ACM/ICPC TEMPLATE

NKU -> HOT

October 17, 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;
```

```
 8 const int EXP = 10;
 9 const int INF = 1000000000;
10
11 int nn,mm;
12
13 int edges;
14 struct EDGE{
15   int n;
16   int v;
17   EDGE* nxt;
18 }pool[MAXM*2+EXP];
19 EDGE lnk[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){
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;
49   priority_queue <NODE> Q; Q.push(NODE(1,0));
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 = lnk[now.n].nxt; tmp; tmp = tmp->nxt){
56       if(!inSet[tmp->n] && tmp->v < dist[tmp->n]){
57         dist[tmp->n] = tmp->v;
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--){
68      scanf("%d%d", &nn, &mm);
69      edges = 0;
70      for(int i = 1; i <= nn; i++) lnk[i].nxt = 0;
71      for(int i = 1; i <= mm; i++){
72        int aa,bb,vv; scanf("%d%d%d", &aa, &bb, &vv);
73        addEdge(aa, bb, vv);
74      }
75      Prim_Prio();
76      printf("%d\n",sum);
77    }
78    return 0;
79  }
```

3.3 Kruskal -O(MlogM)

```
1   #include <iostream>
2   #include <cstdio>
3   #include <algorithm>
4   using namespace std;
5
6   const int MAXN = 100;
7   const int MAXM = 10000;
8   const int EXP = 10;
9   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;
47   for(int i = 0; i < mm; i++){
48     int aa = find(pool[i].f);
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   int cas;    scanf("%d", &cas);
59   while(cas--){
60     scanf("%d%d", &nn, &mm);
61     for(int i = 0; i < mm; i++)
62       scanf("%d%d%d", &pool[i].f, &pool[i].t, &pool[i].v);
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>
4  #include <queue>
5  using namespace std;
6
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 <= nn; i++)
21     for(int j = 0; j <= nn; j++)
22       map[i][j] = INF;
23 }
24
25 void Dijk(int s){
26   for(int i = 1; i <= nn; i++){
27     dist[i] = INF;
```

```
28      inSet[i] = 0;
29    }
30    dist[s] = 0;
31    for(int i = 1; i <= 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 <= 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--){
50      scanf("%d%d", &nn, &mm);
51      init();
52      for(int i = 1; i <= mm; i++){
53        int aa,bb,dd; scanf("%d%d%d", &aa, &bb, &dd);
54        if(map[aa][bb] > dd){
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)

```
1   #include <iostream>
2   #include <cstdio>
3   #include <cstring>
4   #include <queue>
5   using namespace std;
6
7   const int MAXN = 50000;
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;
16    EDGE *nxt;
17  }pool[MAXM*2+EXP];
18  EDGE lnk[MAXN+EXP];
19
```

```
20 void addEdge (int _f, int _t, int _d){
21   pool[edges].n = _t;
22   pool[edges].d= _d;
23   pool[edges].nxt = lnk[_f].nxt;
24   lnk[_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;
37   NODE(int _n = 0, int _dst = 0){
38     n = _n;
39     dst = _dst;
40   }
41 };
42
43 bool operator <(NODE aa, NODE bb){
44   return aa.dst > bb.dst;
45 }
46
47 void Dijk_Prio(int s){
48   for(int i = 1; i <= 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;
58     inSet[now.n] = 1;
59     for(EDGE * tmp = lnk[now.n].nxt; tmp; tmp = tmp->nxt){
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);
70   while(cas--){
71     edges = 0;
72     scanf("%d%d", &nn, &mm);
73     for(int i = 1; i <= nn; i++) lnk[i].nxt = 0;
74     for(int i = 1; i <= 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

