

# 10월 2주차 스터디

🔗 URL	
📌 강의 번호	
☑ 복습	<input type="checkbox"/>
📌 유형	
📎 자료	
🕒 작성일시	@2021년 10월 16일 오전 9:53

## 10월 2주차 기출문제

Aa 출처	# 문제번호	≡ 문제명	≡ 난이도	🔗 링크
백준	21610	마법사 상어와 비바라기	골드 5	<a href="https://www.acmicpc.net/problem/21610">https://www.acmicpc.net/problem/21610</a>
프로그래머스	1830	브라이언의 고민	레벨 3	<a href="https://programmers.co.kr/learn/courses/30/lessons/1830">https://programmers.co.kr/learn/courses/30/lessons/1830</a>
프로그래머스	42584	주식가격	레벨 2	<a href="https://programmers.co.kr/learn/courses/30/lessons/42584">https://programmers.co.kr/learn/courses/30/lessons/42584</a>

## 김석원

### 2563번: 색종이

문제 가로, 세로의 크기가 각각 100인 정사각형 모양의 흰색 도화지가 있다. 이 도화지 위에 가로, 세로의 크기가 각각 10인 정사각형 모양의 검은색 색종이를 색종이의 변과 도화지의 변이 평행하도록 붙인다. 이러한 방식으로 색종이를 한 장 또는 여러 장 붙인 후 색종이가 붙은 검은

<https://www.acmicpc.net/problem/2563>

BAEKJOON  
ONLINE JUDGE

### ▼ 코드

```
package week10_2;

import java.util.Arrays;
import java.util.Scanner;

public class 색종이 {
    private static int[][] board;

    public static void main(String[] args) {
        final int paper_length = 100;
        final int color_length = 10;
        Scanner sc = new Scanner(System.in);
        int totalcnt = 0;
        int n = sc.nextInt();
        board = new int [paper_length][paper_length];
        for(int idx = 0 ;idx < n; idx++) {
            int x = sc.nextInt();
            int y = sc.nextInt();

            for(int i = x-1; i < x+9; i++) {
                for(int j = y-1; j < y+9; j++) {
```

```

        if(board[i][j] == 0) totalcnt++;
        board[i][j] = 1;
    }
}
} //end i
System.out.println(totalcnt);
}
}


```

#### ▼ 풀이

1. board의 크기는 100\*100
2. board에 넣을 색종이의 크기는 10\*10
3. 입력값은 색종이 왼쪽의 x값과 밑면의 y좌표가 들어온다.
4. 100\*100 2차원 배열을 만들고 인덱스가 0~99임에 유의해서
5. x는 x-1 ~ x-1 + 10
6. y는 y-1 ~ y-1 + 10까지 돌려주면서 board에 1이나 true를 표시해
7. 나중에 이값일 세주거나 돌릴때 해당 board값이 0이면 cnt를 증가시켜 출력

#### 코딩테스트 연습 - 주가가격

초 단위로 기록된 주가가격이 담긴 배열 prices가 매개변수로 주어질 때, 가격이 떨어지지 않은 기간은 몇 초인지를 return 하도록 solution 함수를 완성하세요. 제한사항 prices의 각 가격은 1 이상 10,000 이하인 자연수입니다. prices의 길이는 2 이상 100,000 이하입니다. 입출력 예 입

 <https://programmers.co.kr/learn/courses/30/lessons/42584>



#### ▼ 코드

##### ▼ 풀이전략

10000개의 가격이 들어온다면

0번부터 10000-1번째 인덱스까지 비교하는데

10000번의 비교 + 9999번의 비교 + 9998+ ... +3+2+1을 하면

시그마 1~10000이므로 5000만가까이 된다.

단순비교는 불가

min

##### ▼ 소감




스터디 덕분에 문제라도 읽고 한문제라도 풀려고 노력하는거 같아서 좋았습니다.

