스터디 발표

날짜	@2021년 8월 27일 오후 8:00
태그	#2조 #4주차 #IM대비 #스터디 #의석이의우뚝선산 #자리배정 #팀발표

[8월 4주차 스터디]

1. SWEA 4796번 의석이의 우뚝 선 산

▼ 문제 링크

SW Expert Academy

SW 프로그래밍 역량 강화에 도움이 되는 다양한 학습 컨텐츠를 확인하세요!

https://swexpertacademy.com/main/code/problem/problemDetail.do?contestProbId=AWS2h6AKBCoDFAVT&categoryId=AWS2h6AKBCoDFAVT&categoryType=CODE&problemTitle=4796&orderBy=FIRST_REG_DATETIME&selectCodeLang=ALL&select-1=&pageSize=10&pageIndex=1

첫번째 시도 (with Mathmatical rules)

👌 이 문제의 경우 규칙성만 파악한다면 아주 쉽게 풀 수 있는 문제이다. k 지점에 대해서 왼쪽 구간은 오름찬순 오른쪽 구간은 내림차순이 이루어지는 구간이 문제에서 찾고자 하는 구간이다. 예를 들어 k 지점의 값이 5 이었고 k 지점을 포함하는 구간 중에서 가장 긴 구간이 1 < 2 < 5 > 4 > 3 라고 한다면 증가되는 부분(1, 2) 과 감소되는 부분(4, 3) 의 곱으로 해당 구간에서 규칙을 만족하는 부분집합의 개수를 추출할 수 있다. 1<2<5>4, 1<2<5>4>3, 2<5>4, 2<5>4>3 다음의 4가지 경우의 수가 2 (증가횟수) * 2 (감소횟수) 를 통해서 구할 수 있는 것이다. 따라서 해당 규칙을 이용해서 규칙을 만족하는 동안 증가하는 부분과 감소하는 부분의 횟수를 체크해서 문제를 해결했다.

```
package com.ssafy.im;

import java.io.IOException;
import java.util.Scanner;

/**
 *
 * @author comkkyu
 * @date 21. 8. 25
 *
 * SWEA 4796 - 의석이의 우뚝 선 산
 * https://swexpertacademy.com/main/code/problem/problemDetail.do?contestProbId=AWS2h6AKBCoDFAVT&categoryId=AWS2h6AKBCoDFAVT&categ
 *
 */
public class SWEA_4796_의석이의우뚝선산 {

  public static void main(String[] args) throws IOException {
    Scanner sc = new Scanner(System.in);
    StringBuilder sb = new StringBuilder();
    int[] arr;
    int T = sc.nextInt();

    for (int t = 1; t <= T; t++) {
      int N = sc.nextInt();
      int cnt = 0;
      arr = new int[N];

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

      cnt = solve(arr, N);

      sb.append("#"+t+" "+cnt+"\n");
    }

    System.out.println(sb);
```

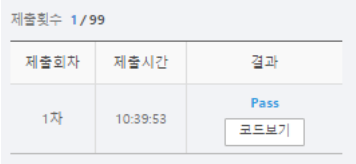
```
    sc.close();
  }

  private static int solve(int[] arr, int n) {
    int high = 0, low = 0, cnt = 0; // 산의 높이 증가 횟수, 산의 높이 감소 횟수, 우뚝 선 산 구간 개수

    for (int i = 1; i < n; i++) {
      if (arr[i - 1] < arr[i]) { // 현재 산이 이전 산보다 높이가 높다면 high++
        if (low > 0) { // 단 이미 한번 낮아졌다가 다시 높아진 경우는 우뚝 선 산이 아니므로
          cnt += high * low; // 지금까지 구한 high 와 low 값으로 구간의 경우의 수를 구한다.
          high = low = 0; // 새로운 구간을 찾아야 하므로 다시 0으로 초기화
        }
        high++;
      } else { // 현재 산의 높이가 이전 산보다 낮으면 low++
        low++;
      }
    }

    cnt += high * low; // 최종적으로 low 로 마무리 된 경우가 있을 수도 있으므로 한번더 경우의 수를 더 해준다.
    return cnt;
  }

}
```



2. 백준 10157번 자리배정

▼ 문제 링크

10157번: 자리배정

어떤 공연장에는 가로로 C개, 세로로 R개의 좌석이 C×R격자형으로 배치되어 있다. 각 좌석의 번호는 해당 격자의 좌표 (x,y)로 표시된다. 예를 들어보자. 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 번호

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

BAE/KJOON ONLINE JUDGE

첫번째 시도 (with LinkedList, Recursive)

❌ 달팽이 모양으로 탐색하는 단순한 문제이다. 시작점으로 부터 4방 탐색을 진행하는데 상, 우, 하, 좌 순으로 해당 방향으로 갈 수 있을때 탐색을 진행하면 된다. 대기번호가 배열의 크기를 넘어가면 0을 출력하고 해당 좌표를 찾으면 문제에서는 왼쪽 하단이 1,1 이므로 해당 좌석번호에 맞게 출력해주면 간단하게 해결되는 문제이다.

```
package com.ssafy.im;

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

/**
 *
 * @author comkkyu
 * @date 21. 8. 25
 *
 * 백준 10157 - 자리배정
 * https://www.acmicpc.net/problem/10157
 *
 */
public class B10157_자리배정 {

  static int[][] map;
  static int[] dx = {-1, 0, 1, 0}; // 상, 우, 하, 좌
  static int[] dy = {0, 1, 0, -1};
  static int R, C;
```

