USACO MAR07 Problem 'ranking' Analysis

by Richard Peng

To simplify notation, we use X->Y to denote the existence of a path of queries leading from a to b, from which we could deduce cow X is ranked higher than cow Y.

An obvious upper bound for the number of queries to ask is the number of pairs of cows for which we have neither of X->Y and Y->X. It's slightly less obvious that this is also the lower bound.

Suppose there exist a set of queries in which the relationship between X and Y is not queried and from the original data we could not deduce X->Y or Y->X. Then we could answer the queries in a way so X and Y's ranking end up being adjacent to each other (by making all other cows rank either lower or higher than both of X and Y). In such a case, it's clear a comparison between X and Y is necessary as otherwise, we could switch the position of X and Y without altering the results of the queries.

The code for this problem is straight forward DFS to check which cows can be 'reached' by a set of information already given for each cow. Each round of DFS runs in O(M) time, giving a runtime of O(MN)

Here is Brian Dean's code:

```
#include <stdio.h>
#define MAX_N 1000
#define MAX_M 10000

typedef struct
{
  int dest;
  int next_edge;
} Edge;

char TC[MAX_N][MAX_N];
char beenthere[MAX_N];
int N, M, first_edge[MAX_N];
Edge edges[MAX_M];

void DFS(int s, int i)
{
  int e;
  TC[s][i] = 1;
  beenthere[i] = 1;
  e = first_edge[i];
  while (e != -1) {
    if (!beenthere[edges[e].dest])
      DFS(s, edges[e].dest);
    e = edges[e].next_edge;
  }
}
```

```
int main(void)
{
  int i, j, k, m;
  FILE *fp;

  fp = fopen ("ranking.in", "r");
  fscanf (fp, "%d %d", &N, &M);
  for (i=0; i<N; i++) first_edge[i] = -1;
  m = 0;
  for (k=0; k<M; k++) {
    fscanf (fp, "%d %d", &i, &j);
    TC[i-1][j-1] = 1;
    edges[m].dest = j-1;
    edges[m].next_edge = first_edge[i-1];
    first_edge[i-1] = m++;
  }
  fclose (fp);

  for (i=0; i<N; i++) {
    for (j=0; j<N; j++) beenthere[j] = 0;
    DFS(i, i);
  }

  k = 0;
  for (i=0; i<N; i++)
    for (j=i+1; j<N; j++)
      if (TC[i][j]==0 && TC[j][i]==0) k++;

  fp = fopen ("ranking.out", "w");
  fprintf (fp, "%d\n", k);
  fclose (fp);
  return 0;
}
```