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);
8          MyReader in = new MyReader();
9          String str;
10     }
11
```

```
12      public static void main(String args[]) {
13          new Main().run();
14      }
15
16      void debug(Object...x) {
17          System.out.println(Arrays.deepToString(x));
18      }
19  }
20
21  class MyReader {
22      BufferedReader br = new BufferedReader (
23              new InputStreamReader (System.in));
24      StringTokenizer in;
25      String next() {
26          try {
27              while (in == null || !in.hasMoreTokens()) {
28                  // Read a new line and split it into tokens
29                  in = new StringTokenizer(br.readLine());
30              }
31              // return next token
32              return in.nextToken();
33          } catch (Exception e) {
34              // EOF
35              return null;
36          }
37      }
38      // Transform the tokens into other types
39      int nextInt() {
40          return Integer.parseInt(next());
41      }
42  }
```

2.2 Java Multithread

BECAREFUL: CALL START FOR EACH THREAD!

```
1   class Test extends Thread {
2       public static int ans;
3       public static int end;
4       public void run() {
5           int now = 0;
6           for (int i = 0; i < 400000000; i++) {
7               now = (now + i) % 9999997;
8           }
9           System.out.println(now);
10          new SubTask(0,0,100000000).start();
11          new SubTask(1,100000000,200000000).start();
12          new SubTask(2,200000000,300000000).start();
13          new SubTask(3,300000000,400000000).start();
14          for (;;) {
15              try {
16                  sleep(200);
17              } catch (Exception e) {}
18              if (end == 4) break;
19          }
20          System.out.println(ans);
21      }
22      public static void main(String[] args) {
```

```
23          new Test().start();
24      }
25  }
26
27  class SubTask extends Thread {
28      private int pos;
29      private int left;
30      private int right;
31      final static int mod = 9999997;
32
33      // init the input data
34      SubTask(int pos,int left,int right) {
35          this.pos = pos;
36          this.left = left;
37          this.right = right;
38      }
39
40      public void run() {
41          // solve the problem
42          int ans = 0;
43          for (int i = left; i < right; i++) {
44              ans = (ans + i) % mod;
45          }
46          // write the answer back
47          synchronized (this) {
48              Test.ans += ans;
49              Test.ans %= mod;
50              Test.end ++;
51          }
52      }
53  }
```

2.3 Binary Search

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

```
1  while (l + 1 < r) {
2    int mid = (l + r) >> 1;
3    if (check(mid)) l = mid;
4    else r = mid;
5  }
6  return mid;
```

3 Graph Theroy

3.1 Prim - O(N^2)

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

```
    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];
```

```
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 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 }
```

3.7 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;
```

```
}
```

3.8 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;
```

```
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;
```

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 }
```