acm.mipt.ru

олимпиады по программированию на Физтехе

Поиск максимального потока в сети, алгортим "поднять и в начало": C++

```
/*
 * An "honest" lift-to-front implementation. Done just as it's said
 * Daniel Shved, MIPT, 2009.
 */
#include <vector>
#include <list>
#include <algorithm>
using namespace std;

typedef vector<int> VInt;
typedef vector<VInt> VVInt;

typedef list<int> LInt;
typedef vector<LInt> VLInt;
typedef LInt::iterator LIter;
typedef vector<LIter> VLIter;

// Input: the network
int n, src, dest;
VVInt c;

// Output: the flow (preflow while the algo is running)
VVInt f;

// Additional data
VInt h, e;
VLInt nei;
VLIter current;

// Lift the vertex. Assumes that this is possible.
void lift(int u)
{
   int height = 2*n;
   for(LIter it = nei[u].begin(); it != nei[u].end(); it++)
      if(c[u][*it] > f[u][*it])
         height = min(height, h[*it]);
   h[u] = height + 1;
}

// Push the flow along the given edge. Assumes that this is possible.
void push(int u, int v)
{
   int value = min(c[u][v] - f[u][v], e[u]);
   f[u][v] += value;
   f[v][u] = -f[u][v];
   e[u] -= value;
   e[v] += value;
}

// Discharge the given vertex. Returns true if lifting occurred.
bool discharge(int u)
{
   bool lifted = false;
   while(e[u] > 0) {
      if(current[u] == nei[u].end()) {
         lift(u);
```

Поиск
Поиск
Раздел «Алгоритмы»
Главная
Форум
Ссылки
El Judge
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

```
            lifted = true;
            current[u] = nei[u].begin();
            continue;
         }
         int v = *current[u];
         if(c[u][v] > f[u][v] && h[u] == h[v]+1)
            push(u, v);
         else
            current[u]++;
      }
      return lifted;
}

// The actual lift-to-front algo (with initialization)
// Returns the max flow value
int ltf()
{
      int u, v;

      // Build the neighbours lists
      nei.resize(n);
      current.resize(n);
      for(u=0; u<n; u++) {
         for(v=0; v<n; v++)
            if(c[u][v] > 0 || c[v][u] > 0)
               nei[u].push_back(v);
         current[u] = nei[u].begin();
      }

      // Initialize the preflow
      f.assign(n, VInt(n, 0));
      e.assign(n, 0);
      h.assign(n, 0);
      h[src] = n;
      for(u=0; u<n; u++)
         if(c[src][u] > 0) {
            e[u] = c[src][u];
            f[src][u] = c[src][u];
            f[u][src] = -f[src][u];
         }

      // lift-to-front
      LInt theList;
      for(u=0; u<n; u++)
         if(u != src && u != dest)
            theList.push_back(u);
      LIter cur = theList.begin();
      while(cur != theList.end()) {
         u = *cur;
         if(discharge(u)) {
            theList.erase(cur);
            cur = theList.insert(theList.begin(), u);
         }
         cur++;
      }
      return e[dest];
}

// A small demo
int main()
{
      int i, j, edgeCount;

      // read the input
      scanf("%d%d%d%d", &n, &edgeCount, &src, &dest);
      c.assign(n, VInt(n, 0));
      while(edgeCount--) {
         int val;
         scanf("%d%d%d", &i, &j, &val);
```

```
      c[i][j] = val;
   }

   // find the max. flow
   int result = ltf();

   // print the results
   printf("max flow value: %d\n", result);
   for(i=0; i<n; i++)
      for(j=0; j<n; j++)
         if(f[i][j] > 0)
            printf("%d -> %d : %d\n", i, j, f[i][j]);
   return 0;
}
```

-- DanielShved - 06 Apr 2009