ACM/ICPC TEMPLATE

NKU -> HOT

October 20, 2012

Contents

1 Introduction

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

2 Utility

2.1 Java Template

```
import java.util.*;
import java.io.*;

class Main {

    void run() {
        //Scanner in = new Scanner(System.in);
        MyReader in = new MyReader();
        String str;
    }

    public static void main(String args[]) {
        new Main().run();
    }

    void debug(Object...x) {
        System.out.println(Arrays.deepToString(x));
    }
}

class MyReader {
    BufferedReader br = new BufferedReader (
            new InputStreamReader (System.in));
    StringTokenizer in;
    String next() {
        try {
            while (in == null || !in.hasMoreTokens()) {
                // Read a new line and split it into tokens
                in = new StringTokenizer(br.readLine());
            }
            // return next token
            return in.nextToken();
        } catch (Exception e) {
            // EOF
            return null;
        }
    }
    // Transform the tokens into other types
    int nextInt() {
        return Integer.parseInt(next());
    }
}
```

2.2 Java Multithread

BECAREFUL: CALL START FOR EACH THREAD!

```
class Test extends Thread {
    public static int ans;
    public static int end;
    public void run() {
        int now = 0;
        for (int i = 0; i < 400000000; i++) {
            now = (now + i) % 9999997;
        }
        System.out.println(now);
        new SubTask(0,0,100000000).start();
        new SubTask(1,100000000,200000000).start();
        new SubTask(2,200000000,300000000).start();
        new SubTask(3,300000000,400000000).start();
        for (;;) {
            try {
                sleep(200);
            } catch (Exception e) {}
            if (end == 4) break;
        }
        System.out.println(ans);
    }
    public static void main(String[] args) {
        new Test().start();
    }
}

class SubTask extends Thread {
    private int pos;
    private int left;
    private int right;
    final static int mod = 9999997;

    // init the input data
    SubTask(int pos,int left,int right) {
        this.pos = pos;
        this.left = left;
        this.right = right;
    }

    public void run() {
        // solve the problem
        int ans = 0;
        for (int i = left; i < right; i++) {
            ans = (ans + i) % mod;
        }
        // write the answer back
        synchronized (this) {
            Test.ans += ans;
            Test.ans %= mod;
            Test.end ++;
        }
    }
}
```

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>
using namespace std;

const int MAXN = 100;
const int EXP = 10;
const int INF = 1000000000;

int nn;
int map[MAXN+EXP][MAXN+EXP];

int sum;
bool inSet[MAXN+EXP];
int dist[MAXN+EXP];

void Prim(){
  sum = 0;
  for(int i = 1; i <= nn; i++) inSet[i] = 0, dist[i] = INF;
  dist[1] = 0;
  for(int i = 0; i < nn; i++){
    int min = INF, idx = 0;
    for(int j = 1; j <= nn; j++)
      if(!inSet[j] && dist[j] < min)
        min = dist[j], idx = j;
    inSet[idx] = 1;
    sum += min;
    for(int j = 1; j <= nn; j++)
      if(!inSet[j] && dist[j] > map[idx][j])
        dist[j] = map[idx][j];
  }
}

int main(){
  while(scanf("%d\n",&nn) == 1 && nn){
    for(int i = 1; i <= nn; i++)
      for(int j = 1; j <= nn; j++)
        scanf("%d",&map[i][j]);
    Prim();
    printf("%d\n",sum);
  }
  return 0;
}
```

3.2 Prim- O(MlogN)

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

const int MAXN = 100;
const int MAXM = 10000;
const int EXP = 10;
const int INF = 1000000000;

int nn,mm;

int edges;
struct EDGE{
  int n;
  int v;
  EDGE* nxt;
}pool[MAXM*2+EXP];
EDGE lnk[MAXN+EXP];

void addEdge(int _f, int _t, int _v){
  pool[edges].n = _t;
  pool[edges].v = _v;
  pool[edges].nxt = lnk[_f].nxt;
  lnk[_f].nxt = &pool[edges];
  edges++;
}

struct NODE{
  int n;
  int dst;
  NODE(int _n = 0, int _dst = 0){
    n = _n;
    dst = _dst;
  }
};
bool operator <(NODE aa, NODE bb){
  return aa.dst > bb.dst;
}

int sum;
bool inSet[MAXN+EXP];
int dist[MAXN+EXP];

void Prim_Prio(){
  sum = 0;
  for(int i = 1; i <= nn; i++) inSet[i] = 0, dist[i] = INF;
  dist[1] = 0;
  priority_queue <NODE> Q; Q.push(NODE(1,0));
  while(Q.size()){
    NODE now = Q.top(); Q.pop();
    if(inSet[now.n]) continue;
    inSet[now.n] = 1;
    sum += now.dst;
    for(EDGE* tmp = lnk[now.n].nxt; tmp; tmp = tmp->nxt){
      if(!inSet[tmp->n] && tmp->v < dist[tmp->n]){
        dist[tmp->n] = tmp->v;
        Q.push(NODE(tmp->n,tmp->v));
      }
    }

  }
}

int main(){
  int cas; scanf("%d",&cas);
  while(cas--){
    scanf("%d%d", &nn, &mm);
    edges = 0;
    for(int i = 1; i <= nn; i++) lnk[i].nxt = 0;
    for(int i = 1; i <= mm; i++){
      int aa,bb,vv; scanf("%d%d%d", &aa, &bb, &vv);
      addEdge(aa, bb, vv);
    }
    Prim_Prio();
    printf("%d\n",sum);
  }
  return 0;
}
```

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;

int nn,mm;

struct EDGE{
  int f;
  int t;
  int v;
}pool[MAXM+EXP];

bool cmp(EDGE a, EDGE b){
  return a.v < b.v;
}

int fa[MAXN+EXP];
int find(int x){
  int r = x;
  while(r != fa[r]) r = fa[r];
  while(x != r){
    int tmp = fa[x];
    fa[x] = r;
    x = tmp;
  }
  return r;
}

void uni(int aa, int bb){
  int xx = find(aa);
  int yy = find(bb);
  if(xx != yy) fa[yy] = xx;
}

int sum;

void Kruskal(){
  sum = 0;
  sort(pool, pool+mm, cmp);
  for(int i = 1; i <= nn; i++) fa[i] = i;
  for(int i = 0; i < mm; i++){
    int aa = find(pool[i].f);
    int bb = find(pool[i].t);
    if(aa == bb) continue;
    sum += pool[i].v;
    uni(aa, bb);
  }
}


int main(){
  int cas;    scanf("%d", &cas);
  while(cas--){
    scanf("%d%d", &nn, &mm);
    for(int i = 0; i < mm; i++)
      scanf("%d%d%d", &pool[i].f, &pool[i].t, &pool[i].v);
    Kruskal();
    printf("%d\n",sum);
  }
  return 0;
}
```

3.4 Dijkstra - O(N^2)

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

const int MAXN = 1000;
const int EXP = 10;
const int INF = 1000000000;

int nn;
int mm;

int map[MAXN][MAXN];

int dist[MAXN+EXP];
bool inSet[MAXN+EXP];

void init(){
  for(int i = 0; i <= nn; i++)
    for(int j = 0; j <= nn; j++)
      map[i][j] = INF;
}

void Dijk(int s){
  for(int i = 1; i <= nn; i++){
    dist[i] = INF;
    inSet[i] = 0;
  }
  dist[s] = 0;
  for(int i = 1; i <= nn; i++){
    int min = INF, idx = 0;
    for(int j = 1; j <= nn; j++){
      if(!inSet[j] && dist[j] < min){
        min = dist[j];
        idx = j;
      }
    }
    inSet[idx] = 1;
    for(int j = 1; j <= nn; j++){
      if(!inSet[j] && dist[idx] + map[idx][j] < dist[j])
        dist[j] = dist[idx] + map[idx][j];
    }
  }
}

int main(){
  int cas; scanf("%d", &cas);
  while(cas--){
    scanf("%d%d", &nn, &mm);
    init();
    for(int i = 1; i <= mm; i++){
      int aa,bb,dd; scanf("%d%d%d", &aa, &bb, &dd);
      if(map[aa][bb] > dd){
        map[aa][bb] = map[bb][aa] = dd;
      }
    }
    Dijk(1);
    cout<<dist[nn]<<endl;
  }
  return 0;
}
```

