BUG x Com:BE 프로그래밍 경진 대회 풀이

한성대학교 학술소모임 BUG

A. HSU

태그 - 구현

예상 난이도 – **Bronze V**

- 그냥 출력하면 되는 문제입니다.

A. HSU

출제진의 풀이 (Python)

```
print("""##     ##   #######   ##     ##
##     ##  ##     ##  ##     ##
##     ##  ###        ##     ##
#########   #######   ##     ##
##     ##        ###  ##     ##
##     ##  ##     ##  ##     ##
##     ##   #######    #######""")
```

B. 환상의 콤비

태그 – 수학, 조합론, 사칙연산

예상 난이도 – **Bronze II**

- 세 사람이 일직선 상에 있는 경우를 제외한, BUG 인원과 Com:BE 인원 중 3명을 선택하는 경우의 수를 출력하면 됩니다.
- 팩토리얼을 사용해도 되고 (풀이 1 참고), 사칙연산만으로 풀 수 있습니다. (풀이 2 참고)

B. 환상의 콤비

출제진의 풀이 1 (Python)

```python
import sys
from math import factorial
input = sys.stdin.readline


def combi(n, r):
    if n < r:
        return 0
    return factorial(n) // factorial(r) // factorial(n-r)


a, b = map(int, input().split())
result = combi(a + b, 3) - combi(a, 3) - combi(b, 3)
print(result)
```

B. 환상의 콤비

출제진의 풀이 2 (C++)

```cpp
#include <iostream>
using namespace std;

int main(void) {
    cin.tie(0)->sync_with_stdio(0);
    int b, c;
    cin >> b >> c;
    cout << (b * (b - 1) / 2) * c + (c * (c - 1) / 2) * b;
}
```

C. 도미노

태그 – 브루트포스

예상 난이도 – **Silver V**

- 이중 for문으로 모든 도미노를 탐색하면 되는 문제입니다.
- 도미노가 list[i]보다 크거나 같고, list[i] + k 보다 작거나 같은 경우의 수를 세 주고, 그 중 최대값을 출력하면 됩니다.
- 사실 이분 탐색을 사용하면 더 빠르게 풀 수 있습니다. (풀이 2 참고)

C. 도미노

출제진의 풀이 1 (Java)

```java
import java.io.*;
import java.util.*;

public class domino {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));
        StringTokenizer st = new StringTokenizer(br.readLine());

        int n = Integer.parseInt(st.nextToken());
        int k = Integer.parseInt(st.nextToken());
        int[] list = new int[n];
        for (int i = 0; i < n; i++) {
            list[i] = Integer.parseInt(br.readLine());
        }

        int ans = 0;
        for (int i = 0; i < n; i++) {
            int amt = 0;
            for (int j = 0; j < n; j++) {
                if(list[j] >= list[i] && list[j] <= list[i] + k) {
                    amt++;
                }
            }
            if (amt > ans) {
                ans = amt;
            }
        }

        bw.write(String.valueOf(ans));
        bw.flush();
        bw.close();
        br.close();
    }
}
```

C. 도미노

출제진의 풀이 2 (Python)

```python
import sys
input = sys.stdin.readline

n, k = map(int, input().split())
dominos = sorted([int(input()) for _ in range(n)])

result = 0
for i in range(n):
    count = 0
    for j in range(n):
        if dominos[j] >= dominos[i] and dominos[j] <= dominos[i] + k:
            count += 1
    result = max(result, count)

print(result)
```

D. 맥도날드

태그 – 이분 탐색, 매개 변수 탐색

예상 난이도 – **Silver III**

- 빅맥 한 개의 가격이 5,000원 이므로, 목표 판매 수량은 목표 매출 ÷ 5000 입니다.
- 시간을 기준으로 이분 탐색을 돌리면 됩니다.

