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

```java
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 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;
}
```

3.7 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;
}
```

3.8 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;
      }
    }
    int sum = 0;
    for(int v = 1; v <= M; v++) sum += W[match[v]][v];
    return -sum;
  }
} km;
```

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;
}
```

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;
}
```