3.5 Dijkstra - O(MlogN)

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

const int MAXN = 50000;
const int MAXM = 50000;
const int EXP = 10;
const int INF = 1000000000;

int edges;
struct EDGE{
  int n;
  int d;
  EDGE *nxt;
}pool[MAXM*2+EXP];
EDGE lnk[MAXN+EXP];

void addEdge (int _f, int _t, int _d){
  pool[edges].n = _t;
  pool[edges].d= _d;
  pool[edges].nxt = lnk[_f].nxt;
  lnk[_f].nxt = &pool[edges];
  edges++;
}

int nn;
int mm;

int dist[MAXN+EXP];
bool inSet[MAXN+EXP];

struct NODE{
  int n;
  int dst;
  NODE(int _n = 0, int _dst = 0){
    n = _n;
    dst = _dst;
  }
};

bool operator <(NODE aa, NODE bb){
  return aa.dst > bb.dst;
}

void Dijk_Prio(int s){
  for(int i = 1; i <= nn; i++){
    dist[i] = INF;
    inSet[i] = 0;
  }
  priority_queue <NODE> Q;
  dist[s] = 0;
  Q.push(NODE(s,dist[s]));
  while(Q.size()){
    NODE now = Q.top(); Q.pop();
    if(inSet[now.n] == 1) continue;
    inSet[now.n] = 1;
    for(EDGE * tmp = lnk[now.n].nxt; tmp; tmp = tmp->nxt){
      if(!inSet[tmp->n] && dist[now.n] + tmp->d < dist[tmp->n]){
        dist[tmp->n] = dist[now.n] + tmp->d;
        Q.push(NODE(tmp->n, dist[tmp->n]));
      }
    }
  }
}

int main(){
  int cas; scanf("%d", &cas);
  while(cas--){
    edges = 0;
    scanf("%d%d", &nn, &mm);
    for(int i = 1; i <= nn; i++) lnk[i].nxt = 0;
    for(int i = 1; i <= mm; i++){
      int aa,bb,dd; scanf("%d%d%d", &aa, &bb, &dd);
      addEdge(aa, bb, dd);
      addEdge(bb, aa, dd);
    }
    Dijk_Prio(1);
    //cout<<dist[?]
  }
  return 0;
}
```

3.6 Dijkstra with heap

```
#include <cstdio>
#include <cstring>

using namespace std;

const int maxN=1010;
const int inf=2000000000;

class DJ_heap {
public:
  int data[maxN];
  int index[maxN];
  int pos[maxN];
  int tot;
  void init (int n,int st) {
    for (int i = 2; i <= n; i++) {
      data[i] = inf;
      int now = (i == st ? 1 : i);
      index[i] = now;
      pos[now] = i;
    }
    data[1] = 0;
    index[1] = st;
    pos[st] = 1;
    tot = n;
  }
  void fix_down(int x) {
    for (int son = x + x; son <= tot; x = son, son = x + x) {
      if (son < tot && data[son+1] < data[son])
        son++;
      if (data[x] > data[son]) {
        int tmp=data[x]; data[x]=data[son]; data[son]=tmp;
        tmp=index[x]; index[x]=index[son]; index[son]=tmp;
        pos[index[x]]=x;
        pos[index[son]]=son;
      }
    }
  }
  void fix_up(int x) {
    for (int fa = x>>1; x > 1; x = fa, fa = x>>1) {
      if (data[fa] > data[x]) {
        int tmp=data[fa]; data[fa]=data[x]; data[x]=tmp;
        tmp=index[fa]; index[fa]=index[x]; index[x]=tmp;
        pos[index[x]]=x;
        pos[index[fa]]=fa;
      }
    }
  }
  void change(int x,int newdata) {
    data[pos[x]]=newdata;
    fix_up(pos[x]);
  }
  void pop(int &x,int &dist) {
    x=index[1];
    dist=data[1];
    index[1]=index[tot];
    data[1]=data[tot];
    pos[x]=0;
    pos[index[tot--]]=1;
    fix_down(1);
  }
  bool empty() {
    return tot==0;
  }
};

int a[1010][2000];
int b[1010][2000];
int dist[1010];
bool visit[1010];

DJ_heap q;

```

```
int main() {
  int n,m;
  scanf("%d%d",&m,&n);
  while (m--) {
    int f,t,cost;
    scanf("%d%d%d",&f,&t,&cost);
    a[f][++a[f][0]]=t;
    b[f][++b[f][0]]=cost;
    a[t][++a[t][0]]=f;
    b[t][++b[t][0]]=cost;
  }
  memset(dist,64,sizeof(dist));
  q.init(n,n);
  dist[n]=0;
  while (!q.empty()&&!visit[1]) {
    int v,d;
    q.pop(v,d);
    for (int i=1;i<=a[v][0];i++)
      if (!visit[a[v][i]] && dist[a[v][i]] > dist[v]+b[v][i]) {
        dist[a[v][i]] = dist[v] + b[v][i];
        q.change(a[v][i],dist[a[v][i]]);
      }
    visit[v]=1;
  }
  printf("%d\n",dist[1]);
  return 0;
}
```

3.7 Bellman-Ford

```
#include <iostream>
#include <cstdio>

using namespace std;

const int MAXN = 1000;
const int MAXM = 2000;
const int EXP = 10;
const int INF = 1000000000;

int mm,nn;

int vf[MAXM+EXP],vt[MAXM+EXP],vc[MAXM+EXP]; 记录边     //

int dist[MAXN+EXP];

void init(){
  scanf("%d%d",&nn,&mm);
  for(int i = 0; i < mm; i++){
    scanf("%d%d%d",vf+i,vt+i,vc+i);
  }
}

void Bellman_Ford(int s){
  for(int i = 1; i <= nn; i++)    dist[i] = INF;
  dist[s]=0;
  for(int i = 0; i < nn-1; i++){
    for(int i = 0; i < mm; i++){
      if(dist[vf[i]] + vc[i] < dist[vt[i]]){
        dist[vt[i]] = dist[vf[i]] + vc[i];
      }
      if(dist[vt[i]] + vc[i] < dist[vf[i]]){
        dist[vf[i]] = dist[vt[i]] + vc[i];
      }
    }
  }
}

