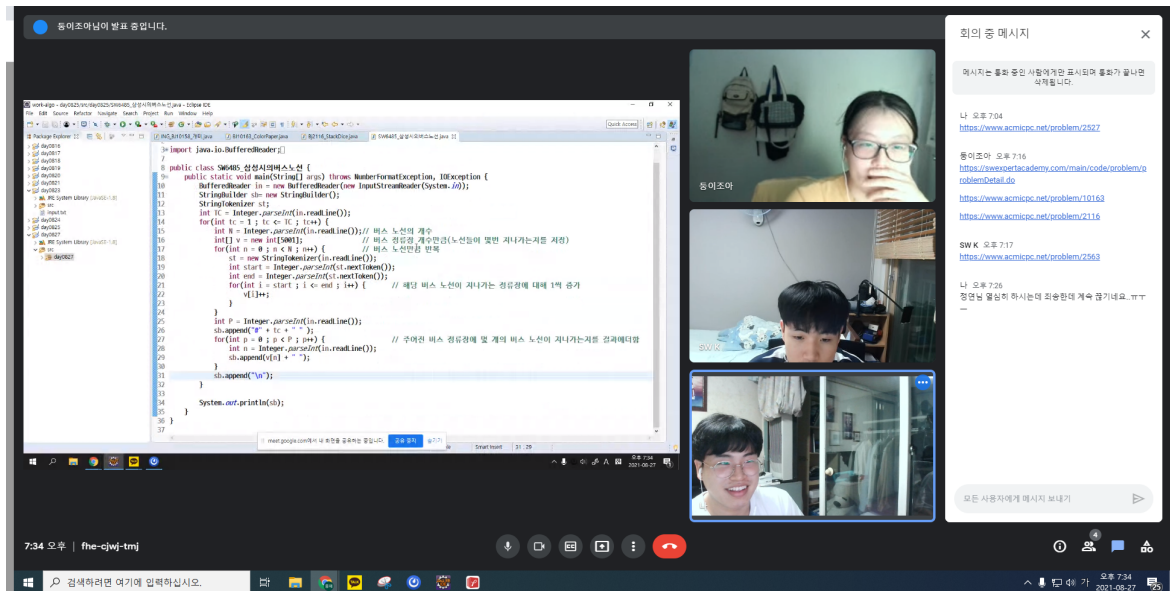


4주차 스터디 - IM 대비 문제 풀이

- 일시 8월 27일 19시
- 멤버 김중재, 김석원, 우정연

▼ 스터디 사진



2798번: 블랙잭

카지노에서 제일 인기 있는 게임 블랙잭의 규칙은 상당히 쉽다. 카드의 합이 21을 넘지 않는 한도 내에서, 카드의 합을 최대한 크게 만드는 게임이다. 블랙잭은 카지노마다 다양한 규정이 있다. 한국 최고의 블랙잭 고수 김정인은 새로운 블랙잭 규칙을 만들어 상근, 창영이와 게임하려고 한다. 김정인 버전의 블랙잭에서 각 카드에는 양의 정수가 쓰여 있

<https://www.acmicpc.net/problem/2798>

BAEKJOON
ONLINE JUDGE

▼ 코드(for문), (comb버전 추가 예정,,)

▼ for문버전

```
package baekjoon_Easy;

import java.util.Scanner;

public class B_2798 {
    public static void main(String[] args) {
        //N개의 번호를 입력받아 3개를 선택해(조합) 그 합이 M과 작거나 같은 수중 최대값을 출력하는 문제.
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int M = sc.nextInt();
        int max = 0;
        int[] cards = new int[N];

        for(int i = 0; i < N; i++) {
            cards[i] = sc.nextInt();
        }

        for (int i = 0; i < N - 2; i++) {
            for (int j = i + 1; j < N - 1; j++) {
                for (int k = j + 1; k < N; k++) {
                    int sum = cards[i] + cards[j] + cards[k];
                    if( max < sum && sum <= M) { // M보다 작는데 최대값이면 max에 대입!!!
                        max = sum;
                    }
                }
            }
        }
    }
}
```

```

    }
    System.out.println(max);
}
}

