort(P,P+N,compx());
8
      mindist=INF;
9
      int I=0;
10
      Box box;
      for (int i = 0; i < N; ++i) {
11
12
        eventp=P[i];
13
        Circle cir=C[eventp.c];
        double x=eventp.x-mindist,negy=cir.c.y-cir.r-mindist,posy=cir.c.y+
14
            cir.r+mindist:
15
16
        while (1 < i \& ls(C[P[1].c].c.x+C[P[1].c].r,x)){
17
          Circle tmp=C[P[1].c];
18
          box.erase(Point(tmp.c.x,tmp.c.y-tmp.r));
19
          box.erase(Point(tmp.c.x,tmp.c.y+tmp.r));
20
          ++|;
21
        }
22
23
        if (!box.empty()){
24
          BoxIter begin=box.lower_bound(Point(NEGINF, negy));
25
          BoxIter end=box.lower_bound(Point(NEGINF, posy));
26
          for(BoxIter iter=begin;iter!=end;++iter){
            Circle tmp=C[iter ->c];
27
```

```
double d=(tmp.c-cir.c).len() - tmp.r - cir.r;
mindist=min(mindist,d);

}

box.insert(Point(cir.c.x,cir.c.y+cir.r,eventp.c));
box.insert(Point(cir.c.x,cir.c.y-cir.r,eventp.c));

}

}
```

main function, save circle's top and bottom vertex

```
int main(){
2
      int T; scanf("%d",&T);
      while (T-->0) {
3
        scanf("%d",&N);
4
5
        for (int i = 0; i < N; ++i) {
          int x,y,r;
6
7
          scanf("%d%d%d",&x,&y,&r);
8
          C[i].c.x=x;C[i].c.y=y;C[i].r=r;
9
          P[i]=Point(C[i].c.x-C[i].r,C[i].c.y,i);
10
11
        sweepline();
        printf("%.6f\n", mindist);
12
13
14
      return 0;
15
```

6.8.7 circle hierarchy

O(NlogN)

sweep line

event point is the most left and right vertex of a circle implicit store the interval cuted by the current sweep line.

POJ 2932 data structures and compare functors

```
const int MAXC=40000;
2
   const int MAXN=2*MAXC;
3
   struct Circle {
4
      Point c; double r;
      Circle(const Point& a=Point(), const double& b=0):c(a),r(b){}
5
6
   };
7
   //a extra INF radius circle
   Circle circle [MAXC+1];
   Point P[MAXN];
10
   int N:
   Point eventp;
11
12
13
   struct compx{
14
     bool operator()(const Point& a, const Point& b)const{
15
        if(equ(a.x,b.x)) return ls(a.y,b.y); else return ls(a.x,b.x);
16
17
   };
18
19
   struct Interval {
20
     int p;
     int i,j;
21
      bool lowi, lowj;
22
```

```
23
      Interval(int par=0,int a=0,bool a0=false,int b=0,bool b0=false):
24
        p(par), i(a), j(b), lowi(a0), lowj(b0) {}
25
   };
   inline pair < double, double > get_itv(const Circle& c, const double& x){
26
      double d=fabs(c.c.x-x);
27
     double del=c.r*c.r - d*d;
28
29
      if(lseq(del,0.0f)) del=0.0f;
30
      del=sqrt(del);
31
      return make_pair(c.c.y-del,c.c.y+del);
32
33
   struct compcircle{
34
      bool operator()(const Interval& a,const Interval& b)const{
35
        pair < double , double > itv0 = get_itv(circle[a.i], eventp.x);
36
        pair < double > itv1 = get_itv(circle[a.j], eventp.x);
37
        double x0,x1;
38
        if (a.lowi) x0=itv0.first; else x0=itv0.second;
39
        if (a.lowj) x1=itv1.first; else x1=itv1.second;
40
41
        itv0=get_itv(circle[b.i], eventp.x);
42
        itv1=get_itv(circle[b.j], eventp.x);
43
        double y0, y1;
44
        if (b.lowi) y0=itv0.first; else y0=itv0.second;
45
        if(b.lowj) y1=itv1.first;else y1=itv1.second;
46
47
        if(equ(x0,y0)) return ls(x1,y1);
48
        else return ls(x0,y0);
49
50
   };
51
52 //-
                -union set-
   int parent[MAXC+1];
54
   inline void make_set(){for(int i=0;i<=N;++i) parent[i]=i;}</pre>
   //attach y to x
   inline void link(int x,int y){parent[y]=x;}
```