int main(){
  init();
  Bellman_Ford(1);
  printf("%d\n",dist[nn]);
  return 0;
}
```

3.8 Shortest Path Faster Algorithm

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

const int MAXN = 50000;
const int MAXM = 50000;
const int EXP = 10;
const int INF = 1000000000;

int edges;
struct EDGE{
  int n;
  int d;
  EDGE *nxt;
}pool[MAXM*2+EXP];
EDGE lnk[MAXN+EXP];

void addEdge (int _f, int _t, int _d){
  pool[edges].n = _t;
  pool[edges].d= _d;
  pool[edges].nxt = lnk[_f].nxt;
  lnk[_f].nxt = &pool[edges];
  edges++;
}

int nn;
int mm;

bool inQ[MAXN+EXP];
int dist[MAXN+EXP];

void spfa(int s){
  for(int i = 0; i <= nn; i++){
    inQ[i] = 0;
    dist[i] = INF;
  }
  queue<int> Q; Q.push(s);
  inQ[s] = 1; dist[s] = 0;
  while(Q.size()){
    int now = Q.front(); Q.pop();
    inQ[now] = 0;
    for(EDGE* tmp = lnk[now].nxt; tmp; tmp = tmp->nxt){
      if(dist[now] + tmp->d < dist[tmp->n]){
        dist[tmp->n] = dist[now] + tmp->d;
        if(!inQ[tmp->n]) {
          Q.push(tmp->n);
          inQ[tmp->n] = 1;
        }
      }
    }
  }
}

int main(){
  int cas; scanf("%d", &cas);
  while(cas--){
    edges = 0;
    scanf("%d%d", &nn, &mm);
    for(int i = 1; i <= nn; i++) lnk[i].nxt = 0;
    for(int i = 1; i <= mm; i++){
      int aa,bb,dd; scanf("%d%d%d", &aa, &bb, &dd);
      addEdge(aa, bb, dd);
      addEdge(bb, aa, dd);
    }
    spfa(1);
    //cout<<dist[?]
  }
  return 0;
}
```

3.9 Network Flow - ISAP[NON-ORIGINAL]

```
#include <cstring>
#include <cstdio>
#include <queue>
#include <algorithm>
#include <vector>

using namespace std;

const int MAXN = 210;
const int MAXM=500010;
const int inf = 2E9;

typedef struct {int v,next,val;} edge;
struct SAP {
  edge e[MAXM];
  int p[MAXN],eid;
  inline void clear(){memset(p,-1,sizeof(p));eid=0;}
  inline void insert1(int from,int to,int val) {
    e[eid].v=to;
    e[eid].val=val;
    e[eid].next=p[from];
    p[from]=eid++;
    swap(from,to);
    e[eid].v=to;
    e[eid].val=0;
    e[eid].next=p[from];
    p[from]=eid++;
  }
  inline void insert2(int from,int to,int val) {
    e[eid].v=to;
    e[eid].val=val;
    e[eid].next=p[from];
    p[from]=eid++;
    swap(from,to);
    e[eid].v=to;
    e[eid].val=val;
    e[eid].next=p[from];
    p[from]=eid++;
  }
  int n;
  int h[MAXN];
  int gap[MAXN];
  int source,sink;
  inline int dfs(int pos,int cost) {
    if (pos==sink) {
      return cost;
    }
    int j,minh=n-1,lv=cost,d;
    for (j=p[pos];j!=-1;j=e[j].next) {
      int v=e[j].v,val=e[j].val;
      if(val>0) {
        if (h[v]+1==h[pos]) {
          if (lv<e[j].val) d=lv;
          else d=e[j].val;
          d=dfs(v,d);
          e[j].val-=d;
          e[j^1].val+=d;
          lv-=d;
          if (h[source]>=n) return cost-lv;
          if (lv==0) break;
        }
        if (h[v]<minh)  minh=h[v];
      }
    }
    if (lv==cost) {
      --gap[h[pos]];
      if (gap[h[pos]]==0) h[source]=n;
      h[pos]=minh+1;
      ++gap[h[pos]];
    }
    return cost-lv;
  }
  int run() {
    int ret=0;
    memset(gap,0,sizeof(gap));
    memset(h,0,sizeof(h));
    gap[source]=n;
    while (h[source]<n) ret+=dfs(source,inf);
```

```
    return ret;
  }
} solver;

int main() {
  int N,M;
  while (scanf("%d%d",&M,&N)!=EOF) {
    solver.source = 1;
    solver.sink = N;
    solver.n = N;
    solver.clear();
    while (M--) {
      int f,t,w;
      scanf("%d%d%d",&f,&t,&w);
      solver.insert1(f,t,w);
    }
    printf("%d\n",solver.run());
  }
  return 0;
}
```

3.10 Bipartite Graph Matching

```
#include <cstdio>
#include <cstring>

bool adj[555][555];
bool visit[555];
int match[555];
int n;

bool dfs(int now) {
    for (int i = 1; i <= n; i++) {
        if (visit[i] == false && adj[now][i]) {
            visit[i] = true;
            int tt = match[i];
            match[i] = now;
            if (tt == -1 || dfs(tt)) return true;
            match[i] = tt;
        }
    }
    return false;
}

int main() {
    int m;
    scanf("%d%d",&n,&m);
    for (int i = 0; i < m; i++) {
        int f,t; scanf("%d%d",&f,&t);
        adj[f][t] = true;
    }
    int ans = 0;
    memset(match,0xff,sizeof(match));
    for (int i = 1; i <= n; i++) {
        memset(visit,0,sizeof(visit));
        if (dfs(i)) ans++;
    }
    printf("%d\n",ans);
    return 0;
}
```

3.11 Minimun Cost Flow [TO BE TESTED!]

```
#include <iostream>
#include <queue>
#include <cstring>

using namespace std;

int n,m,ans,t,f;
int maxf[210][210],flow[210][210],cost[210][210];
int fa[210],dist[210];
bool inque[210];

inline int abs(int a) {return a > 0 ? a : -a ;}
void init() {
  int a[210][2]={0},b[210][2]={0},s=0,sa=0,sb=0;
  memset(maxf,0,sizeof(maxf));
  memset(flow,0,sizeof(flow));
  memset(cost,0,sizeof(cost));
  for (int i=1;i<=n;i++){
    for (int j=1;j<=m;j++) {
      char tt;
      cin>>tt;
      if (tt=='H') {
        a[++sa][0]=i;
        a[sa][1]=j;
      }
      if (tt=='m') {
        b[++sb][0]=i;
        b[sb][1]=j;
      }
    }
  }
  s=sa;
  for (int i = 1;i <= s; i++) {
    for (int j = 1;j <= s; j++) {
      cost[i][s+j] = abs(a[i][0]-b[j][0])+abs(a[i][1]-b[j][1]);
      cost[s+j][i] = cost[i][s+j];
      maxf[i][s+j] = 1;
    }
  }
  for (int i = 1; i <= s; i++)
    maxf[0][i]=maxf[s+i][s+s+1]=1;
  n = t = s + s + 1;
  f = 0;
  ans = 0;
}

inline int value(int i,int j) {
  return flow[j][i] > 0 ? -cost[i][j] : cost[i][j];
}

bool spfamark() {
  memset(fa,0,sizeof(fa));
  memset(inque,0,sizeof(inque));
  memset(dist, 0x3f, sizeof(dist));
  queue<int> q;
  q.push(f); inque[f] = true; dist[f]=0;
  while (!q.empty()) {
    int now = q.front(); q.pop(); inque[now] = false;
    for (int i = 0; i <= n;i++)
      if ((maxf[now][i] - flow[now][i] > 0)
          && dist[now] + value(now,i) < dist[i]){
        dist[i] = dist[now] + value(now,i);
        fa[i] = now;
        if (!inque[i]) {
          inque[i]=1;
          q.push(i);
        }
      }
  }
  return dist[t] != 0x3f3f3f3f;
}


int main() {
  while (cin >> n >> m && n && m) {
    init();
    while (spfamark()) {
      for(int i = t; i != f; i = fa[i]) {
        ans+=value(fa[i],i);
        flow[fa[i]][i]++;
        flow[i][fa[i]]--;
      }
    }
    cout << ans << endl;
  }
  return 0;
}
```