```

▼ comb버전

```

package baekjoon_Easy;

import java.util.Scanner;

public class B_2798_comb {
    static int N;
    static int M;
    static int max;
    static int[] numbers;
    static int[] cards;
    public static void main(String[] args) {
        //N개의 번호를 입력받아 3개를 선택해(조합) 그 합이 M과 작거나 같은 수중 최대값을 출력하는 문제.
        Scanner sc = new Scanner(System.in);
        //카드갯수 인풋
        N = sc.nextInt();
        //블랙잭의 목표수 원래 21이지만 이 문제에서는 임의의 M
        M = sc.nextInt();
        //max값 초기화
        max = 0;
        //카드 초기화
        cards = new int[N];
        numbers = new int[3];
        //N개의 카드 인풋
        for(int i = 0; i < N; i++) {
            cards[i] = sc.nextInt();
        }
        //end input
        comb(0, 0);
        System.out.println(max);
    }
    /**
     * nC3을 수행하는 함수. numbers배열에 뽑은 수를 저장한다.
     * @param target : 대입할 수
     * @param cnt : 대입할 인덱스
     */
    static void comb(int target, int cnt) {
        if(cnt == 3) {
            calc();
            return;
        }
        if(target == N) return;
        numbers[cnt] = cards[target];
        comb(target+1, cnt+1);
        comb(target+1, cnt);
    }

    /**
     * 완성된 numbers배열을 모두 더해서 M보다 작거나 같을때 최대값을 대입하는 함수
     */
    static void calc() {
        int sum = numbers[0] + numbers[1] + numbers[2];
        if(sum <= M && max < sum) {
            max = sum;
        }
        // System.out.println(max);
    }
}

```

10163번: 색종이

문제 평면에 색깔이 서로 다른 직사각형 모양의 색종이 N장이 하나씩 차례로 놓여진다. 이때 색종이가 비스듬하게 놓이는 경우는 없다. 즉, 모든 색종이의 변은 서로 평행하거나, 서로 수직이거나 둘 중 하나이다. 그림-1은 1번, 2번, 3번 세 장의 색종이가 순서대로 놓인 상태를 보여준다. 그림-1 여기에 그림-2에서 보인 것처럼 4번 색종이가 하나 더 놓이

<https://www.acmicpc.net/problem/10163>

BAEKJOON
ONLINE JUDGE

▼ 10163. 색종이

```

import java.io.BufferedReader;
import java.io.IOException;

```

```

import java.io.InputStreamReader;
import java.util.StringTokenizer;

public class BJ10163_ColorPaper {
    static int N; // 색종이의 개수
    static int SIZE = 1001; // 색종이를 놓을 평면의 크기
    static int[][] map; // 평면
    static int[] size; // 최종적으로 보이는 각 색종이의 넓이
    public static void main(String[] args) throws NumberFormatException, IOException {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st;
        StringBuilder sb = new StringBuilder();
        N = Integer.parseInt(in.readLine());
        map = new int[SIZE][SIZE];
        size = new int[N];
        for(int n = 1 ; n <= N ; n++) {
            st = new StringTokenizer(in.readLine());
            int si = Integer.parseInt(st.nextToken()); // 색종이의 시작 i값
            int sj = Integer.parseInt(st.nextToken()); // 색종이의 시작 j값
            int w = Integer.parseInt(st.nextToken()); // 색종이의 너비
            int h = Integer.parseInt(st.nextToken()); // 색종이의 높이
            for(int i = si ; i < w + si ; i++) {
                for(int j = sj ; j < h + sj ; j++) { // 색종이가 차지하는 면적에 해당 색종이의 번호를 저장해줌(다른 색종이가 나중에 덮는 경우, 그 색종이로 덮어)
                    map[i][j] = n;
                }
            }
        }
        for(int i = 0 ; i < SIZE ; i++) { // 각 색종이가 차지하는 면적을 각각 계산
            for(int j = 0 ; j < SIZE ; j++) {
                if(map[i][j] > 0) size[map[i][j] - 1]++;
            }
        }
        for(int n = 0 ; n < N ; n++) { // 각 색종이의 결과 면적을 StringBuilder에 더함
            sb.append(size[n] + "\n");
        }
        System.out.println(sb);
    }
}

```

2116번: 주사위 쌓기

천수는 여러 종류의 주사위를 가지고 쌓기 놀이를 하고 있다. 주사위의 모양은 모두 크기가 같은 정육면체이며 각 면에는 1부터 6까지의 숫자가 하나씩 적혀있다. 그러나 보통 주사위처럼 마주 보는 면에 적혀진 숫자의 합이 반드시 7이 되는 것은 아니다. 주사위 쌓기 놀이는 아래에서부터 1번 주사위, 2번 주사위, 3번 주사위, ... 의 순서로 쌓는 것이다.

