COCI 2006/2007 Contest #1

고려대학교 장홍준

A. MODULO

- 1000 이하의 음이 아닌 정수가 주어지기 때문에 [1001]크기의 배열을 잡아서 counting 해주는 방법

- map<int, int>를 사용해서 마지막에 size()를 출력하는 방법

A. MODULO

```
#include <stdio.h>
#include <map>
using namespace std;
map <int, int> d;
int main() {
    for (int i = 1; i <= 10; i++) {
        int x; scanf("%d", &x);
        d[x % 42] = 1;
    }
    printf("%d", d.size());
}
```

B. HERMAN

- 파이(Pi)값은 컴퓨터에서 계산기를 켜서 ‘p’를 누르면 나옵니다.
- 원의 넓이는 Pi*R*R, 택시 기하학에서의 원은 한 변의 길이가 루트(2)R인 정사각형이므로 넓이가 2*R*R

C. OKVIRI

- 5개의 열(세로줄)씩 한 단위로 보면 생각하기 좋다.
- 새로운 한 단위가 추가될 때, 이전 단위의 가장 오른쪽 세로줄 하나를 없애고 삽입하면 된다.
- 새로운 한 단위를 추가할 때, 3k번째 단위이면 # 대신 *.

C. OKVIRI

```c
#include <stdio.h>
int m = 1;
char S[22], A[5][222];
void Do(char *s) {
    m++;
    for (int i = 0; i < 5; i++) {
        if (s[i] == '#' && A[i][m] == '*') continue;
        A[i][m] = s[i];
    }
}
int main() {
    scanf("%s", S + 1);
    for (int n = 1; S[n]; n++) {
        m--;
        if (n % 3 == 0) {
            Do("..*.."); Do(".*.*."); Do("*...*");
            A[2][m] = S[n];
            Do(".*.*."); Do("..*..");
        }
        else {
            Do("..#.."); Do(".#.#."); Do("#...#");
            A[2][m] = S[n];
            Do(".#.#."); Do("..#..");
        }
    }
    for (int i = 0; i < 5; i++) puts(A[i] + 1);
}
```

D. SLIKAR

- BFS를 사용하시거나 전체 그리드(S)에 대해서 1턴 뒤의 새로운 그리드(a)를 구하고 갱신하는 식으로 해도 됨.

- 한 턴에 대해서 고슴도치 먼저 움직이고, 물이 번져나감.

- O((NM)^2)

D. SLIKAR

```c
#include <stdio.h>
int n, m, X, Y, X2, Y2;
char S[55][55], a[55][55];
void copy(int x) {
    if (x) for (int i = 1; i <= n; i++) for (int j = 1; j <= m; j++) a[i][j] = S[i][j];
    else for (int i = 1; i <= n; i++) for (int j = 1; j <= m; j++) S[i][j] = a[i][j];
}
int px[4] = { 0, 0, 1, -1 };
int py[4] = { 1, -1, 0, 0 };
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) {
        scanf("%s", S[i] + 1);
        for (int j = 1; j <= m; j++) {
            if (S[i][j] == 'S') X = i, Y = j;
            if (S[i][j] == 'D') X2 = i, Y2 = j;
        }
    }
    for (int r = 1; r <= n*m; r++) {
        copy(1);
        for (int i = 1; i <= n; i++) for (int j = 1; j <= m; j++) if (S[i][j] == '.' || S[i][j] == 'D') {
            for (int k = 0; k < 4; k++) {
                int x = i + px[k], y = j + py[k];
                if (S[x][y] == 'S') a[i][j] = 'S';
            }
        }
        if (a[X2][Y2] == 'S') {
            printf("%d", r);
            return 0;
        }
        copy(0);
        for (int i = 1; i <= n; i++) for (int j = 1; j <= m; j++) if (S[i][j] == '.' || S[i][j] == 'S') {
            for (int k = 0; k < 4; k++) {
                int x = i + px[k], y = j + py[k];
                if (S[x][y] == '*') a[i][j] = '*';
            }
        }
        copy(0);
    }
    puts("KAKTUS");
}
```

E. BOND

- 0번째 요원부터 N-1번째 요원, 0번 미션부터 N-1번째 미션까지 있다고 가정.
- X(i) : i번째 요원이 수행한 미션
- D(person, status)

:0번째 요원부터 person번째 요원까지, status상황일 때의 확률 곱의 최댓값

D(1, 10) : 0번째 요원과 1번째 요원이 10=2^3+2^1 상황의 선택을 했다.