```
sweep line
```

left and right point as event point

store following in the BST:(Interval structure)

- -- parent
- -- which circle contribute the two end points
- -- if these end points is bottom vertex of the interval

```
typedef set < Interval . compcircle > Ityl:
2
   typedef Itvl::iterator Iter;
 3
    void debug(Itvl itvl){
      for(Iter k=itvl.begin(); k!=itvl.end();++k){
 4
 5
        printf("parent: %d lower: %d %d upper: %d %d\n",k->p,k->i,k->lowi,
            k \rightarrow j, k \rightarrow lowj);
 6
 7
    }
 8
   void sweepline(){
9
      make_set();
      sort(P,P+2*N,compx());
10
11
      Itvl itvl;
12
      itvl.insert(Interval(0,0,true,0,false));
13
14
      for(int i=0; i<2*N;++i){
15
        eventp=P[i];
16
```

```
17
        if(eventp.in){
18
          Iter lower=itvl.lower bound(Interval(0, eventp.i, true, eventp.i,
              false));
19
          --lower:
          link(lower->p, eventp.i);
20
          Interval tmp=*lower;
21
22
          itvl.erase(lower);
          itvl.insert(Interval(tmp.p,tmp.i,tmp.lowi,eventp.i,true));
23
24
          itvl.insert(Interval(eventp.i, eventp.i, true, eventp.i, false));
25
          itvl.insert(Interval(tmp.p,eventp.i,false,tmp.j,tmp.lowj));
26
27
        else{
28
          Iter d=itvl.find(Interval(0, eventp.i, true, eventp.i, false));
29
          Iter l=d;--l;
30
          Iter u=d;++u;
31
          Interval low=*I,up=*u;
          itvl.erase(d);itvl.erase(l);itvl.erase(u);
32
33
          itvl.insert(Interval(low.p,low.i,low.lowi,up.j,up.lowj));
34
        }
35
     }
36
   }
```

as for POJ 2932 don't ask for the total hierarchy, so this is a special sweep line algorithm to speed up.

```
void sweepline(){
1
2
      make set():
3
      sort(P, P+2*N, compx());
4
      Itvl itvl;
5
      itvl.insert(Interval(0,0,true,0,false));
6
7
      for(int i=0; i<2*N;++i){
8
        eventp=P[i];
9
10
        if (eventp.in) {
11
          Iter lower=itvl.lower_bound(Interval(0, eventp.i, true, eventp.i,
              false));
12
          —lower;
13
          link(lower->p, eventp.i);
14
          if(lower->p == 0){
15
            Interval tmp=*lower;
16
            itvl.erase(lower);
17
            itvl.insert(Interval(tmp.p,tmp.i,tmp.lowi,eventp.i,true));
18
            itvl.insert(Interval(eventp.i, eventp.i, true, eventp.i, false));
19
            ityl.insert(Interval(tmp.p,eventp.i,false,tmp.j,tmp.lowj));
20
          }
21
        }
22
        else{
23
          Iter d=itvl.find(Interval(0, eventp.i, true, eventp.i, false));
24
          if(d==itvl.end()) continue;
25
          Iter l=d;--l;
26
          Iter u=d;++u;
          Interval low=*1,up=*u;
27
28
          itvl.erase(d);itvl.erase(l);itvl.erase(u);
29
          itvl.insert(Interval(low.p,low.i,low.lowi,up.j,up.lowj));
30
        }
31
      }
32
   }
```

- 7 Math
- 7.1 Catalan Number