3.12 Kuhn-Munkras [NON-ORIGINAL]

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

```
class KM_class {
private:
  int match[maxm];
  int lx[maxn];
  int ly[maxm];
  bool vis_x[maxn];
  bool vis_y[maxm];
  int slack;

public:
  bool DFS(int u) {
    vis_x[u] = true;
    int tmp;
    for(int v = 1; v <= M; v++) {
      tmp = lx[u] + ly[v] - W[u][v];
      if(tmp == 0) {
        if(!vis_y[v]) {
          vis_y[v] = true;
          if(match[v] == 0 || DFS(match[v]) ) {
            match[v] = u;
            return true;
          }
        }
      } else {
        slack = min(slack,tmp);
      }
    }
    return false;
  }

  int KM() {
    memset(match,0,sizeof(match));
    memset(ly,0,sizeof(ly));
    for(int u = 1; u <= N; u++) {
      lx[u] = W[u][1];
      for(int v = 2; v <= M; v++) {
        lx[u] = max(lx[u],W[u][v]);
      }
    }

    for(int u = 1; u <= N; u++) {
      while(1) {
        slack = INT_MAX;
        memset(vis_x,0,sizeof(vis_x));
        memset(vis_y,0,sizeof(vis_y));
        if(DFS(u)) break;
        for(int i = 1; i <= N; i++)
          if(vis_x[i])
            lx[i] -= slack;
        for(int i = 1; i <= M; i++)
          if(vis_y[i])
            ly[i] += slack;
      }
    }
    int sum = 0;
    for(int v = 1; v <= M; v++) sum += W[match[v]][v];
    return -sum;
  }
} km;
```

3.13 Cut Vetrix and Edge

```
#include <cstdio>
#include <cstring>
#include <algorithm>

using namespace std;

bool a[110][110];
int visit[110];
int deep[110];
int back[110];
bool cut[110];
int n,ans;
```

```


void dfs(int k,int fa,int d) {
    visit[k]=1;
    back[k]=deep[k]=d;
    int tot=0;
    for (int i=1;i<=n;i++) {
        if (a[k][i] && i!=fa && visit[i]==1)
            back[k]=min(back[k],deep[i]);
        if (a[k][i] && visit[i]==0) {
            dfs(i,k,d+1);
            tot++;
            back[k]=min(back[k],back[i]);
            if ((k==1 &&tot>1) || (k!=1 && back[i]>=deep[k]))
                if (!cut[k]) {
                    cut[k]=1;
                    ans++;
                }
          //if back[i]>deep[k] k,i is bridge;
        }
    }
    visit[k]=2;
}

int main() {
    while (1) {
        scanf("%d",&n);
        if (n==0)
            break;
        memset(a,0,sizeof(a));
        memset(back,0,sizeof(back));
        memset(cut,0,sizeof(cut));
        memset(deep,0,sizeof(deep));
        memset(visit,0,sizeof(visit));
        ans=0;
        int f;
        while (scanf("%d",&f) && f>0)  {
            while (getchar()!=10)  {
                int t;
                scanf("%d",&t);
                a[f][t]=a[t][f]=1;
            }
        }
        dfs(1,0,0);
        printf("%d\n",ans);
    }
    return 0;
}
```

3.14 Strongly Connected Components

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

using namespace std;
```

```

vector<int> a[10010];
vector<int> b[10010];
stack<int> tt;
int fa[10010];
int d[10010];
int size[10010];
bool visit[10010];
int n,m;

void dfs(int k) {
  visit[k]=1;
  for (int i=0;i<a[k].size();i++)
    if (!visit[a[k][i]])
      dfs(a[k][i]);
  tt.push(k);
}

void dfs(int k,int FA) {
  fa[k]=FA;
  size[FA]++;
  visit[k]=1;
  for (int i=0;i<b[k].size();i++)
    if (!visit[b[k][i]])
      dfs(b[k][i],FA);
}

int main() {
  scanf("%d%d",&n,&m);
  for (int i=0;i<m;i++) {
    int f,t;
    scanf("%d%d",&f,&t);
    a[f].push_back(t);
    b[t].push_back(f);
  }
  memset(visit,0,sizeof(visit));
  for (int i=1;i<=n;i++)
    if (!visit[i])
      dfs(i);
  memset(visit,0,sizeof(visit));
  int s=0;
  for (;!tt.empty();tt.pop())
    if (!visit[tt.top()])
      dfs(tt.top(),++s);
  for (int i=1;i<=n;i++) {
    int f=fa[i];
    for (int j=0;j<b[i].size();j++) {
      int t=fa[b[i][j]];
      if (f!=t)
        d[t]++;
    }
  }
  vector<int> ans;
  for (int i=1;i<=s;i++)
    if (d[i]==0)
      ans.push_back(size[i]);
  if (ans.size()==1)
    printf("%d\n",ans[0]);
  else
    puts("0");
  return 0;
}
```

4 String Algorithm

4.1 ELF Hash

```
int elfhash(char *key) {
  unsigned int h = 0;
  while(*key) {
    h = (h << 4) + *key++;
    unsigned int g=h&0Xf0000000L;
    if (g) h ^= g >> 24;
    h &= ~g;
  }
  return h%MOD;
}
```

4.2 Aho-Corasick Automation

```
#include <cstdio>
#include <cstring>
#include <queue>

using std::queue;

void toInt(char s[]) {
  for (int i = 0; s[i]; i++) {
    if (s[i]=='A') s[i]='0';
    if (s[i]=='G') s[i]='1';
    if (s[i]=='T') s[i]='2';
    if (s[i]=='C') s[i]='3';
  }
}

struct trie{
  trie *next[4];
  trie *fail;
  bool isend;
};

trie pool[1010];
trie *head;
trie *root;

void insert(char s[]) {
  trie *now=root;
  for (;;) {
    if (s[0]==0) {
      now->isend=1;
      return;
    }
    int tt=s[0]-'0';
    if (now->next[tt]==NULL)
      now->next[tt]=++head;
    now=now->next[tt];
    s++;
  }
}

void buildFaliure() {
  queue<trie*> q;
  for (int i=0;i<4;i++)
    if (root->next[i])  {
      root->next[i]->fail=root;
      q.push(root->next[i]);
    } else root->next[i]=root;
  while (!q.empty()) {
    trie *now=q.front(); q.pop();
    for (int i=0;i<4;i++) {
      trie *u=now->next[i];
      if (u) {
        q.push(u);
        trie *v=now->fail;
        while (v->next[i]==NULL)
          v=v->fail;
        u->fail=v->next[i];
      }
    }
    if (now->fail->isend) now->isend=1;
  }
}

int dp[1010][1010];

trie* go(trie *now,char ch) {
  ch -= '0';
  trie *ans = now;
  while (ans -> next[ch] == NULL)
    ans = ans -> fail;
  return ans -> next[ch];
}

int main() {
  int ii=1;
  for (;;) {
    int n;
    scanf("%d",&n);
    if (n==0) break;
    root=head=pool;
    memset(dp,0x7f,sizeof(dp));
    memset(pool,0,sizeof(pool));

    static char buf[30];
    for (int i=0;i<n;i++) {
      scanf("%s",buf);
      toInt(buf);
      insert(buf);
    }
    buildFaliure();

    static char word[1010];
    scanf("%s",word);
    toInt(word);
    dp[0][0]=0;
    n=head-pool;
    int len;
    for (len=0;word[len];len++) {
      for (int j=0;j<n;j++)
        if (dp[j][len]<=len) {
          for (char ch='0';ch<='3';ch++) {
            trie *tmp=go(pool+j,ch);
            if (tmp->isend) continue;
            int next=tmp-pool;
            int delta=0; if (ch!=word[len]) delta++;
            if (dp[next][len+1] > dp[j][len] + delta)
              dp[next][len+1] = dp[j][len] + delta;
          }
        }
    }
    int ans=2000000000;
    for (int j=0;j<n;j++)
      if (dp[j][len]<ans) ans=dp[j][len];
    if (ans>len) ans=-1;
    printf("Case %d: %d\n",ii++,ans);
  }
  return 0;
}
```

4.3 Suffix array