(1번째 미션과 3번째 미션을 수행했을 때에 가능한 확률 곱의 최댓값)

E. BOND

```cpp
#include <stdio.h>
#include <algorithm>
using namespace std;
const double eps = 1e-9;
int n;
double x[20][20], d[20][1 << 20];
double f(int N, int status) {
    if (N < 0) return 1;
    if (d[N][status] > -eps) return d[N][status];
    double &res = d[N][status];
    res = 0;
    for (int i = 0; i < n; i++) {
        if ((1 << i)&status) {
            double v = f(N - 1, (1 << i) ^ status);
            res = max(res, v * x[N][i]);
        }
    }
    return res;
}
int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; i++) for (int j = 0; j < n; j++) scanf("%lf", &x[i][j]), x[i][j] /= 100;
    for (int i = 0; i < n; i++) for (int j = 0; j < (1 << n); j++) d[i][j] = -1;
    printf("%.10lf", f(n - 1, (1 << n) - 1)*100.0);
}
```

0명이 0개의 작업을 할 확률은 1

//계산했으면 다시 계산하지말자

i번째 작업을 N번 사람이 하고싶은데, 현재 status에 할 수 있다고 표기되어있니?

최댓값으로 갱신^^

모든 사람이 모든 작업을 할당했을 때의 답을 구해죠

E. BOND

- 그런데 방금처럼 짜면 메모리 초과가 납니다.
- 잘 관찰해보면 i번째 사람에 대한 DP 테이블을 채울 때, i-1번째 사람까지 다 계산되어있는 D[i-1][~]만 참조합니다.
- 그래서 i를 0, 1, 2, … , N-1까지 채워나가는데

i가 0, 2, 4, 6, … 일 때에는 D[0][~]에 값을 넣고

I가 1, 3, 5, 7, … 일 때에는 D[1][~]에 값을 넣으면 D[2][2^N]개의 배열만 가지고 답을 구할 수 있습니다. 그런데 처음에 D[0]을 채우고, D[1]을 계산한 다음, D[0]을 다시 초기화해주고 D[1]을 가지고 D[0]을 계산해야하겠습니다.

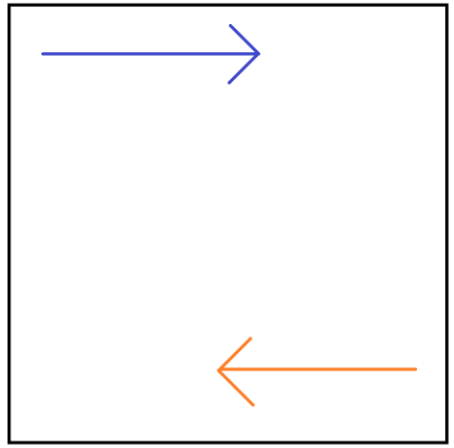
E. BOND

```cpp
#include <stdio.h>
#include <algorithm>
using namespace std;
const double eps = 1e-9;
int n;
double x[20][20], d[2][1 << 20];
int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; i++) for (int j = 0; j < n; j++) scanf("%lf", &x[i][j]), x[i][j] /= 100;
    for (int i = 0; i < n; i++) d[0][1 << i] = x[0][i];

    for (int i = 1; i < n; i++) {
        for (int j = 0; j < (1 << n); j++) d[i & 1][j] = 0;
        for (int j = 0; j < (1 << n); j++) {
            int cnt = 0;
            for (int k = 0; k < n; k++) if (j&(1 << k)) ++cnt;
            if (cnt != i) continue;
            for (int k = 0; k < n; k++) if (j ^ (1 << k)) {
                d[i & 1][j ^ (1 << k)] = max(d[i & 1][j ^ (1 << k)], d[!(i & 1)][j] * x[i][k]);
            }
        }
    }
    printf("%.8lf", d[(n - 1) & 1][(1 << n) - 1] * 100.0);
}
```

F. DEBUG

- 문제에서 요구하는 것이 가장 큰 정사각형 킬러.
- 큰 정사각형부터 크기를 줄어나가면서 킬러가 존재하는지 판별해보자.
- 정사각형의 크기를 결정했다면 푸른색에 대응되는 주황색 칸이 일치하는지 판별.
- 크기가 1보다 큰 정사각형을 찾는 것에 주의



F. DEBUG

```cpp
#include <stdio.h>
#include <algorithm>
using namespace std;
int n, m;
char s[305][305];
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 0; i < n; i++) scanf("%s", s[i]);
    for (int R = min(n, m); R >= 2; R--) {
        for (int i = 0; i < n - R + 1; i++) {
            for (int j = 0; j < m - R + 1; j++) {
                bool ok = 1;
                int k2 = i + R - 1, l2;
                for (int k = i; k < i + R; k++) {
                    l2 = j + R - 1;
                    for (int l = j; l < j + R; l++) {
                        if (s[k][l] != s[k2][l2]) {
                            ok = 0;
                            break;
                        }
                        l2--;
                    }
                    if (!ok) break;
                    k2--;
                }
                if (ok) {
                    printf("%d", R);
                    return 0;
                }
            }
        }
    }
    puts("-1");
}
```