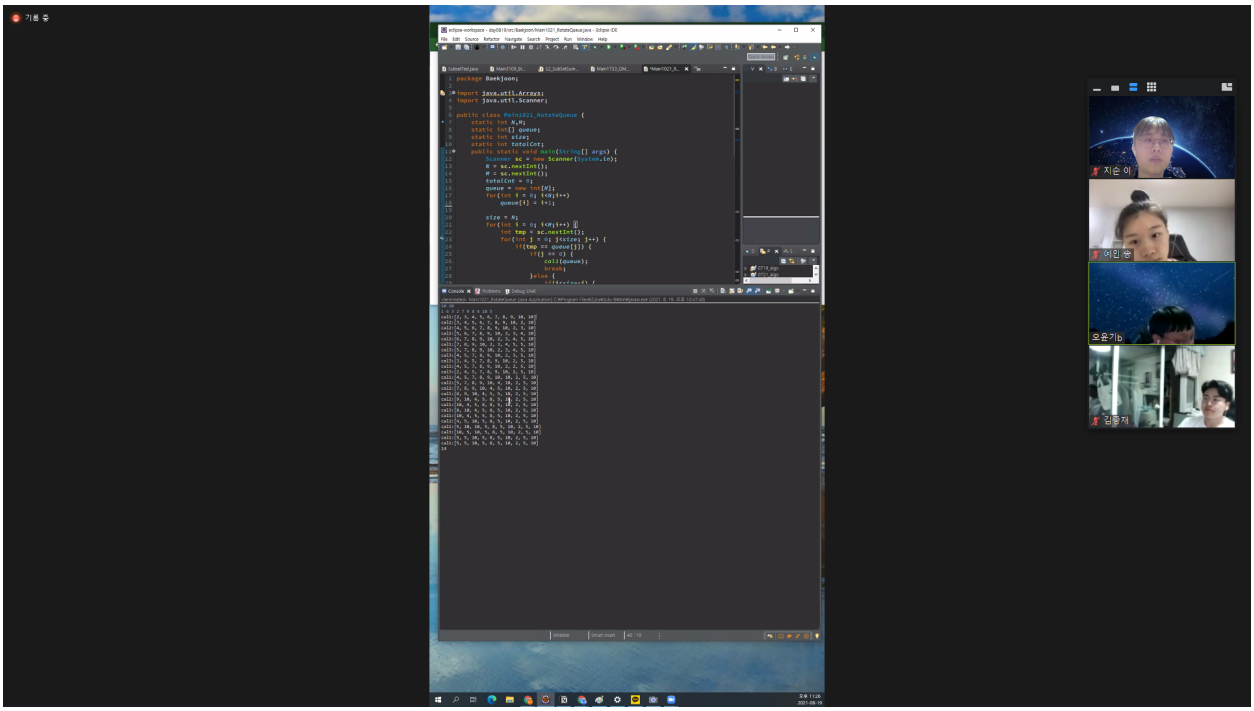
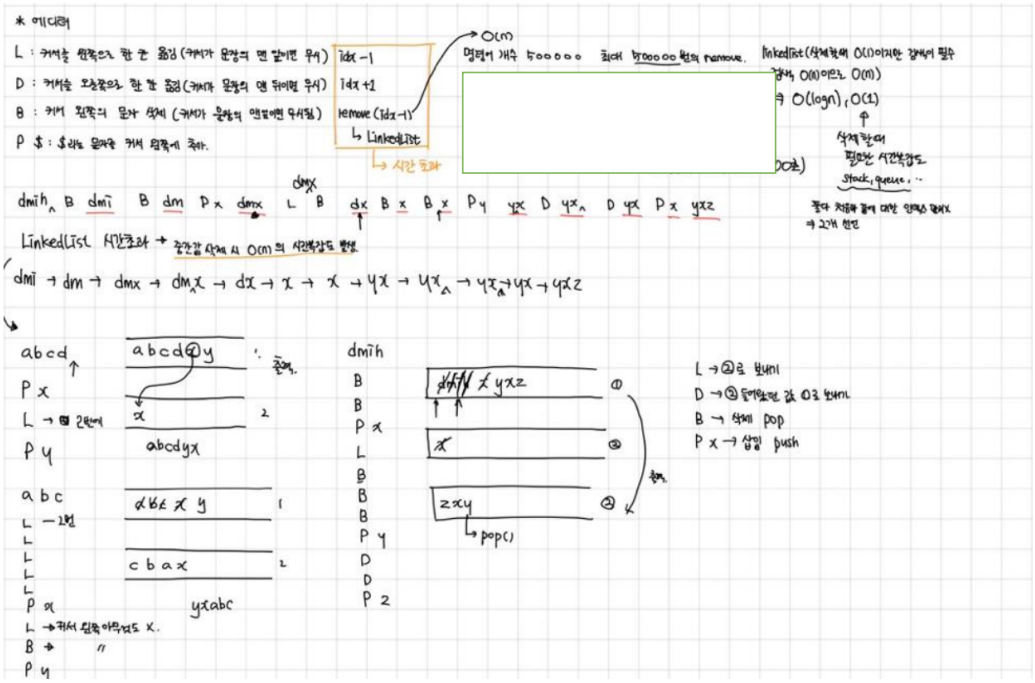
0819 알고리즘 스터디 3주차 그룹모임

