USACO Traingate 'fence' Analysis

by Hal Burch

Assuming you pick the lowest index vertex connected to each node, the Eulerian Path algorithm actually determines the path requested, although in the reverse direction. You must start the path determination at the lowest legal vertex for this to work.

```
#include <stdio.h>
#include <string.h>

#define MAXI 500
#define MAXF 1200

char conn[MAXI][MAXI];
int deg[MAXI],nconn,touched[MAXI],path[MAXF],plen;

/* Sanity check routine */
void fill(int loc)
 {
  int lv;

  touched[loc] = 1;
  for (lv = 0; lv < nconn; lv++)
    if (conn[loc][lv] && !touched[lv])
      fill(lv);
 }

/* Sanity check routine */
int is_connected(int st)
 {
  int lv;
  memset(touched, 0, sizeof(touched));
  fill(st);
  for (lv = 0; lv < nconn; lv++)
    if (deg[lv] && !touched[lv])
      return 0;
  return 1;
 }

/* this is exactly the Eulerian Path algorithm */
void find_path(int loc)
 {
  int lv;

  for (lv = 0; lv < nconn; lv++)
    if (conn[loc][lv])
     {
      /* delete edge */
      conn[loc][lv]--;
      conn[lv][loc]--;
      deg[lv]--;
      deg[loc]--;
```

```
      /* find path from new location */
      find_path(lv);
     }

  /* add this node to the `end' of the path */
  path[plen++] = loc;
 }

int main(int argc, char **argv)
 {
  FILE *fin = fopen("fence.in", "r"), *fout = fopen("fence.out", "w");
  int nfen, lv, x, y;

  fscanf (fin, "%d", &nfen);
  for (lv = 0; lv < nfen; lv++)
   {
    fscanf (fin, "%d %d", &x, &y);
    x--; y--;
    conn[x][y]++;
    conn[y][x]++;
    deg[x]++;
    deg[y]++;
    if (x >= nconn) nconn = x+1;
    if (y >= nconn) nconn = y+1;
   }

  /* find first node of odd degree */
  for (lv = 0; lv < nconn; lv++)
    if (deg[lv] % 2 == 1) break;
  /* if no odd-degree node, find first node with non-zero degree */
  if (lv >= nconn)
    for (lv = 0; lv < nconn; lv++)
      if (deg[lv]) break;
#ifdef CHECKSANE
  if (!is_connected(lv)) /* input sanity check */
   {
    fprintf (stderr, "Not connected?!?\n");
    return 0;
   }
#endif

  /* find the eulerian path */
  find_path(lv);

  /* the path is discovered in reverse order */
  for (lv = plen-1; lv >= 0; lv--)
    fprintf (fout, "%i\n", path[lv]+1);
}
```