Stare śmieci

Zadania z KI:

- Zakłady
- Miasta partnerskie
- Monety Jaślandii kontratakują
- Rugbyści
- XORy i mnożenie

Silnie spójne składowe, mosty, dwuspójne składowe

Oto bardzo „zoptymalizowany" kod na silnie spójne składowe:

```
template<class V, class E> struct Graf {
  struct Edge : E {int v; Edge(E p, int w) : E(p), v(w) {}};
  struct Vertex : V,vector<Edge> {
  };
  vector<Vertex> g;
  Graf(int n=0) : g(n) {}
  void AddEdge(int p, int k, E d) {
    Edge v(d,k);
    g[p].PB(v);
  }
  void AddEdge(int p, int k) {E d; AddEdge(p,k,d);}
  vector<int> vis;
  Graf<V,E> *nowy;
  void SccDfs(int v,int numer, bool sortuj){
    g[v].s=1; if (!sortuj) vis[v]=numer;
    FOREACH(it,g[v]) if (!g[it->v].s) SccDfs(it->v,numer,sortuj);
    else if (!sortuj && numer > vis[it->v])
    nowy->AddEdge(g[it->v].t,(vis[it->v]=numer));
    if (sortuj) vis.PB(v); else g[v].t=numer;
  }
  /*dla kazdego wierzcholka obliczamy numer silnie spojej skladowej,*/
  /*do ktorej on nalezy, numer ten jest w polu t,*/
  /*zwracamy graf prosty SCC, posortowany topologicznie*/
  Graf<V,E> Scc(){
    Graf<V,E> gt(SIZE(g)),res(SIZE(g)),*tab[]={this, & gt};
    gt.nowy=&res; gt.vis.resize(SIZE(g)); vis.clear();
    REP(i,SIZE(g)) gt.vis[i]=-1;
    REP(i,SIZE(g)) FOREACH(it,g[i]) gt.AddEdge(it->v,i);
    REP(i,2){
      FOREACH(it,tab[i]->g) it->s = 0;
```

```
      int l_sklad=0,v;
      FORD(j,SIZE(g)-1,0)
      if (!tab[i]->g[v=(i?vis[j]:j)].s) tab[i]->SccDfs(v,l_sklad++,1-i);
      if (i) res.g.resize(l_sklad);
    }
    REP(i,SIZE(g)) g[i].t=gt.g[i].t;
    return res;
  }
};
```

Oto mosty:

```
typedef vector<PII> VPII;
template<class V, class E> struct Graf {
  struct Edge : E {int v; Edge(E p, int w) : E(p), v(w) {}};
  struct Vertex : V,vector<Edge> {
  };
  vector<Vertex> g;
  Graf(int n=0) : g(n) {}
  void AddEdge(int p, int k, E d) {
    Edge v(d,k);
    g[p].PB(v);
  }
  void AddEdge(int p, int k) {E d; AddEdge(p,k,d);}
  int id,l;
  VPII *X;
  /*procedura zapisuje w argumencie wszystkie mosty                      */
  /*kazdy most jest zapisany dokladnie raz, jako para (a,b) gdzie a<b    */
  /*zakladam, ze graf jest prosty, nie ma petli i wielokrotnych krawedzi */
  /*zakladam, ze kazda krawedz jest na 2 listach sasiedztwa, czyli ze    */
  /*zostala dodana w obie strony                                         */
  void Bridges(VPII &res){
    res.clear();
    X=&res;
    id=l=0;
    FOREACH(it,g) it->t=-1;
    REP(i,SIZE(g)) if (g[i].t==-1) BriSearch(i,-1);
  }
  void BriSearch(int v,int u){
    g[v].t=g[v].low=l++;
    FOREACH(it,g[v]){
      int w=it->v;
      if (g[w].t==-1){
        BriSearch(w,v);
        if (g[w].low>g[v].t){
          /*bridge  v -- w*/
          X->PB(MP(min(v,w),max(v,w)));
```

```
        } else g[v].low<?=g[w].low;
      } else if (w!=u) g[v].low<?=g[w].t;
    }
  }
};
```