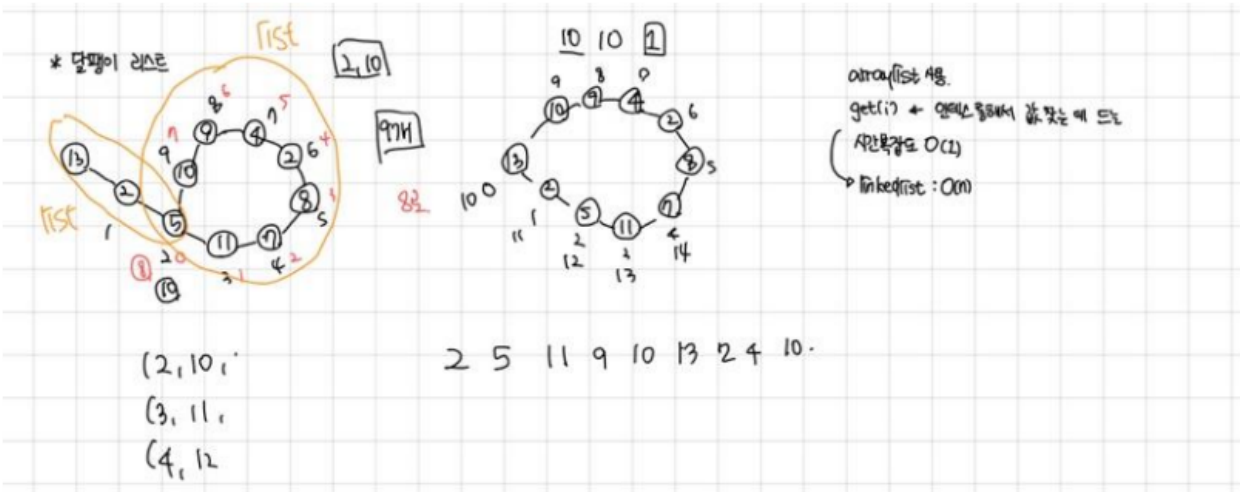
8월 19일 발표화면



문제 풀이시 각자 고민했던 문제들 1~2문제씩 발표하는 방식으로 스터디 진행하였음.