```
int wa[maxn],wb[maxn],wv[maxn],ws[maxn];
int cmp(int *r,int a,int b,int l)
{return r[a]==r[b]&&r[a+l]==r[b+l];}
void da(int *r,int *sa,int n,int m) {
  int i,j,p,*x=wa,*y=wb,*t;
  for(i=0;i<m;i++) ws[i]=0;
  for(i=0;i<n;i++) ws[x[i]=r[i]]++;
  for(i=1;i<m;i++) ws[i]+=ws[i-1];
  for(i=n-1;i>=0;i--) sa[--ws[x[i]]]=i;
  for(j=1,p=1;p<n;j*=2,m=p) {
    for(p=0,i=n-j;i<n;i++) y[p++]=i;
    for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
    for(i=0;i<n;i++) wv[i]=x[y[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--) sa[--ws[wv[i]]]=y[i];
    for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
      x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
  }
}
```

5 Data Struct

5.1 Binary Indexed Tree

BECAREFUL WHILE I == 0 !!!

```
int sum(int k) {
  int ans = 0;
  for (int i = k; i > 0; i -= i & -i)
    ans += a[i];
  return ans;
}

void change(int k,int n,int delta) {
  for (int i = k; i <= n; i += i & -i)
    a[i] += delta;
}
```

5.2 Inversion

```
#include <cstdio>

int a[500010];
int t[500010];
long long ans;

void merge(int a[],int sizea,int b[],int sizeb) {
    int nowa = 0;
    int nowb = 0;
    int s = 0;
    while (nowa < sizea && nowb < sizeb) {
        if (a[nowa]<=b[nowb])
            t[s++]=a[nowa++];
        else
        if (a[nowa]>b[nowb]) {
            t[s++] = b[nowb++];
            ans += sizea - nowa;
        }
    }
    while (nowa<sizea)
        t[s++]=a[nowa++];
    while (nowb<sizeb)
        t[s++]=b[nowb++];
}

void sort(int a[],int size) {
    if (size < 2)
        return;
    int lsize = size>>1;
    int rsize = size-lsize;
    sort(a, lsize);
    sort(a + lsize, rsize);
    merge(a, lsize, a+lsize, rsize);
    for (int i = 0; i < size; i++)
        a[i] = t[i];
}

```

```

int main() {
    while (1) {
        int n;
        scanf("%d",&n);
        if (!n)
            break;
        for (int i=0;i<n;i++)
            scanf("%d",a+i);
        ans = 0;
        sort(a,n);
        printf("%lld\n",ans);
    }
}
```

5.3 BigInt Multiply with FFT

```
#include <cstdio>
#include <cstring>
#include <cmath>

typedef long long Long;
const int MAXN=32768;
const double pi=acos(-1.0);
const Long MOD=100000;
const int TEN=5;

double ra[MAXN];
double ia[MAXN];
double rb[MAXN];
double ib[MAXN];
double rc[MAXN];
double ic[MAXN];
char a[MAXN];
char b[MAXN];
int slena;
int slenb;
int lena;
int lenb;
int n,logn;
Long ans[MAXN];

int rev(int x,int bit)
{
        int ans=0;
        for (int i=0;i<bit;i++)
        {
                ans<<=1;
                if (x&1) ans|=1;
                x>>=1;
        }
        return ans;
}

void fft(double ir[],double ii[],int size,int mark)
{
        static double R[MAXN];
        static double I[MAXN];
        double delta=mark*2*pi;
        for (int i=0;i<size;i++)
        {
                int tt=rev(i,logn);
                R[tt]=ir[i];
                I[tt]=ii[i];
        }
        for (int s=1;s<=logn;s++)
        {
                int m=1<<s;
                double rwm=cos(delta/m);
                double iwm=sin(delta/m);
                for (int k=0;k<n;k+=m)
                {
                        double rw=1;
                        double iw=0;
                        for (int j=0;j<m/2;j++)
                        {
                                // t=w*A[k+j+m/2];
                                double rt=rw*R[k+j+m/2]-iw*I[k+j+m/2];
                                double it=rw*I[k+j+m/2]+iw*R[k+j+m/2];
                                // u=A[k+j];
                                double ru=R[k+j];
                                double iu=I[k+j];

                                // A[k+j]=u+t;
                                R[k+j]=ru+rt;
                                I[k+j]=iu+it;

                                //A[k+j+m/2]=u-t;
                                R[k+j+m/2]=ru-rt;
                                I[k+j+m/2]=iu-it;

                                double rnw=rw*rwm-iw*iwm;
                                double inw=rw*iwm+iw*rwm;
                                rw=rnw; iw=inw;
                        }
                }
        }
        for (int i=0;i<size;i++)
        {
                ir[i]=R[i];
                ii[i]=I[i];
        }
}

double POW[10] = {1,1E1,1E2,1E3,1E4,1E5,1E6,1E7,1E8,1E9};

int next(char str[])
{
        int len=0;
        for (str[len]=getchar();str[len]>='0';str[len]=getchar())
                len++;
        str[len]=0;
        return len;
}

```

```
int main()
{
        int nn=0;
        scanf("%d",&nn); getchar();
        while (nn--)
        {
                memset(ra,0,n<<3);
                memset(ia,0,n<<3);
                memset(rb,0,n<<3);
                memset(ib,0,n<<3);
                memset(ans,0,n<<3);

                slena=next(a);
                int cnt=0; lena=0;
                for (int j=slena-1;j>=0;j--)
                {
                        ra[lena]=ra[lena]+(a[j]-'0')*POW[cnt++];
                        if (cnt==TEN) {lena++; cnt=0;}
                }
                if (ra[lena]>0.1)       lena++;

                slenb=next(b);
                cnt=0; lenb=0;
                for (int j=slenb-1;j>=0;j--)
                {
                        rb[lenb]=rb[lenb]+(b[j]-'0')*POW[cnt++];
                        if (cnt==TEN) {lenb++; cnt=0;}
                }
                if (rb[lenb]>0.1)       lenb++;

                n=1; logn=0;
                while (n<lena || n<lenb) {n+=n;logn++;}
                n+=n; logn++;

                fft(ra,ia,n,1);
                fft(rb,ib,n,1);
                for (int i=0;i<n;i++)
                {
                        rc[i]=ra[i]*rb[i]-ia[i]*ib[i];
                        ic[i]=ra[i]*ib[i]+rb[i]*ia[i];
                }
                fft(rc,ic,n,-1);
                for (int i=0;i<n;i++)
                        ans[i]=(Long)(rc[i]/n+0.5);
                for (int i=0;i<n-1;i++)
                {
                        ans[i+1]+=ans[i]/MOD;
                        ans[i]%=MOD;
                }
                bool print=0;
                for (int i=n-1;i>=0;i--)
                {
                        if (!print && (ans[i]>0 || i==0))
                        {
                                print=1;
                                printf("%lld",ans[i]);
                        } else
                        if (print)
                                printf("%05lld",ans[i]);
                }
                putchar(10);
            }
            return 0;
        }
```

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;}
```