```
1   #include <cstdio>
2   #include <cstring>
3
4   using namespace std;
5
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) {
16      for (int i = 2; i <= n; i++) {
17        data[i] = inf;
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])
30          son++;
31        if (data[x] > data[son]) {
32          int tmp=data[x]; data[x]=data[son]; data[son]=tmp;
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) {
40      for (int fa = x>>1; x > 1; x = fa, fa = x>>1) {
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];
56      index[1]=index[tot];
57      data[1]=data[tot];
58      pos[x]=0;
59      pos[index[tot--]]=1;
60      fix_down(1);
61    }
62    bool empty() {
63      return tot==0;
64    }
65  };
66
67  int a[1010][2000];
68  int b[1010][2000];
69  int dist[1010];
70  bool visit[1010];
71
72  DJ_heap q;
73
74  int main() {
75    int n,m;
76    scanf("%d%d",&m,&n);
77    while (m--) {
78      int f,t,cost;
79      scanf("%d%d%d",&f,&t,&cost);
80      a[f][++a[f][0]]=t;
81      b[f][++b[f][0]]=cost;
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<=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    }
98    printf("%d\n",dist[1]);
99    return 0;
100 }
```

3.7 Bellman-Ford

```
1  #include <iostream>
2  #include <cstdio>
```

```
3
4  using namespace std;
5
6  const int MAXN = 1000;
7  const int MAXM = 2000;
8  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++){
20     scanf("%d%d%d",vf+i,vt+i,vc+i);
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++){
29       if(dist[vf[i]] + vc[i] < dist[vt[i]]){
30         dist[vt[i]] = dist[vf[i]] + vc[i];
31       }
32       if(dist[vt[i]] + vc[i] < dist[vf[i]]){
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

```
1  #include <iostream>
2  #include <cstdio>
3  #include <cstring>
4  #include <queue>
5  using namespace std;
6
7  const int MAXN = 50000;
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;
16   EDGE *nxt;
17 }pool[MAXM*2+EXP];
18 EDGE lnk[MAXN+EXP];
19
20 void addEdge (int _f, int _t, int _d){
21   pool[edges].n = _t;
22   pool[edges].d= _d;
23   pool[edges].nxt = lnk[_f].nxt;
24   lnk[_f].nxt = &pool[edges];
25   edges++;
26 }
27
28 int nn;
29 int mm;
30
31 bool inQ[MAXN+EXP];
32 int dist[MAXN+EXP];
33
34 void spfa(int s){
35   for(int i = 0; i <= nn; i++){
36     inQ[i] = 0;
37     dist[i] = INF;
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;
47         if(!inQ[tmp->n]) {
48           Q.push(tmp->n);
49           inQ[tmp->n] = 1;
50         }
51       }
52     }
53   }
54 }
55
56 int main(){
57   int cas; scanf("%d", &cas);
58   while(cas--){
59     edges = 0;
60     scanf("%d%d", &nn, &mm);
61     for(int i = 1; i <= nn; i++) lnk[i].nxt = 0;
62     for(int i = 1; i <= mm; i++){
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

```
 1 #include <cstring>
 2 #include <cstdio>
 3 #include <queue>
 4 #include <algorithm>
 5 #include <vector>
 6
 7 using namespace std;
 8
 9 const int MAXN = 210;
10 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,sizeof(p));eid=0;}
18   inline void insert1(int from,int to,int val) {
19     e[eid].v=to;
20     e[eid].val=val;
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;
```

```
54            else d=e[j].val;
55            d=dfs(v,d);
56            e[j].val-=d;
57            e[j^1].val+=d;
58            lv-=d;
59            if (h[source]>=n) return cost-lv;
60            if (lv==0) break;
61          }
62          if (h[v]<minh)  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    }
73    int run() {
74      int ret=0;
75      memset(gap,0,sizeof(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

```
1  #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) {
```

```
    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') {
```

```
27          b[++sb][0]=i;
28          b[sb][1]=j;
29        }
30      }
31    }
32    s=sa;
33    for (int i = 1;i <= 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+j][i] = cost[i][s+j];
37        maxf[i][s+j] = 1;
38      }
39    }
40    for (int i = 1; i <= s; i++)
41      maxf[0][i]=maxf[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,sizeof(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;
59      for (int i = 0; i <= n;i++)
60        if ((maxf[now][i] - flow[now][i] > 0)
61            && dist[now] + value(now,i) < dist[i]){
62          dist[i] = dist[now] + value(now,i);
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();
77      while (spfamark()) {
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;
```

```
  }
  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++)
```

```
51            if(vis_y[i])
52              ly[i] += slack;
53        }
54      }
55      int sum = 0;
56      for(int v = 1; v <= M; v++) sum += W[match[v]][v];
57      return -sum;
58    }
59  } km;
```