## 김유진

### 코딩테스트 연습 - 주식가격

초 단위로 기록된 주식가격이 담긴 배열 prices가 매개변수로 주어질 때, 가격이 떨어지지 않은 기간은 몇 초인지를 return 하도록 solution 함수를 완성하세요. 제한사항 prices의 각 가격은 1 이상 10,000 이하인 자연수입니다. prices의 길이는 2 이상 100,000 이하입니다. 입출력 예 입

 <https://programmers.co.kr/learn/courses/30/lessons/42584>



#### ▼ code

```
import java.util.Arrays;
import java.util.Stack;

/**
 * @FileName : P42584_주식가격_2.java
 * @Date : 2021. 10. 11.
 * @작성자 : KimYuJin
 * @특이점 : stack이 비어 있는 경우 처리를 안해서 조금 헤멤 ㅋㅋㅋ
 */
public class P42584_주식가격_2 {
    public static void main(String[] args) {
        int[] prices = { 1, 2, 3, 2, 3 };
        System.out.println(Arrays.toString(solution(prices)));
    }

    static public int[] solution(int[] prices) {
        int size = prices.length;
        int[] answer = new int[size];
        Stack<Integer> s = new Stack<Integer>();

        for (int i = 0; i < size; i++) {
            while (!s.isEmpty() && prices[s.peek()] > prices[i]) {
                int tmp = s.pop();
                answer[tmp] = i - tmp;
            }
            s.push(i);
        }

        while (!s.isEmpty()) {
            int tmp = s.pop();
            answer[tmp] = size - 1 - tmp;
        }

        return answer;
    }
}
```

### 21610번: 마법사 상어와 비바라기

마법사 상어는 파이어볼, 토네이도, 파이어스톰, 물복사버그 마법을 할 수 있다. 오늘 새로 배운 마법은 비바라기이다. 비바라기를 시전하면 하늘에 비구름을 만들 수 있다. 오늘은 비바라기를 크기가 N×N인 격자에서 연습하려고 한다. 격자의 각 칸에는 바구니가 하나 있고, 바구니는 칸

 <https://www.acmicpc.net/problem/21610>

BAE<K>JOON>  
ONLINE JUDGE

#### ▼ code

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

```

public class B21610_마법사상어와비바라기_1 {
    static int N, M, cnt;
    static int[][] com, map;
    static boolean[][] chk;
    static int[] dr = { 0, -1, -1, -1, 0, 1, 1, 1 };
    static int[] dc = { -1, -1, 0, 1, 1, 1, 0, -1 };
    static Cloud[] c;

    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][N];
        for (int r = 0; r < N; r++) {
            st = new StringTokenizer(br.readLine());
            for (int c = 0; c < N; c++) {
                map[r][c] = Integer.parseInt(st.nextToken());
            }
        }
        com = new int[M][2];
        for (int m = 0; m < M; m++) {
            st = new StringTokenizer(br.readLine());
            com[m][0] = Integer.parseInt(st.nextToken()) - 1;
            com[m][1] = Integer.parseInt(st.nextToken());
        }

        initCloud();

        simul();

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

    static int cntWater() {
        int ans = 0;
        for (int r = 0; r < N; r++) {
            for (int c = 0; c < N; c++) {
                ans += map[r][c];
            }
        }
        return ans;
    }

    static void simul() {
        for (int m = 0; m < M; m++) {
            chk = new boolean[N][N];
            moveRainDeleteCloud(m);

            waterCopyMakeCloud();
        }
    }

    static void waterCopyMakeCloud() {
        int[][] tmp = new int[N][N];
        for (int i = 0; i < cnt; i++) {
            for (int d = 0; d < 4; d++) {
                int nr = c[i].r + dr[d * 2 + 1];
                int nc = c[i].c + dc[d * 2 + 1];
                if (nr >= 0 && nr < N && nc >= 0 && nc < N) {
                    if (map[nr][nc] > 0)
                        ++tmp[c[i].r][c[i].c];
                }
            }
        }

        Stack<Cloud> s = new Stack<Cloud>();

        for (int r = 0; r < N; r++) {
            for (int c = 0; c < N; c++) {
                map[r][c] += tmp[r][c];
            }
        }
    }
}

```

```

        if (!chk[r][c] && map[r][c] >= 2) {
            s.push(new Cloud(r, c));
            map[r][c] -= 2;
        }
    }
}

cnt = s.size();
c = new Cloud[cnt];
for (int i = 0; i < cnt; i++) {
    c[i] = s.pop();
}

static void moveRainDeleteCloud(int m) {
    for (int i = 0; i < cnt; i++) {
        c[i].r += dr[com[m][0]] * com[m][1];
        c[i].c += dc[com[m][0]] * com[m][1];
        c[i].rePos();
        int nr = c[i].r;
        int nc = c[i].c;
        map[nr][nc] += 1;
        chk[nr][nc] = true;
    }
}

static void initCloud() {
    c = new Cloud[4];
    c[0] = new Cloud(N - 1, 0);
    c[1] = new Cloud(N - 1, 1);
    c[2] = new Cloud(N - 2, 0);
    c[3] = new Cloud(N - 2, 1);
    cnt = 4;
}

static class Cloud {
    int r, c;

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

    void rePos() {
        if (this.r >= N)
            this.r = this.r % N;
        if (this.c >= N)
            this.c = this.c % N;
        int tmp = 6000 - 6000 % N;
        if (this.r < 0)
            this.r = (this.r + tmp) % N;
        if (this.c < 0)
            this.c = (this.c + tmp) % N;
    }
}
}