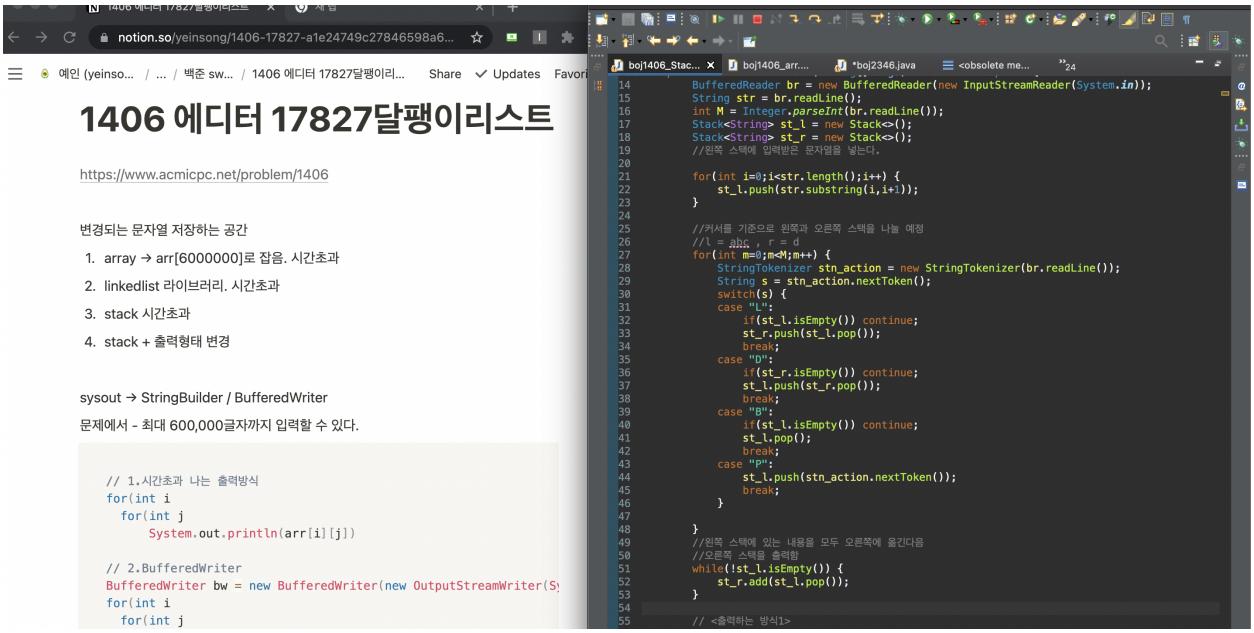
<이지순> - 에디터, 달팽이 리스트





:고민한 부분을 필기로 정리하여 발표.

<송예인> - 달팽이 리스트와 에디터



notion.so/yeinsong/1406-17827-a1e24749c27846598a6...

예인 (yeinso... / ... / 백준 sw... / 1406 에디터 17827달팽이리... Share Updates

1406 에디터 17827달팽이리스트

https://www.acmicpc.net/problem/1406

변경되는 문자열 저장하는 공간

1. array → arr[6000000]로 잡음. 시간초과
2. linkedlist 라이브러리. 시간초과
3. stack 시간초과
4. stack + 출력형태 변경

sysout → StringBuilder / BufferedWriter

문제에서 - 최대 600,000글자까지 입력할 수 있다.

```
// 1.시간초과 나는 출력방식
for(int i
  for(int j
      System.out.println(arr[i][j])

// 2.BufferedWriter
BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(Sy
for(int i
  for(int j
```

boj1406_Stac... boj1406_arr.... *boj2346.java <obsolete me... »24

```
14          BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
15          String str = br.readLine();
16          int M = Integer.parseInt(br.readLine());
17          Stack<String> st_l = new Stack<>();
18          Stack<String> st_r = new Stack<>();
19          //왼쪽 스택에 입력받은 문자열을 넣는다.
20
21          for(int i=0;i<str.length();i++) {
22              st_l.push(str.substring(i,i+1));
23          }
24
25          //커서를 기준으로 왼쪽과 오른쪽 스택을 나눌 예정
26          //l = abc , r = d
27          for(int m=0;m<M;m++) {
28              StringTokenizer stn_action = new StringTokenizer(br.readLine());
29              String s = stn_action.nextToken();
30              switch(s) {
31              case "L":
32                  if(st_l.isEmpty()) continue;
33                  st_r.push(st_l.pop());
34                  break;
35              case "D":
36                  if(st_r.isEmpty()) continue;
37                  st_l.push(st_r.pop());
38                  break;
39              case "B":
40                  if(st_l.isEmpty()) continue;
41                  st_l.pop();
42                  break;
43              case "P":
44                  st_l.push(stn_action.nextToken());
45                  break;
46              }
47
48          }
49          //왼쪽 스택에 있는 내용을 모두 오른쪽에 옮긴다음
50          //오른쪽 스택을 출력함
51          while(!st_l.isEmpty()) {
52              st_r.add(st_l.pop());
53          }
54
55          // <출력하는 방식1>
```

:시간초과가 나는 이유에 대한 상세한 고찰과 System.out.println 함수를 분석하심.

<오윤기> - 회전하는 큐

초기상태 : 1 2 3 4 5 6 7 8 9 10
2번1회 : 2 3 4 5 6 7 8 9 10 1
1번연산 : 3 4 5 6 7 8 9 10 1
3번 3회 : 9 10 1 3 4 5 6 7 8
1번연산: 10 1 3 4 5 6 7 8

1. 2번 연산과 3번 연산의 횟수를 최소화 하기 위하여 시행 전에 왼쪽연산과 오른쪽 연산중 더 적은 횟수로 1번째 자리로 이동할 수 있는 연산의 종류를 판단한다
* 판단방법: 현재 수행할 인덱스 i와 [(큐 사이즈) - i] 를 비교하여 i가 작은경우 2번연산,
그렇지 않은 경우 3번연산을 수행하도록 진행.
2. 1번의 과정을 진행했다면 1번연산을 수행.
3. 추출해야할 모든 원소가 남아 있지 않을 때 까지 1~3번 반복수행
4. 2번연산과 3번연산을 진행할 때마다 정적변수 CNT의 값을 올려주고 3번의 진행과정이 끝나면 CNT 출력

:연결 리스트로도 풀이가 가능하지만 배열을 이용하여 문제 주어진 상황 그대로 풀이를 진행함.