3.13 Cut Vetrix and Edge

```
1  #include <cstdio>
2  #include <cstring>
3  #include <algorithm>
4
5  using namespace std;
6
7  bool a[110][110];
8  int visit[110];
9  int deep[110];
10 int back[110];
11 bool cut[110];
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
37 int main() {
38     while (1) {
39         scanf("%d",&n);
40         if (n==0)
41             break;
42         memset(a,0,sizeof(a));
43         memset(back,0,sizeof(back));
44         memset(cut,0,sizeof(cut));
45         memset(deep,0,sizeof(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;
52                  scanf("%d",&t);
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

```
 1  #include <cstdio>
 2  #include <cstring>
 3  #include <stack>
 4  #include <vector>
 5
 6  using namespace std;
 7
 8  vector<int> a[10010];
 9  vector<int> b[10010];
10  stack<int> tt;
11  int fa[10010];
12  int d[10010];
13  int size[10010];
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() {
35    scanf("%d%d",&n,&m);
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,sizeof(visit));
43    for (int i=1;i<=n;i++)
44      if (!visit[i])
45        dfs(i);
46    memset(visit,0,sizeof(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++) {
52      int f=fa[i];
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

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

4.2 Aho-Corasick Automation

```
1  #include <cstdio>
2  #include <cstring>
3  #include <queue>
4
5  using std::queue;
6
7  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];

```

```
66  trie* go(trie *now,char ch) {
67    ch -= '0';
68    trie *ans = now;
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;
78      scanf("%d",&n);
79      if (n==0) break;
80      root=head=pool;
81      memset(dp,0x7f,sizeof(dp));
82      memset(pool,0,sizeof(pool));
83
84      static char buf[30];
85      for (int i=0;i<n;i++) {
86        scanf("%s",buf);
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[j][len]<=len) {
101           for (char ch='0';ch<='3';ch++) {
102             trie *tmp=go(pool+j,ch);
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     }
111     int ans=2000000000;
112     for (int j=0;j<n;j++)
113       if (dp[j][len]<ans) ans=dp[j][len];
114     if (ans>len) ans=-1;
115     printf("Case %d: %d\n",ii++,ans);
116   }
117   return 0;
118 }
```

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];
}

```

```
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++)
46             scanf("%d",a+i);
47         ans = 0;
48         sort(a,n);
49         printf("%lld\n",ans);
50     }
51 }
```

5.3 BigInt Multiply with FFT

```
1  #include <cstdio>
2  #include <cstring>
3  #include <cmath>
4
5  typedef long long Long;
6  const int MAXN=32768;
7  const double pi=acos(-1.0);
8  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];
17 char a[MAXN];
18 char b[MAXN];
19 int slena;
20 int slenb;
21 int lena;
22 int lenb;
23 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
38 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          {
51                  int m=1<<s;
52                  double rwm=cos(delta/m);
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];
63                                  // u=A[k+j];
64                                  double ru=R[k+j];
65                                  double iu=I[k+j];
66
67                                  // A[k+j]=u+t;
68                                  R[k+j]=ru+rt;
69                                  I[k+j]=iu+it;
70
71                                  //A[k+j+m/2]=u-t;
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          }
81          for (int i=0;i<size;i++)
82          {
83                  ir[i]=R[i];
84                  ii[i]=I[i];
85          }
86  }
87
88  double POW
        [10]={1,10,100,1000,10000,100000,1000000,10000000,100000000,1000000000};
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);
107                 memset(rb,0,n<<3);
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[j]-'0')*POW[cnt++];
116                         if (cnt==TEN) {lena++; cnt=0;}
117                 }
118                 if (ra[lena]>0.1)       lena++;
119
120                 slenb=next(b);
121                 cnt=0; lenb=0;
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; logn=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                 }
140                 fft(rc,ic,n,-1);
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>=0;i--)
150                 {
151                         if (!print && (ans[i]>0 || i==0))
152                         {
153                                 print=1;
154                                 printf("%lld",ans[i]);
```

```
155                         } else
156                         if (print)
157                                 printf("%05lld",ans[i]);
158                 }
159                 putchar(10);
160         }
161         return 0;
162 }
```