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

    map = new int[R][C];

    int K = Integer.parseInt(br.readLine());

    solve(K);
  }

  private static void solve(int k) {

    if (k > R * C) { // 대기번호가 배열의 크기를 벗어난다면
      System.out.println(0); // 관객에게 좌석을 배정할 수 없으므로 0을 출력
      return;
    }

    int cnt = 1; // 시작개수는 1
    int cx = R - 1; // 시작 행
    int cy = 0; // 시작 열
    int dir = 0; // 처음에는 위 방향으로

    while (cnt != k) {
      map[cx][cy] = cnt; // cnt 대기번호를 가진 관객의 좌석 위치는 map[cx][cy] 임을 표시

      int nx = cx + dx[dir]; // 현재 방향에 대한 다음 위치의 행
      int ny = cy + dy[dir]; // 현재 방향에 대한 다음 위치의 열

      if (!isInside(nx, ny) || map[nx][ny] != 0) { // 배열범위를 벗어나거나 이미 예약된 좌석이라면
        dir = (dir + 1) == 4 ? 0 : dir + 1; // 방향을 바꿔줘야 함.
        nx = cx + dx[dir]; // 바꾼 방향에 대해서 탐색한 행 정보로 바꿔준다.
        ny = cy + dy[dir]; // 바꾼 방향에 대해서 탐색한 열 정보로 바꿔준다.
      }

      // 현재 좌표를 갱신
      cx = nx;
      cy = ny;
      // 탐색이 끝났으므로 cnt 1 증가
      cnt++;
    }

    System.out.println((cy+1)+" "+(R-cx)); // 문제에서는 왼쪽 한단이 (1,1) 이므로 주어진 형식에 맞춰서 바꿔서 출력
  }

  private static boolean isInside(int r, int c) {
    return r >= 0 && r < R && c >= 0 && c < C;
  }
}
```

comkkyu	10157	맞았습니다!!	19688 KB	196 ms	Java 11 / 수정	1436 B

스터디 팀원 풀이 발표

1. 백준 2810번 컵홀더

▼ 문제 링크

2810번: 컵홀더

십년이면 강산이 변한다. 강산이네 동네에 드디어 극장이 생겼고, 강산이는 극장에 놀러갔다. 매점에서 콜라를 산 뒤, 자리에 앉은 강산이는 큰 혼란에 빠졌다. 양쪽 컵홀더를 이미 옆 사람들이 차지했기 때문에 콜라를 꽂을 컵 홀더가 없었기 때문이다. 영화를 보는 내내 콜라를 손에 들고 있던 강산이는 극장에 다시 왔을 때는 꼭 콜라를 컵 홀

https://www.acmicpc.net/problem/2810

BAE/KJOON ONLINE JUDGE

안예지님 풀이

〈 백준 - 2810 컵홀더 〉

- 큐를 이용해서 Peek이 S면 한번 Poll L이면 두번 Poll
- Poll을 할때마다 컵홀더 카운터 ++

```
public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int num = Integer.parseInt(br.readLine());
    Queue<Character> qu = new LinkedList<>();
    StringBuilder sb = new StringBuilder();
    String seat = br.readLine();

    int cnt = 1;
    int person = 0;
    for (int i = 0; i < seat.length(); i++) {
        qu.add(seat.charAt(i));
        // System.out.println(seat.charAt(i));
    }
```

→ 문자열로 받은 Seat을 문자로 잘라 큐에 저장

```
int cnt = 1;
int person = 0;
for (int i = 0; i < seat.length(); i++) {
    qu.add(seat.charAt(i));
    // System.out.println(seat.charAt(i));
}

while (!qu.isEmpty()) {

    if (qu.peek() == 'S') {
        qu.poll();
        cnt += 1;
        person++;

    } else if (qu.peek() == 'L') {
        qu.poll();
        qu.poll();

        cnt += 1;
        person += 2;
    }
}

if (cnt <= person) {
    System.out.println(cnt);
} else if (cnt > person) {
    System.out.println(person);
}
```

→ 만약 S면
컵홀더와 사람 ++ → 한번 Poll
→ L이면
두번 Poll
커플좌석이므로 → 사람은 2명 +
→ 컵홀더보다 사람이 많으면 컵홀더 수 출력 아니면 사람수 출력

2. 백준 2581번 소수

▼ 문제링크

2581번: 소수

BAEKJOON ONLINE JUDGE

https://www.acmicpc.net/problem/2581

김응철님 풀이 (with 에라토스테네스의 체)

```
import java.util.Scanner;

public class 소수_2581 {

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      int M = sc.nextInt();
      int N = sc.nextInt();

      int sum = 0, min = Integer.MAX_VALUE;

      boolean[] arr = new boolean[N + 1];

      arr[0] = true;
      arr[1] = true;

      for (int i = 2; i <= Math.sqrt(N + 1); i++) {
         for (int j = i * i; j < N + 1; j += i) {
            arr[j] = true;
         }
      }
      // arr = 60 61 62 63 ... 100

      for (int i = M; i < N + 1; i++) {
         if (arr[i] == false) {
            if (min > i) {
               min = i;
            }
            sum += i;
         }
      }

      if (sum == 0)
         System.out.println(-1);
      else
         System.out.printf("%d\n%d", sum, min);

   }
}
//4부터 100까지  4 6 8 10
//9부터 100까지 9 12 15
//16부터 100까지 16 20
```

새롭게 알게된 내용

1. 에라토스테네스의 체 (소수를 찾는 알고리즘)

   ▼ 관련 링크

에라토스테네스의 체 - 위키백과, 우리 모두의 백과사전

https://ko.wikipedia.org/wiki/%EC%97%90%EB%9D%BC%ED%86%A0%EC%8A%A4%ED%85%8C%EB%84%A4%EC%8A%A4%EC%9D%98_%EC%B2%B4