<https://www.acmicpc.net/problem/2116>

BAEKJOON
ONLINE JUDGE

▼ 2116. 주사위 쌓기

```

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

public class Bj2116_StackDice {
    static int N; // 주사위의 수
    static int[][] diceArr; // 주사위 전개도를 저장하는 배열
    static int[] oppositeIdx = {5, 3, 4, 1, 2, 0}; // 주사위에서 서로 반대되는 인덱스를 저장
    static int ans; // 답
    public static void main(String[] args) throws NumberFormatException, IOException {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st;
        N = Integer.parseInt(in.readLine());
        diceArr = new int[N][6];
        for(int i = 0 ; i < N ; i++) { // N개의 주사위 전개도를 저장
            st = new StringTokenizer(in.readLine());
            for(int j = 0 ; j < 6 ; j++) {
                diceArr[i][j] = Integer.parseInt(st.nextToken());
            }
        }
        // 첫 번째 주사위의 윗면만 설정해주면 그 이후는 계산만 해주면 됨
        // 첫 번째 주사위의 윗면이 될 수 있는 경우는 1 ~ 6
        for(int i = 0 ; i < 6 ; i++) {
            int upIdx = i; // 첫 번째 주사위 윗면 설정
            int sum = 0, n = 0; // 이 때의 옆면 최댓값 총 합, 몇 번째 주사위인지
            while(true) { // 무한 반복
                int max = Integer.MIN_VALUE;
                for(int j = 0 ; j < 6 ; j++) { // 해당 주사위에 대해
                    if(j == upIdx || j == oppositeIdx[upIdx]) continue; // 윗면, 아랫면이 아닌 칸 중
                    max = Integer.max(max, diceArr[n][j]); // 최댓값을 저장
                }
                sum += max;
                n++;
            }
            ans = Math.max(ans, sum);
        }
        System.out.println(ans);
    }
}

```

```

    }
    sum += max;
    if(++n >= N) break;        // 모든 주사위에 대해 검사했다면 중단
    for(int j = 0 ; j < 6 ; j++) {    // 윗칸 주사위에 대해
        if(diceArr[n][j] == diceArr[n - 1][upIdx]) {        // 아랫칸 주사위의 윗면과 같은 값이라면 -> 윗칸 주사위의 아랫면임
            upIdx = oppositeIdx[j];        // 아랫면의 반대편에 있는 윗면의 인덱스를 저장
            break;
        }
    }
    }
    ans = Integer.max(ans, sum);    // 더 큰 값을 결과로 저장
}
System.out.println(ans);
}
}
}

```

SW Expert Academy

SW 프로그래밍 역량 강화에 도움이 되는 다양한 학습 콘텐츠를 확인하세요!

https://swexpertacademy.com/main/code/problem/problemDetail.do?contestProbId=AWczm7QaACgDFAWn&categoryId=AWczm7QaACgDFAWn&categoryType=CODE&problemTitle=%EC%82%BC%EC%84%B1%EC%8B%9C%EC%9D%98&orderBy=FIRST_REG_DATETIME&selectCodeLang=ALL&select-1=&pageSize=10&pageIndex=1

SW Expert Academy

▼ 6485. 삼성시의 버스노선

```

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

public class SW6485_삼성시의버스노선 {
    public static void main(String[] args) throws NumberFormatException, IOException {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        StringBuilder sb = new StringBuilder();
        StringTokenizer st;
        int TC = Integer.parseInt(in.readLine());
        for(int tc = 1 ; tc <= TC ; tc++) {
            int N = Integer.parseInt(in.readLine()); // 버스 노선의 개수
            int[] v = new int[5001]; // 버스 정류장 개수만큼(노선들이 몇번 지나가는지를 저장)
            for(int n = 0 ; n < N ; n++) { // 버스 노선만큼 반복
                st = new StringTokenizer(in.readLine());
                int start = Integer.parseInt(st.nextToken());
                int end = Integer.parseInt(st.nextToken());
                for(int i = start ; i <= end ; i++) { // 해당 버스 노선이 지나가는 정류장에 대해 1씩 증가
                    v[i]++;
                }
            }
            int P = Integer.parseInt(in.readLine());
            sb.append("#" + tc + " ");
            for(int p = 0 ; p < P ; p++) { // 주어진 버스 정류장에 몇 개의 버스 노선이 지나가는지를 결과에더함
                int n = Integer.parseInt(in.readLine());
                sb.append(v[n] + " ");
            }
            sb.append("\n");
        }

        System.out.println(sb);
    }
}

```

10157번: 자리배정

어떤 공연장에는 가로로 C개, 세로로 R개의 좌석이 C×R격자형으로 배치되어 있다. 각 좌석의 번호는 해당 격자의 좌표 (x,y)로 표시된다. 예를 들어, 아래 그림은 가로 7개, 세로 6개 좌석으로 구성된 7×6격자형 좌석배치를 보여주고 있다. 그림에서 각 단위 사각형은 개별 좌석을 나타내며, 그 안에 표시된 값 (x,y)는 해당 좌석의 번호를 나타낸다.