```

## 배나영

---

### 코딩테스트 연습 - 주식가격

초 단위로 기록된 주식가격이 담긴 배열 prices가 매개변수로 주어질 때, 가격이 떨어지지 않은 기간은 몇 초인지를 return 하도록 solution 함수를 완성하세요. 제한사항 prices의 각 가격은 1 이상 10,000 이하인 자연수입니다. prices의 길이는 2 이상 100,000 이하입니다. 입출력 예 입

🔗 <https://programmers.co.kr/learn/courses/30/lessons/42584>



#### ▼ code

```
package week2;

import java.util.Scanner;

public class P42584_주식가격 {

    public static int[] solution(int[] prices) {
        // answer: 가격이 떨어지지 않은 기간 저장
        // prices의 인덱스=초로 생각
        int[] answer=new int[prices.length];

        for(int index=0; index<prices.length;index++) { // prices 각 인덱스마다 자기보다 작은 게 언제 처음으로 나오는지 저장
            int check=index; // check: 자기보다 작은 게 언제 처음으로 나오는지 저장

            while(true) { // 자기보다 작은 값을 찾을 때까지
                if(check==prices.length-1) break; // 끝날 때까지 자기보다 작은 게 없는 경우 while 탈출
                if(prices[check]<prices[index]) break; // 자기보다 작은 값을 찾을 경우 while 탈출
                check++; // 자기보다 작은 것 찾지 못했으면 다음 위치 탐색
            }

            answer[index]=check-index; // 자기보다 작은 게 처음 나오는 초-현재=가격이 떨어지지 않은 기간
        }

        return answer;
    }

    public static void main(String[] args) {

        int[] prices= {1,2,3,2,3};
        int[] answer=solution(prices);

        for(int i=0;i<answer.length;i++) {
            System.out.print(answer[i]+" ");
        }
    }
}
```

#### ▼ 소감



코딩 트랙을 진행하면서 자연스럽게 알고리즘을 소홀히 하게 되었는데 스터디를 하느라 조금씩이라도 알고리즘 문제를 풀 수 있게 되어 좋았습니다.

## 김응철

### 코딩테스트 연습 - 주식가격

초 단위로 기록된 주식가격이 담긴 배열 prices가 매개변수로 주어질 때, 가격이 떨어지지 않은 기간은 몇 초인지를 return 하도록 solution 함수를 완성하세요. 제한사항 prices의 각 가격은 1 이상 10,000 이하인 자연수입니다. prices의 길이는 2 이상 100,000 이하입니다. 입출력 예 입

🔗 <https://programmers.co.kr/learn/courses/30/lessons/42584>



#### ▼ code

```
import java.util.*;

import java.io.*;

public class Solution_주식가격 {

    // price[]로 인풋이 들어오면 같은 크기의 answer[]에서 초 적여주기.

    public static void main(String[] args) {
        int[] prices = { 1, 2, 3, 2, 3 };

        int[] result = new int[5];

        result = solution(prices);

        for (int i = 0; i < 5; i++) {

            System.out.print(result[i] + " ");

        }

    }

    public static int[] solution(int[] prices) {

        int[] answer = {};

        answer = new int[prices.length];

        for (int i = 0; i < prices.length - 1; i++) {

            answer[i] = bf(prices, prices[i], i + 1, 0);

        }

        return answer;

    }

    private static int bf(int[] prices, int start, int i, int cnt) {

        if (prices[i] < start || i == prices.length - 1)

            return cnt + 1;

        return bf(prices, start, i + 1, cnt + 1);

    }

}
```

#### ▼ 소감



하나의 문제를 다양한 관점에서 볼 수 있다는 점이 항상 유익하다고 느껴집니다. 다음주는 더 많은 문제를 풀 수 있게 노력해보겠습니다.