6.3 3D Point and Vector

```
struct Point{
  double x,y,z;
  explicit Point(const double& a=0,const double& b=0,const double& c=0):x(a),y(b),z(c){}
  Point(const Point& p):x(p.x),y(p.y),z(p.z){}
  Point operator-() const{return Point(-x,-y,-z);}
  Point operator-(const Point& a) const{return Point(x-a.x,y-a.y,z-a.z);}
  Point operator+(const Point& a) const{return Point(x+a.x,y+a.y,z+a.z);}
  Point operator*(const double& s) const{return Point(s*x,s*y,s*z);}
  double sqlen() const{return x*x+y*y+z*z;}
  double len() const{return sqrt(sqlen());}
  void norm(){double l=len();x/=l;y/=l;z/=l;}
};
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

```
inline bool line_mindist(const Point& p0,const Vec& d0,
const Point& p1,const Vec& d1,
double* t,double* k){
  double a=dot(d0,d0),b=-dot(d0,d1),c=-dot(d1,d1),d=dot(d0,p1-p0),
  e=dot(d1,p1-p0);
  double det=a*c+b*b;
  if(equ0(det)) return false;
  *t=(d*c-b*e)/det;
  *k=(a*e+b*d)/det;
  return true;
}
```

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

```
struct Point{
  double x,y;
  explicit Point(const double& a=0,const double& b=0):x(a),y(b){}
  Point(const Point& p):x(p.x),y(p.y){}
  Point& operator=(const Point& p){x=p.x;y=p.y;return *this;}
  Point operator-() const{return Point(-x,-y);}
  Point operator-(const Point& a) const{return Point(x-a.x,y-a.y);}
  Point operator+(const Point& a) const{return Point(x+a.x,y+a.y);}
  Point operator*(const double& s) const{return Point(s*x,s*y);}
  double sqlen() const{return x*x+y*y;}
  double len() const{return sqrt(sqlen());}
  void norm(){double l=len();x/=l;y/=l;}
};
typedef Point Vec;
```

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

```
inline bool seg_overlap(const Point& a1,const Point& a2,const Point& b1,const Point& b2){
  if(coline(a1,a2,b1) && coline(a1,a2,b2)){
    double k1,k2;
    Vec axis=a2-a1;
    Vec temp=b1-a1;
    if(equ0(axis.y)){
      k1=temp.x/axis.x;
      temp=b2-a1;
      k2=temp.x/axis.x;
    }
    else{
      k1=temp.y/axis.y;
      temp=b2-a1;
      k2=temp.y/axis.y;
    }
    if((gr(k1,1.0f) && gr(k2,1.0f)) || (ls(k1,0.0f) && ls(k2,0.0f))) return false;
    else return true;
  }
  else return false;
}
```

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

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

6.6.7 2D circle - circle tangent

require circles no touch, no intersect
POJ 2416

```
struct Seg{
  Point p0,p1;
  Seg(const Point& m=Point(),const Point& n=Point()):p0(m),p1(n){
  }
};

inline void tan_2circle(const Point& C0,const double& R0,
const Point& C1,const double& R1,
Seg* segs){
  Point c0=C0,c1=C1;double r0=R0,r1=R1;
  if(gr(R1,R0)){swap(c0,c1);swap(r0,r1);}
  Vec v=c1-c0;double cdist=v.len();

  Point p=c0+v*(r0/(r1+r0));
  Point t[2];
  point_circle_tan(p,c0,r0,t);
  Vec u=p-t[0];Point w;
  w=t[0]+u*((r1+r0)/r0);
  segs[0]=Seg(t[0],w);
  u=p-t[1];w=t[1]+u*((r1+r0)/r0);
  segs[1]=Seg(t[1],w);

  if(equ(r0,r1)){
    u=v;u.norm();u=u*r0;
    Vec offset=vec_rotate(u,pi*0.5);
    segs[2]=Seg(c0+offset,c1+offset);
    offset=vec_rotate(u,-pi*0.5);
    segs[3]=Seg(c0+offset,c1+offset);
  }
  else{
    p=c0+v*(r0/(r0-r1));
    Point k[2];
    point_circle_tan(p,c1,r1,t);point_circle_tan(p,c0,r0,k);
    segs[2]=Seg(t[0],k[0]);
    segs[3]=Seg(t[1],k[1]);
  }
}
```

6.6.8 2D circle - polygon intersection area

POJ 3675
schindlerlee, thanks

```
//CCW
inline double poly_area(const vector<Point>& p){
  Point prev=p[0],cur;
  double area=0;
  for(int i=1;i<p.size();++i){
    cur=p[i];
    area+=0.5 * cross(prev,cur);
    prev=cur;
  }
  return area;
}

double R;
Point C;
vector<Point> P;
int main(){
  while(scanf("%lf",&R)==1){
  P.clear();
  int N;
  scanf("%d",&N);
  for(int i=0;i<N;++i){
    double x,y;scanf("%lf%lf",&x,&y);
    P.push_back(Point(x,y));
  }
  C=Point(0.0f,0.0f);
  P.push_back(P[0]);
  double area=0.0f;
  double t,k;
  for(int i=0;i<P.size()-1;++i){
    Point s0=P[i],s1=P[i+1];
    if(line_circle_intersect(s0,s1-s0,C,R,&t,&k)){
      Point p0=s0+(s1-s0)*t;
      Point p1=s0+(s1-s0)*k;
      if(lseq(k,0.0f) || greq(t,1.0f)){
        area+=0.5 * R*R * angle_diff(s0,s1);
      }
      else if(lseq(t,0.0f) && greq(k,1.0f)){
        area+=0.5 * cross(s0,s1);
      }
      else if(lseq(t,0.0f) && greq(k,0.0f) && lseq(k,1.0f)){
        area+=0.5 * cross(s0,p1);
        area+=0.5 * R*R * angle_diff(p1,s1);
      }
```

```
        else if(greq(k,1.0f) && greq(t,0.0f) && lseq(t,1.0f)){
          area+=0.5 * R*R * angle_diff(s0,p0);
          area+=0.5 * cross(p0,s1);
        }
        else{
          area+=0.5 * R*R * angle_diff(s0,p0);
          area+=0.5 * cross(p0,p1);
          area+=0.5 * R*R * angle_diff(p1,s1);
        }
      }
      else area+=0.5f * R*R * angle_diff(s0,s1);
    }
    printf("%.2f\n",fabs(area));
  }//while case
  return 0;
}
```

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

```
  inline bool ray_intersect(const Point& p,const Vec& u,
  const Point& q,const Vec& v,
  double* t,double* k){
    double a=u.x,b=-v.x,c=u.y,d=-v.y;
    double e=q.x-p.x,f=q.y-p.y;
    double det=a*d-b*c;
    if(equ0(det)) return false;
    else{
      *t=(e*d-b*f)/det;
      *k=(a*f-e*c)/det;
      return true;
    }
  }
```

6.7.2 line - circle intersection

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

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;
Point P[MAXP];
int N;

int H[MAXP],lower[MAXP];
bool vis[MAXP];
int top;
int monotone_chain(){
  memset(vis,0,sizeof(vis));
  int utop=0,ltop=0;
  for(int i=N-1;i>=0;--i){
    if(utop<2) H[utop++]=i;
    else{
      while(utop>1 && cross(P[H[utop-1]]-P[H[utop-2]],P[i]-P[H[utop-1]]) <= 0) --utop;
      H[utop++]=i;
    }
  }
  for(int i=0;i<utop;++i) vis[H[i]]=true;
  for(int i=0;i<N;++i){
    if(ltop<2) lower[ltop++]=i;
    else{
      while(ltop>1 && cross(P[lower[ltop-1]]-P[lower[ltop-2]],P[i]-P[lower[ltop-1]]) <= 0)
      --ltop;
      lower[ltop++]=i;
    }
  }
  for(int i=0;i<ltop;++i) if(!vis[lower[i]]) H[utop++]=lower[i];
  return utop;
}
```