D. 맥도날드

출제진의 풀이 (Python)

```python
import sys
input = sys.stdin.readline


def search(start, end, target):
    min_time = 0
    while start <= end:
        mid = (start + end) // 2
        if sum(map(lambda x: mid//x, mcdonald)) >= target:
            min_time = mid
            end = mid - 1
        else:
            start = mid + 1
    return min_time


n, m = map(int, input().split())
mcdonald = list(map(int, input().split()))
print(search(0, int(1e12), m//5000))
```

E. 술 게임

태그 – 브루트포스

난이도 – **Silver I**

- 범위가 10^{16}으로 매우 커 보일 수 있지만, 실제로 박수를 치는 경우는 적습니다.
- 숫자가 반복되는 수(예 : 77777777)의 한자리씩 변경한 다음 그 숫자가 X와 Y 사이에 존재하는 수인지 확인하여 문제를 해결할 수 있습니다.
- 10^2 에서 10^{16} 까지 박수를 치는 경우의 수는 모두 17×9 이며, 이는 완전 탐색을 통하여 구하기에 충분히 작은 수입니다.

E. 술 게임

출제진의 풀이 (C++)

```cpp
#include <cassert>
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <vector>
using namespace std;

int main(void) {
    cin.tie(0)->sync_with_stdio(0);
    long long x, y;
    cin >> x >> y;

    int result = 0;
    for (int sz = 3; sz <= 17; sz++) {
        for (int d0 = 0; d0 < 10; d0++) {
            string S(sz, '0' + d0);
            for (int d1 = 0; d1 < 10; d1++) {
                if (d0 == d1)
                    continue;
                for (int i = 0; i < sz; i++) {
                    S[i] = '0' + d1;
                    long long num = atoll(S.c_str());
                    if (S[0] != '0' && x <= num && num <= y) {
                        result++;
                    }
                    S[i] = '0' + d0;
                }
            }
        }
    }
    cout << result << endl;
}
```

F. 얼음 미로

태그 – 그래프 이론, 그래프 탐색, 너비 우선 탐색

난이도 – **Gold IV**

- 최악의 케이스에서 $O(N * M * X)$ 인 500 * 500 * 1000 의 로직으로 접근하면, 2.5억이므로 시간초과(TLE)가 발생합니다.
- BFS 탐색으로 최단거리를 구하되 얼음이 있는 위치는 우선 이동시키고, 얼음이 녹을 때까지 그 자리에서 기다리는 형태로 구현하면, $O(N * M + X)$ 의 시간 복잡도로 문제를 해결할 수 있습니다.

F. 얼음 미로

출제진의 풀이 (Java)

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayDeque;
import java.util.StringTokenizer;

public class Main {

    private static class Node {
        int r, c;

        public Node(int r, int c) {
            this.r = r;
            this.c = c;
        }
    }

    private static final int[] ry = {-1, 0, 1, 0};
    private static final int[] rx = {0, 1, 0, -1};

    private static int[][] map;
    private static int[][] dist;
    private static int N, M;

    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());

        N = Integer.parseInt(st.nextToken());
        M = Integer.parseInt(st.nextToken());
        map = new int[N][M];
        dist = new int[N][M];

        for (int i = 0; i < N; i++) {
            st = new StringTokenizer(br.readLine());
            for (int j = 0; j < M; j++) {
                map[i][j] = Integer.parseInt(st.nextToken());
            }
        }

        ArrayDeque<Node> queue = new ArrayDeque<>();
        queue.addLast(new Node(0, 0));
        dist[0][0] = 0;
        map[0][0] = -1;

        while (!queue.isEmpty()) {
            Node now = queue.pollFirst();
            if (map[now.r][now.c] >= dist[now.r][now.c]) {
                dist[now.r][now.c]++;
                queue.addLast(now);
                continue;
            }
            for (int i = 0; i < 4; i++) {
                int nr = now.r + ry[i];
                int nc = now.c + rx[i];
                if (nr < 0 || nc < 0 || nr >= N || nc >= M)
                    continue;
                if (dist[nr][nc] > 0)
                    continue;
                dist[nr][nc] = dist[now.r][now.c] + 1;
                queue.addLast(new Node(nr, nc));
            }
        }
        System.out.println(dist[N - 1][M - 1]);
    }
```

G. 미스터리 사인

태그 – 넌센스, 런타임 전의 전처리

난이도 – **Unrated**

- 1번 : 두 수의 곱
- 2번 : 각 수를 제곱한 후, 두 수의 합
- 3번 : HTTP 응답 코드의 합
- 4번 : 이진수 덧셈
- 5번 : 소인수 분해 후 나열

G. 미스터리 사인

출제진의 풀이 (Python)

```python
print(460 + 9220 + 801 + 92 + 341222319)
```