****



**程序设计综合实践课程报告**

**图论实验**

**学生姓名**

**学院名称 智能与计算学部**

**专 业 工科试验班(智能与计算类)**

**学 号**

# 1. 六度分离

## 1.1题目分析

**可以转换为最短路问题。每两个认识的人间距离都为1，这样的在任意两个人之间求一个最短距离，然后找出这些最短距离的最大值，如果这个值大于7证明这两个人之间隔了多于六个人了。**

## 1.2 题目代码（带注释）

|  |
| --- |
| #include <iostream>  #include <cstring>  #define inf 0x3f3f3f  using namespace std;  int mp[105][105];  void floyd(int n)  { //核心代码      for (int k = 0; k < n; k++) //“中介在外面 ”      for (int i = 0; i < n; i++)          for (int j = 0; j < n; j++)          if (mp[i][j] > mp[i][k] + mp[k][j])              mp[i][j] = mp[i][k] + mp[k][j];  }  int main()  {      int n, m, p1, p2;      while (~scanf("%d%d", &n, &m)) {          for (int i = 0; i < n; i++)          { //初始化：等腰直角三角形              mp[i][i] = 0;              for (int j = 0; j < i; j++)                  mp[i][j] = mp[j][i] = inf;          }          for (int i = 0; i < m; i++)          { //读入数据更新              scanf("%d%d", &p1, &p2);              mp[p1][p2] = mp[p2][p1] = 1;          }          floyd(n);          int flag = 0;          for (int i = 0; i < n; i++)          { //任意两人之间不超过6人最大6人，此时dis=7              for (int j = 0; j < n; j++)              {                  if (mp[i][j] > 7)                  {                      flag = 1;                      j = n, i = n;                  }              }          }          if (flag) cout << "No\n";          else cout << "Yes\n";      }      return 0;  } |
|  |

# 2. 确定比赛名次

## 2.1题目分析

**注意符合条件的排名可能不是唯一的，此时要求输出时编号小的队伍在前；输入数据保证是正确的，即输入数据确保一定能有一个符合要求的排名。**

## 2.2 题目代码（带注释）

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <algorithm>  using namespace std;  int d[550], book[550][550];  int main()  {      int n, m;      while (~scanf("%d%d", &n, &m)) {          int a, b, k;          memset(d, 0, sizeof d);          memset(book, 0, sizeof book);          for (int i = 0; i < m; i++) {              scanf("%d%d", &a, &b);              if (!book[a][b]) {                  d[b]++;                  book[a][b] = 1;              }          }          for (int i = 1; i <= n; i++) {              for (int j = 1; j <= n; j++) {                  if (d[j] == 0) {                      if (i == n) {                          printf("%d\n", j);                      }                      else                      {                          printf("%d ", j);                      }                      d[j]--;                      k = j;                      break;                  }              }              for (int j = 1; j <= n; j++) {                  if (book[k][j]) {                      d[j]--;                  }              }          }      }      return 0;  } |
|  |

# 3. Agri-Net

## 3.1题目分析

**求最小生成树，用prim算法。从顶点的角度为出发点。**

## 3.2 题目代码（带注释）

|  |
| --- |
| #include <stdio.h>  int m, n, sum, e[110][110], book[110], dis[110];  int inf = 99999999;  void Prim()  {      int i, j, k, min;      for (i = 1; i <= n; i++)      {          dis[i] = e[1][i];          book[i] = 0;      }      dis[1] = 0;      book[1] = 1;      for (i = 1; i < n; i++)      {          min = inf;          for (j = 1; j <= n; j++)          if (book[j] == 0 && dis[j] < min)          {              min = dis[j];              k = j;          }          sum += min;          book[k] = 1;          for (j = 1; j <= n; j++)          if (book[j] == 0 && dis[j] > e[k][j])          dis[j] = e[k][j];      }  }  int main()  {      int i, j;      while (scanf("%d", &n) != EOF)      {          sum = 0;          for (i = 1; i <= n; i++)          for (j = 1; j <= n; j++)              scanf("%d", &e[i][j]);              Prim();              printf("%d\n", sum);      }      return 0;  } |
|  |

# 4. Bad Cowtractors

## 4.1题目分析

**求最大生成树，与上面的求最少生成树差不多。**

## 4.2 题目代码（带注释）

|  |
| --- |
| #include <iostream>  using namespace std;    int e[1001][1001], dis[1001], vis[1001], sum, N, M;  void Prime()  {      int max, k;      for (int i = 1; i <= N; i++)      {          dis[i] = e[1][i];          vis[i] = 0;      }      dis[1] = 0;      vis[1] = 1;      for (int i = 1; i < N; i++)      {          k = -1;          max = 0;          for (int j = 1; j <= N; j++)          {              if (vis[j] == 0 && dis[j] > max)              max = dis[k = j];          }          if (k == -1)          {              cout << "-1" << endl;              return;          }          vis[k] = 1;          sum += max;          for (int j = 1; j <= N; j++)          {              if (vis[j] == 0 && dis[j] < e[k][j])              dis[j] = e[k][j];          }      }      cout << sum << endl;  }  int main()  {      int a, b, c;      cin >> N >> M;      for (int i = 1; i <= N; i++)          for (int j = 1; j <= N; j++)              e[i][j] = 0;          for (int i = 0; i < M; i++)          {              cin >> a >> b >> c;              if (c > e[a][b])                  e[a][b] = e[b][a] = c;          }      Prime();      return 0;  } |
|  |

# 5. Till the Cows Come Home

## 5.1题目分析

**就是要问从某个点到一个固定的点的最短距离。由题可知这是一个单个源点到其他顶点的最小距离问题，由此可用Dijkstra's算法求最短路径。**

## 5.2 题目代码（带注释）

|  |
| --- |
| #include <stdio.h>  #define Max 0x3fffffff  int map[1005][1005];  int dis[1005];  void dijkstra(int n)  {      int visit[1001] = {0};      int min, i, j, k;      visit[1] = 1;      for (i = 1; i < n; ++i)      {          min = Max;          k = 1;          for (j = 1; j <= n; ++j)          {              if (!visit[j] && min > dis[j])              {                  min = dis[j];                  k = j;              }          }          visit[k] = 1;          for (j = 1; j <= n; ++j)          {              if (!visit[j] && dis[j] > dis[k] + map[k][j])              dis[j] = dis[k] + map[k][j];          }      }      printf("%d\n", dis[n]);  }  int main()  {      int t, n, i, j, from, to, cost;      while (scanf("%d%d", &t, &n) != EOF)      {          for (i = 1; i <= n; ++i)          {              map[i][i] = 0;              for (j = 1; j < i; ++j)                  map[i][j] = map[j][i] = Max;          }          for (i = 1; i <= t; ++i)          {              scanf("%d%d%d", &from, &to, &cost);              if (cost < map[from][to]) //可能有多条路，只记录最短的                  map[from][to] = map[to][from] = cost;          }          for (i = 1; i <= n; ++i)              dis[i] = map[1][i];          dijkstra(n);      }  } |
|  |