6.8.2 3D convec hull

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

```
//CCW
struct Face{
  Point p0,p1,p2;
  Face(const Point& a=Point(),const Point& b=Point(),const Point& c=Point()):
  p0(a),p1(b),p2(c){
  }
};
inline Vec face_normal(const Face& f){
  Vec u=f.p1-f.p0;
  Vec v=f.p2-f.p0;
  Vec n=cross(u,v);
  n.norm();
  return n;
}
inline bool point_above_face(const Point& p,const Face& f){
  Vec u=p-f.p0;
  Vec n=face_normal(f);
  return gr(dot(u,n),0.0f);
}


int edge[MAXP][MAXP];
int main(){
  int N;
  vector<Point> Points;
  vector<Face> CH;
  while(scanf("%d",&N)==1){
  for(int i=0;i<N;++i){
    double x,y,z;
    scanf("%lf%lf%lf",&x,&y,&z);
    Points.push_back(Point(x,y,z,i));
  }
  //special case

  CH.push_back(Face(Points[0],Points[1],Points[2]));
  CH.push_back(Face(Points[2],Points[1],Points[0]));

  for(int i=3;i<Points.size();++i){
    Point cur=Points[i];
    for(int j=0;j<CH.size();++j){
      Face f=CH[j];
      if(point_above_face(cur,f)){
        edge[f.p0.id][f.p1.id]=1;
        edge[f.p1.id][f.p2.id]=1;
        edge[f.p2.id][f.p0.id]=1;
      }
      else{
        edge[f.p0.id][f.p1.id]=-1;
        edge[f.p1.id][f.p2.id]=-1;
        edge[f.p2.id][f.p0.id]=-1;
      }
    }
    vector<Face> T;
    for(int j=0;j<CH.size();++j){
      Face f=CH[j];
      if(edge[f.p0.id][f.p1.id]==1){
        if(edge[f.p1.id][f.p0.id]==-1) T.push_back(Face(f.p0,f.p1,cur));
        if(edge[f.p2.id][f.p1.id]==-1) T.push_back(Face(f.p1,f.p2,cur));
        if(edge[f.p0.id][f.p2.id]==-1) T.push_back(Face(f.p2,f.p0,cur));
      }
      else{
        T.push_back(f);
      }
    }
    swap(CH,T);
  }
  double area=0.0f;
  for(int i=0;i<CH.size();++i){
    Face f=CH[i];
    Vec crs=cross(f.p1-f.p0,f.p2-f.p0);
    area+=0.5f * crs.len();
  }
  printf("%.3f\n",area);
}
return 0;
}
```

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;}
int max_dist;
void rc(){
  int s;
  int p=0,q=Next(p);
  while(area(P[H[p]],P[H[Next(p)]],P[H[Next(q)]]) >
  area(P[H[p]],P[H[Next(p)]],P[H[q]])){
    q=Next(q);
  }
  s=p;
  do{
    Vec d=P[H[p]]-P[H[q]]; int temp=d.sqlen();
    max_dist=max(max_dist,temp);
    int np=p,nq=q;
    if(area(P[H[p]],P[H[Next(p)]],P[H[Next(q)]]) >
    area(P[H[p]],P[H[Next(p)]],P[H[q]])) nq=Next(q);
    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);
    if(nq!=q) q=Next(q);
    else p=Next(p);
  } while(q!=Next(s));
}
```

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);
  }
};
```

```
inline double area(const Point& a,const Point& b,const Point& c){return cross(b-a,c-a);}

const int MAXP=10000;
int M,N;
Point P[MAXP],Q[MAXP];

double mindist;
```

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

```
inline int nextp(int i){return (i+1)%M;}
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,
                         const Point& c,const Point& d){
  Vec u=b-a;
  Vec r0=vec_rotate_left(u),r1=vec_rotate_left(u);
  double t,k;
  if(ray_intersect(a,r0,c,d-c,&t,&k)) if(ls(k,1) && gr(k,0)) return true;
  if(ray_intersect(b,r1,c,d-c,&t,&k)) if(ls(k,1) && gr(k,0)) return true;
  Vec v=d-c;
  r0=vec_rotate_left(v);r1=vec_rotate_left(v);
  if(ray_intersect(c,r0,a,b-a,&t,&k)) if(ls(k,1) && gr(k,0)) return true;
  if(ray_intersect(d,r1,a,b-a,&t,&k)) if(ls(k,1) && gr(k,0)) return true;
  return false;
}
```

return two parallel segments' distance

```
inline double dist_2seg(const Point& a,const Point& b,
                        const Point& c,const Point& d){
  return dist2line(c,a,b-a);
}
```

rotating calipers

```
void rc(){
  int p,q;
  int sp;
  int cnt=0;
  double f;
  f=10001;
  for(int i=0;i<M;++i) if(P[i].x<f){f=P[i].x;p=i;}
  f=-10001;
  for(int i=0;i<N;++i) if(Q[i].x>f){f=Q[i].x;q=i;}
  while(gr(area(P[p],P[nextp(p)],Q[nextq(q)]),area(P[p],P[nextp(p)],Q[q]))) q=nextq(q);
  sp=p;

  if(point_over_seg(Q[q],P[p],P[nextp(p)]))
    mindist=min(mindist,dist2line(Q[q],P[p],P[nextp(p)]-P[p]));
  else mindist=min(mindist,(P[p]-Q[q]).len());
  do{
    int np=p,nq=q;
    if(gr(area(P[p],P[nextp(p)],Q[nextq(q)]),area(P[p],P[nextp(p)],Q[q]))) nq=nextq(q);
    if(gr(area(Q[q],Q[nextq(q)],P[nextp(p)]),area(Q[q],Q[nextq(q)],P[p]))) np=nextp(p);
    if(nq==q && np!=p){
      if(point_over_seg(Q[q],P[p],P[nextp(p)]))
        mindist=min(mindist,dist2line(Q[q],P[p],P[nextp(p)]-P[p]));
      else mindist=min(mindist,(Q[q]-P[nextp(p)]).len());
      p=nextp(p);
    }
    else if(np==p && nq!=q){
      if(point_over_seg(P[p],Q[q],Q[nextq(q)]))
        mindist=min(mindist,dist2line(P[p],Q[q],Q[nextq(q)]-Q[q]));
      else mindist=min(mindist,(P[p]-Q[nextq(q)]).len());
      q=nextq(q);
    }
    else if(np==p && nq==q){
      if(seg_over_seg(P[p],P[nextp(p)],Q[q],Q[nextq(q)]))
        mindist=min(mindist,dist_2seg(P[p],P[nextp(p)],Q[q],Q[nextq(q)]));
      else{
        //mindist=min(mindist,(P[nextp(p)]-Q[nextq(q)]).len());
        mindist=min(mindist,(P[nextp(p)]-Q[q]).len());
        //mindist=min(mindist,(Q[nextq(q)]-P[p]).len());
      }
      p=nextp(p);
    }
    if(p==sp) ++cnt;
  }while(!(cnt>1&&p==nextp(sp)));
}
```

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;