Dwuspójne składowe i mosty:

```
template<class V, class E> struct Graf {
  struct Edge : E {int v; Edge(E p, int w) : E(p), v(w) {}};
  struct Vertex : V,vector<Edge> {
  };
  vector<Vertex> g;
  Graf(int n=0) : g(n) {}
  void AddEdgeU(int p, int k, E d) {
    Edge e(d,k); e.rev=SIZE(g[k])+(p==k); g[p].PB(e);
    e.v=p; e.rev=SIZE(g[p])-1; g[k].PB(e);
  }
  void AddEdgeU(int p, int k) {E d; AddEdgeU(p,k,d);}
  int id,l;
  typedef typename vector<Edge>::iterator EIT;
  vector<EIT> stos;
  #define DccZaznacz(bri) e->dcc=g[e->v][e->rev].dcc=id,\
                          e->bridge=g[e->v][e->rev].bridge=bri
  /*procedura ta powoduje policzenie dla kazdej krawedzi czy jest             */
  /*mostem (pole bridge) oraz numeru dwuspojnej skladowej (pole dcc), a takze */
  /*funkcji low dla kazdego wierzcholka oraz numeru przeszukiwania (pole t)   */
  /*UWAGA: zakladamy, ze graf jest prosty, czyli nie ma wielokrotnych krawedzi*/
  /*oraz krawedzi postaci (v,v), krawedzie dodajemy funkcja AddEdgeU()        */
  void Dcc(){
    id=l=0; stos.clear();
    FOREACH(it,g) it->t=-1;
    REP(i,SIZE(g)) if (g[i].t==-1) DccSearch(i,-1);
  }
  void DccSearch(int v,int u){
    EIT e;
    g[v].t=g[v].low=l++;
    FOREACH(it,g[v]){
      int w=it->v;
      if (g[w].t==-1){
        stos.PB(it);
        DccSearch(w,v);
        if (g[w].low>=g[v].t){
          int ile=0;
          do{
            e=stos.back();
            DccZaznacz(0);
```

```
            stos.pop_back(); ile++;
          } while (e!=it);
          if (ile==1) DccZaznacz(1);
          id++;
        } else g[v].low<?=g[w].low;
      } else if (g[w].t<g[v].t && w!=u) stos.PB(it),g[v].low<?=g[w].t;
    }
  }
};
```

Kilka zadań:

1. Dany jest graf skierowany. Ile minimalnie krawędzi trzeba dodać, by był silnie spójny?

2. Dany jest spójny graf nieskierowany z wyróżnionym wierzchołkiem u. Dla każdego innego wierzchołka v powiedzieć, na ile sposobów można usunąć z tego grafu jedną krawędź, by wierzchołki u i v były w różnych spójnych składowych.

3. To samo, ale na ile sposobów można usunąć jedną krawędź.

4. Znaleźć wierzchołek, którego usunięcie spowoduje rozpad na największą możliwą liczbę spójnych składowych.

5. Dana jest mapa miasta 500×500, każde pole jest kontrolowane przez jedną z mafii. Pola kontrolowane przez jedną mafię są spójne (pola sąsiadują jeśli mają wspólny bok). Mafia jest duża, jeśli ma przynajmniej K pól. Pole w mieście jest niebezpieczne, jeśli należy do dużej mafii lub istnieje taka duża mafia, że nie da się opuścić miasta z tego pola nie przechodząc przez jakieś pole tej mafii. Ile jest niebezpiecznych pól w tym mieście?

6. Są rycerze, niektórzy się lubią, niektórzy nie, jest co najwyżej 1 000 rycerzy. Poprawnym usadzeniem rycerzy (niekoniecznie wszystkich) przy okrągłym stole nazywamy takie usadzenie przynajmniej 3 rycerzy, że każdy siedzi obok dwójki rycerzy, które zna, oraz przy stole siedzi nieparzyście wielu rycerzy. Wyznacz liczbę rycerzy, którzy nigdy nie usiądą przy okrągłym stole, tj. nie istnieje takie usadzenie zawierające ich.