<https://www.acmicpc.net/problem/10157>

BAEKJOON
ONLINE JUDGE

▼ 10157 - 자리배정

▼ 소스코드

```

package d4s8.week3;

import java.util.Scanner;

public class BOJ_10157_자리배정 {
    static int C, R, K; // 행, 열, 찾는 자리
    static int[][] map;
    static int dir = 0;

    // 우 하 좌 상 : 시계방향
    static int[] dx = { 0, 1, 0, -1 };
    static int[] dy = { 1, 0, -1, 0 };

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        C = sc.nextInt();
        R = sc.nextInt();
        map = new int[C][R];
        K = sc.nextInt();

        if(K > C*R) {
            System.out.println(0);
            return;
        }

        int nowi = 0;
        int nowj = 0;
        map[nowi][nowj] = 1;

        while (true) {
            /**순서 주의. K가 1일 경우까지 생각해야함.**/
            if (map[nowi][nowj] == K) {
                System.out.println(++nowi + " " + ++nowj);
                return;
            }

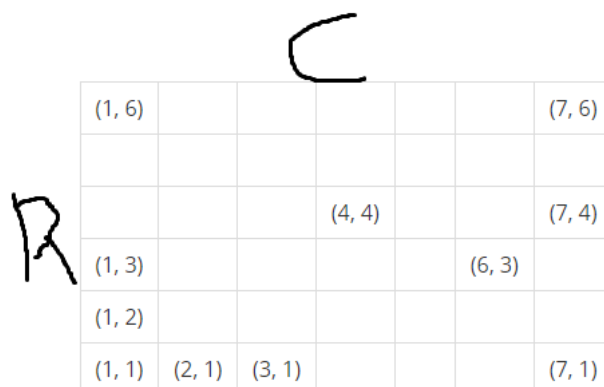
            int nx = nowi + dx[dir];
            int ny = nowj + dy[dir];
            if (nx < 0 || nx >= C || ny < 0 || ny >= R || map[nx][ny] > 0) {
                dir = (dir + 1) % 4;
                continue;
            }
            map[nx][ny] = map[nowi][nowj] + 1;
            nowi = nx;
            nowj = ny;
        }
    }
}

```

로직

- 문제에서 주어진 배열이 평소 알던 배열처럼 만들어서 생각.

오른쪽으로 90도 회전시킴. + 0,0부터 시작



R

(1, 1)	(1, 2)	(1, 3)			(1, 6)
(2, 1)					
(3, 1)					
			(4, 4)		
		(6, 3)			
(7, 1)			(7, 4)		(7, 6)

C

2. (0,0)에서 시작해서 시계방향으로 이동.
3. 인덱스 벗어나거나 이미 방문했으면 방향 바꾸기.
4. 진행하면서 K번째 찾으면 좌표출력.

틀린 이유

- 문제를 제대로 읽지 않아서 좌석 배정할 수 없으면 0 출력하는 것을 빼먹음.

```
/*K를 입력받은 순간 범위보다 크다면 바로 0출력하고 리턴*/
if(K > C*R) {
    System.out.println(0);
    return;
}
```

- K=1인 경우를 생각하지 않았음.

```
while (true) {
    /**순서 주의. K가 1일 경우까지 생각해야함.*/
    if (map[nowi][nowj] == K) {
        System.out.println(++nowi + " " + ++nowj);
        return;
    }

    int nx = nowi + dx[dir];
    int ny = nowj + dy[dir];
    if (nx < 0 || nx >= C || ny < 0 || ny >= R || map[nx][ny] > 0) {
        dir = (dir + 1) % 4;
        continue;
    }
    map[nx][ny] = map[nowi][nowj] + 1;
    nowi = nx;
    nowj = ny;

    /** 순서 주의. if문이 먼저 온다면 map[nx][ny]에 값이 0이므로 K번째 좌석을 찾을 수가 없다.
    if (map[nx][ny] == K) {
        System.out.println(++nx + " " + ++ny);
        return;
    }
}
```

```

/*틀린 예*/
/*1인 경우를 제외하고 다 찾을 수 있음. 그리고 위에 map[nx][ny]에 값 넣는것 보다 앞에
해당 코드를 작성해서 틀렸음.*/
if (map[nx][ny] == K) {
    System.out.println(++nx + " " + ++ny);
    return;
}

/*성공 예*/
/*반복문을 시작하자 마자 map을 확인해서 K 라면 현재 좌표에서 +1씩해서 출력하고 리턴*/
if (map[nowi][nowj] == K) {
    System.out.println(++nowi + " " + ++nowj);
    return;
}