Point P[MAXN];
int N;
struct compx{
  bool operator()(const Point& a,const Point& b)const{
    if(a.x==b.x) return ls(a.y,b.y);
    else return ls(a.x,b.x);
  }
};
struct compy{
  bool operator()(const Point& a,const Point& b)const{
    if(a.y==b.y) return ls(a.x,b.x);
    else return ls(a.y,b.y);
  }
};
```

sweep line algorithm

```
double mindist;

Point eventp;
typedef set<Point,compy> Box;
typedef Box::iterator BoxIter;
void sweepline(){
  sort(P,P+2*N,compx());
  mindist=INF;
  int l=0;
  Box box;
  for(int i=0;i<2*N;++i){
    eventp=P[i];
    double x=eventp.x-mindist,negy=eventp.y-mindist,posy=eventp.y+mindist;

    while(l<i && ls(P[l].x,x)){
      box.erase(P[l]);
      ++l;
    }

    BoxIter a=box.lower_bound(Point(x,negy));
    BoxIter b=box.lower_bound(Point(x,posy));
    for(BoxIter iter=a;iter!=b;iter++){
      Vec d=*iter - eventp;
      if(iter->color != eventp.color) mindist=min(mindist,d.len());
    }
    box.insert(eventp);
  }
}
```

6.8.6 closed circle pair

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

```
struct Circle{
  Point c;double r;
  Circle(const Point& a=Point(),const double& b=0):c(a),r(b){
  }
};
const int MAXN=50000;
Circle C[MAXN];
int N;
Point P[MAXN];

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

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

```
double mindist;

Point eventp;
typedef set<Point,compy> Box;
typedef Box::iterator BoxIter;
void sweepline(){
  sort(P,P+N,compx());
  mindist=INF;
  int l=0;
  Box box;
  for(int i=0;i<N;++i){
    eventp=P[i];
    Circle cir=C[eventp.c];
    double x=eventp.x-mindist,negy=cir.c.y-cir.r-mindist,posy=cir.c.y+cir.r+mindist;

    while(l<i && ls(C[P[l].c].c.x+C[P[l].c].r,x)){
      Circle tmp=C[P[l].c];
      box.erase(Point(tmp.c.x,tmp.c.y-tmp.r));
      box.erase(Point(tmp.c.x,tmp.c.y+tmp.r));
      ++l;
    }

    if(!box.empty()){
      BoxIter begin=box.lower_bound(Point(NEGINF,negy));
      BoxIter end=box.lower_bound(Point(NEGINF,posy));
      for(BoxIter iter=begin;iter!=end;++iter){
        Circle tmp=C[iter->c];
        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(){
  int T;scanf("%d",&T);
  while(T-->0){
    scanf("%d",&N);
    for(int i=0;i<N;++i){
      int x,y,r;
      scanf("%d%d%d",&x,&y,&r);
      C[i].c.x=x;C[i].c.y=y;C[i].r=r;
      P[i]=Point(C[i].c.x-C[i].r,C[i].c.y,i);
    }
    sweepline();
    printf("%.6f\n",mindist);
  }
  return 0;
}
```

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;
const int MAXN=2*MAXC;
struct Circle{
  Point c;double r;
  Circle(const Point& a=Point(),const double& b=0):c(a),r(b){}
};
//a extra INF radius circle
Circle circle[MAXC+1];
Point P[MAXN];
int N;
Point eventp;

struct compx{
  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);
  }
};

struct Interval{
  int p;
  int i,j;
  bool lowi,lowj;
  Interval(int par=0,int a=0,bool a0=false,int b=0,bool b0=false):
    p(par),i(a),j(b),lowi(a0),lowj(b0){}
};
inline pair<double,double> get_itv(const Circle& c,const double& x){
  double d=fabs(c.c.x-x);
  double del=c.r*c.r - d*d;
  if(lseq(del,0.0f)) del=0.0f;
  del=sqrt(del);
  return make_pair(c.c.y-del,c.c.y+del);
}
struct compcircle{
  bool operator()(const Interval& a,const Interval& b)const{
    pair<double,double> itv0=get_itv(circle[a.i],eventp.x);
    pair<double,double> itv1=get_itv(circle[a.j],eventp.x);
    double x0,x1;
    if(a.lowi) x0=itv0.first;else x0=itv0.second;
    if(a.lowj) x1=itv1.first;else x1=itv1.second;

    itv0=get_itv(circle[b.i],eventp.x);
    itv1=get_itv(circle[b.j],eventp.x);
    double y0,y1;
    if(b.lowi) y0=itv0.first;else y0=itv0.second;
    if(b.lowj) y1=itv1.first;else y1=itv1.second;

    if(equ(x0,y0)) return ls(x1,y1);
    else return ls(x0,y0);
  }
};

//------------union set--------------
int parent[MAXC+1];
inline void make_set(){for(int i=0;i<=N;++i) parent[i]=i;}
//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> Itvl;
typedef Itvl::iterator Iter;
void debug(Itvl itvl){
  for(Iter k=itvl.begin();k!=itvl.end();++k){
    printf("parent: %d lower: %d %d  upper: %d %d\n",k->p,k->i,k->lowi,
        k->j,k->lowj);
  }
}
void sweepline(){
  make_set();
  sort(P,P+2*N,compx());
  Itvl itvl;
  itvl.insert(Interval(0,0,true,0,false));

  for(int i=0;i<2*N;++i){
    eventp=P[i];

    if(eventp.in){
      Iter lower=itvl.lower_bound(Interval(0,eventp.i,true,eventp.i,false));
      --lower;
      link(lower->p,eventp.i);
      Interval tmp=*lower;
      itvl.erase(lower);
      itvl.insert(Interval(tmp.p,tmp.i,tmp.lowi,eventp.i,true));
      itvl.insert(Interval(eventp.i,eventp.i,true,eventp.i,false));
      itvl.insert(Interval(tmp.p,eventp.i,false,tmp.j,tmp.lowj));
    }
    else{
      Iter d=itvl.find(Interval(0,eventp.i,true,eventp.i,false));
      Iter l=d;--l;
      Iter u=d;++u;
      Interval low=*l,up=*u;
      itvl.erase(d);itvl.erase(l);itvl.erase(u);
      itvl.insert(Interval(low.p,low.i,low.lowi,up.j,up.lowj));
    }
  }
}
```

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

```
void sweepline(){
  make_set();
  sort(P,P+2*N,compx());
  Itvl itvl;
  itvl.insert(Interval(0,0,true,0,false));

  for(int i=0;i<2*N;++i){
    eventp=P[i];

    if(eventp.in){
      Iter lower=itvl.lower_bound(Interval(0,eventp.i,true,eventp.i,false));
      --lower;
      link(lower->p,eventp.i);
      if(lower->p == 0){
        Interval tmp=*lower;
        itvl.erase(lower);
        itvl.insert(Interval(tmp.p,tmp.i,tmp.lowi,eventp.i,true));
        itvl.insert(Interval(eventp.i,eventp.i,true,eventp.i,false));
        itvl.insert(Interval(tmp.p,eventp.i,false,tmp.j,tmp.lowj));
      }
    }
    else{
      Iter d=itvl.find(Interval(0,eventp.i,true,eventp.i,false));
      if(d==itvl.end()) continue;
      Iter l=d;--l;
      Iter u=d;++u;
      Interval low=*l,up=*u;
      itvl.erase(d);itvl.erase(l);itvl.erase(u);
      itvl.insert(Interval(low.p,low.i,low.lowi,up.j,up.lowj));
    }
  }
}
```

7 Math

7.1 Catalan Number