<김중재> - 풍선 터뜨리기, 회전하는 큐

```java
/*좋은 입력예시
 * 5
 *-5 -5 -5 -5 -5
 **/

import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;

public class BOJ_2346_풍선터뜨리기 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        List<Node> arr = new LinkedList<Node>();

        for (int i = 1; i <= N; i++) {
            arr.add(new Node(i, sc.nextInt()));
        }

        int idx = 0;
        while (arr.size() > 1) {    // 하나 남았으면 의미가 없으므로 탈출
            System.out.print(arr.get(idx).i + " ");
            int data = arr.get(idx).data;
            arr.remove(idx);

            if (data > 0) { // 오른쪽으로 이동 하는 경우
                idx = (idx + data - 1) % arr.size();    // 리스트의 크기보다 커질 수 있으므로 나머지 연산 수행.
            } else {    // 왼쪽으로 이동 하는 경우
                int num = idx + data;

                if (num < 0) {  // 리스트 범위를 벗어난다면 뒤에서 부터 해당 인덱스를 찾아야 됨.
                    int a = Math.abs(num) % arr.size(); // 뒤에서부터 얼마나 더 가야되는지
                    idx = a == 0 ? 0 : arr.size() - a;  // a가 0이 되면 idx를 0으로 해주고, 아니라면 뒤에서 부터 차이나는 만큼 뺀 값을 idx로 해줌.
                } else {
                    idx = num;  // 리스트 범위 벗어나지 않는다면 딱히 계산식이 더 필요하지 않음.
                }
            }
        }
        System.out.print(arr.get(0).i + " ");   // 마지막은 어차피 하나밖에 없으므로 그냥 출력.
    }

    private static class Node {
        int i;
        int data;

        public Node(int i, int data) {
            super();
            this.i = i;
            this.data = data;
        }
    }
}
```

```java
/***
 * 원래 덱 쓸려고했는데 indexOf 가 없어서 연결리스트로 바꿈.
로직
1. 연결리스트에 순서대로 저장
2. 없애려는 수의 인덱스를 indexOf 로 구함.
3. 왼쪽으로 움직일지 오른쪽으로 움직일지 결정
4. move2,move3 함수를 통해 움직이고 연산횟수 더하기.
5. 없애려는수가 맨앞이므로 remove(0)
 * @author Kim
 *
 */

public class BOJ_1021_회전하는큐 {
    static List<Integer> list;
    static int cnt;
    static int N,M;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        list = new LinkedList<Integer>();
        N = sc.nextInt();
        M = sc.nextInt();
        cnt = 0;

        for(int i=1;i<=N;i++) {
            list.add(i);
        }
        /*리스트 입력 완료*/

        for(int i=0;i<M;i++) {
            int num = sc.nextInt();
            int idx = list.indexOf(num);    // 리스트가 계속 이동하기 때문에 입력된 수가 위치한 인덱스를 찾기 위해 사용.
            if(idx !=0) {
                /*left, right : 입력된 num을 제거하기 위해 list의 맨 앞으로 이동시키는 횟수.*/
                int left = idx;
                int right = list.size() - idx;
                if(left < right)    // 왼쪽으로 이동하는게 더 빠른 경우.
                    move2(left);    // 2번 연산
                else                // 오른쪽으로 이동하는게 더 빠른 경우.
                    move3(right);   // 3번 연산
            }
            list.remove(0); // 입력된 num을 제거.
        }

        /*결과 출력*/
        System.out.println(cnt);
    }

    private static void move3(int right) {
        // TODO Auto-generated method stub
        for (int i = 0; i < right; i++) {
            list.add(0,list.remove(list.size()-1)); // 맨뒤에꺼를 빼서 맨 앞에 추가.
            cnt++;
        }
    }

    private static void move2(int left) {
        // TODO Auto-generated method stub
        for (int i = 0; i < left; i++) {
            list.add(list.remove(0));   // 맨앞에꺼를 빼서 맨 뒤에 추가.
            cnt++;
        }
    }
}
```

: 회전하는 큐를 연결 리스트로 푸는 과정에 대해 발표진행.