```

1592번: 영식이와 친구들

일단 1번 자리에 앉은 사람이 공을 받는다. 그리고 나서 공을 다른 사람에게 던진다. 다시 공을 받은 사람은 다시 공을 던지고, 이를 계속 반복한다. 한 사람이 공을 M번 받았으면 게임은 끝난다. (지금 받은 공도 포함하여 센다.)

BAEKJOON
ONLINE JUDGE

<https://www.acmicpc.net/problem/1592>

▼ 1592 - 영식이와 친구들

▼ 코드

• 김중재 코드

```

package d4s8.week3;

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

public class BOJ_1592_영식이와친구들 {
    static int N, M, L; // 번호, 종료조건, 분기조건
    static int[] arr; // 각자 몇번 받았는지 기록하는 배열
    static int cnt, idx; // 몇번 주고 받는지 횟수, 시작 번호

    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());
        L = Integer.parseInt(st.nextToken());
        arr = new int[N + 1];

        cnt = 0;
        idx = 1;
        arr[idx]++;

        while (true) {
            if (arr[idx] == M) { // 해당 자리에서 M번 받았다면 종료
                System.out.println(cnt);
                return;
            }
            if (arr[idx] % 2 != 0) { // 해당 번호에서 홀수번 받았다면 시계방향으로 진행
                idx = (idx + L) % N;
            } else { // 해당 번호에서 짝수번 받았다면 시계방향으로 진행
                if (L >= idx) {
                    idx = N + (idx - L);
                } else {
                    idx = idx - L;
                }
            }
            idx = idx == 0 ? N : idx; // %N 연산을 해서 마지막 번호가 0으로 되기 때문에 필요
            arr[idx]++; // 공 받았으니 +1 해줌.
            cnt++; // 공 던졌으니 +1 해줌.
        }
    }
}

```

• 정연님 코드

```

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

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(in.readLine());
        final int N = Integer.parseInt(st.nextToken());
        final int M = Integer.parseInt(st.nextToken());
        final int L = Integer.parseInt(st.nextToken());
        int result = 0, pos = 0, now = 0;
        int[] friends = new int[N];
        while(true) {
            friends[pos]++;
            now = friends[pos];
            if(now == M) break;
            if(now % 2 == 0) pos = (pos - L) < 0 ? pos + N - L : pos - L;
            else pos = (pos + L) % N;
            result++;
        }
        System.out.println(result);
    }
}

```

삼항연산자를 활용해서 코드가 더 간결하고 깔끔하다.

로직

1. 짝수인 경우와 홀수인 경우 나눠서 생각.

a. 홀수인 경우는 시계방향으로 진행인데, 이 경우에는 $(idx+L)\%N$ 을 해줘도 정확한 번호가 나오므로 딱히 신경쓸게 없음.

b. 짝수인 경우는 $idx - L$ 연산을 했을 경우 0보다 작거나 같은 경우 특별히 처리를 해줘야 함.

i. 먼저 $L < idx$ 라면 무조건 0보다 크므로 그냥 $idx = idx - L$ 을 해주면됨.

ii. $L \geq idx$ 라면 0보다 작거나 같으므로 N에서 $idx - L$ 만큼을 더 움직여야함.

$\Rightarrow idx = N + (idx - L)$

2. 1부터 N까지여서 N+1인 배열을 선언해서 사용했기 때문에 위의 연산을 수행했을 때 idx가 0이 된다면 마지막 번호인 N으로 바꿔줌.

3. 이제 해당 번호가 받는 공의 횟수($arr[idx]$)와 전체 공을 몇번 던지는 지(cnt)를 +1씩 해줌.

4. 계속 진행하다가 어떤 한명이 M번 공을 받으면 바로 cnt를 출력시켜서 종료시킴.

틀린 이유

- 문제를 제대로 읽지 않아서 홀수, 짝수 인덱스가 기준인 줄 알고 조금 해맸음..