USACO OPEN11 Problem 'space3d' Analysis

by Lewin Gan

This is a classic flood-fill problem, but in 3-D. In 2-D, the process looks something like the pseudo-code below:

```
dx = {-1,0,1,0}
dy = {0,-1,0,1}
flood(x,y) {
   if already visited (x,y) return;
   mark (x,y) as visited
   for each direction
     flood(x+dx,y+dy);
}
```

Then, to count asteroids, we loop over each point and flood it if it has not been visited and it contains a part of an asteroid. This takes O(N^2) in 2-D.

In 3-D, however, we have three coordinates (x,y,z). We then have three arrays dx, dy, dz to represent our directions. Our array will be filled with values where two of dx,dy,dz are zero, and the other is positive or negative one, to give us a total of six directions. Then, we can proceed in a way similar to the 2-D case to give us a solution in O(N^3) time

Notice that a DFS using the system's implicit stack may time-out given the large number of states, so using an explicit stack or a BFS is safer.

A concise solution by Andre below:

```
#include <cstdio>
#include <cstdlib>
#include <queue>
using namespace std;

FILE *fin = fopen ("space3d.in", "r"), *fout = fopen ("space3d.out", "w");

const int MAXN = 105,
          dx [] = {0, 1, 0, -1, 0, 0},
          dy [] = {1, 0, -1, 0, 0, 0},
          dz [] = {0, 0, 0, 0, 1, -1};

int N, rock = 0;
bool vis [MAXN][MAXN][MAXN];
char field [MAXN][MAXN][MAXN];

struct triple { int x, y, z; };

void flood (int x, int y, int z)
{
    queue <triple> q; q.push((triple) {x, y, z});
    while (!q.empty ())
    {
        triple top = q.front (); q.pop (); int x = top.x, y = top.y, z =
top.z;

        if (vis [x][y][z])
            continue;

        vis [x][y][z] = true;

        for (int d = 0; d < 6; d++)
        {
            int a = x + dx [d], b = y + dy [d], c = z + dz [d];

            if (field [a][b][c] == field [x][y][z] && !vis [a][b][c])
                q.push ((triple) {a, b, c});
        }
    }

    return;
}

int main ()
{
    fscanf (fin, "%d", &N);

    for (int i = 1; i <= N; i++)
        for (int j = 1; j <= N; j++)
            fscanf (fin, "%s", &field [i][j][1]);

    for (int x = 1; x <= N; x++)
        for (int y = 1; y <= N; y++)
            for (int z = 1; z <= N; z++)
                if (!vis [x][y][z] && field [x][y][z] == '*')
                    flood (x, y, z), rock++;

    fprintf (fout, "%d\n", rock);
    return 0;
}
```