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

```
1 import java.util.*;
2 import java.io.*;
3
4 class Main {
5
6     void run() {
7         //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;
```

```
4
5  const int MAXN = 100;
6  const int EXP = 10;
7  const int INF = 1000000000;
8
9  int nn;
10 int map[MAXN+EXP][MAXN+EXP];
11
12 int sum;
13 bool inSet[MAXN+EXP];
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 <= nn; j++)
23       if(!inSet[j] && dist[j] < min)
24         min = dist[j], idx = j;
25     inSet[idx] = 1;
26     sum += min;
27     for(int j = 1; j <= nn; j++)
28       if(!inSet[j] && dist[j] > map[idx][j])
29         dist[j] = map[idx][j];
30   }
31 }
32
33 int main(){
34   while(scanf("%d\n",&nn) == 1 && nn){
35     for(int i = 1; i <= nn; i++)
36       for(int j = 1; j <= nn; j++)
37         scanf("%d",&map[i][j]);
38     Prim();
39     printf("%d\n",sum);
40   }
41   return 0;
42 }
```

3.2 Prim- O(MlogN)

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

```
#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];
```

```
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
```

```
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){
```

```
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) {
10     for (int i = 1; i <= n; i++) {
11         if (visit[i] == false && adj[now][i]) {
12             visit[i] = true;
13             int tt = match[i];
14             match[i] = now;
15             if (tt == -1 || 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          adj[f][t] = true;
28      }
29      int ans = 0;
30      memset(match,0xff,sizeof(match));
31      for (int i = 1; i <= n; i++) {
32          memset(visit,0,sizeof(visit));
33          if (dfs(i)) ans ++;
34      }
35      printf("%d\n",ans);
36      return 0;
37  }
```

3.11 Minimun Cost Flow [TO BE TESTED!]

```
1   #include <iostream>
2   #include <queue>
3   #include <cstring>
4
5   using namespace std;
6
7   int n,m,ans,t,f;
8   int maxf[210][210],flow[210][210],cost[210][210];
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() {
14    int a[210][2]={0},b[210][2]={0},s=0,sa=0,sb=0;
15    memset(maxf,0,sizeof(maxf));
16    memset(flow,0,sizeof(flow));
17    memset(cost,0,sizeof(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 <= 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;
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 {
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;
  }
```

```
59 } km;
```

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 }
```

5 Data Struct

5.1 Binary Indexed Tree

BECAREFUL WHILE I == 0 !!!

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