3주차_링크드리스트 풀이

17827 달팽이리스트

```
import java.util.LinkedList;
import java.util.Scanner;

public class Baekjoon_17827 {

  public static void main(String[] args) {
    // TODO Auto-generated method stub

    Scanner sc = new Scanner(System.in);


    int N = sc.nextInt();
    int TC = sc.nextInt();
    int V = sc.nextInt() -1;

    int tempLength = N - V;
    int[] arr = new int[TC];

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

    for (int i = 0; i < TC; i++) {
      int k = sc.nextInt();

      if (k < N) // 달팽이리스트 길이 보다 k가 짧으면 그대로 출력
        System.out.println(arr[k]);
      else { //// 달팽이리스트보다 긴 경우
        System.out.println(arr[(k-V) % tempLength + V]); // 연결된 원형 부분에서 인덱스 값을 측정하여 반환
      }
    }
  }
}
==>
첫 코딩은 하나씩 카운트해가며 해당 인덱스를 찾는 과정으로 값을 출력했었으나 시간초과.
인덱스로 바로 접근하는 방식으로 코드를 수정하였으나 여전히 시간초과
```

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

public class BJ17827_SnailList {
  public static void main(String[] args) throws IOException {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    StringBuilder sb = new StringBuilder();
    StringTokenizer st = new StringTokenizer(in.readLine());
    final int N = Integer.parseInt(st.nextToken()); // 노드 개수
    final int M = Integer.parseInt(st.nextToken()); // 질문 횟수
    final int V = Integer.parseInt(st.nextToken()); // N번 노드(1번부터 번호를 매길 때, 마지막 노드)가 가리키는 노드의 번호
    int[] list = new int[N];
    st = new StringTokenizer(in.readLine());
    for(int i = 0 ; i < N ; i++) {  // N개의 정수를 저장
      list[i] = Integer.parseInt(st.nextToken());
    }
    for(int i = 0 ; i < M ; i++) {  // M개의 질문에 대해
      int k = Integer.parseInt(in.readLine());  // K번째 노드의 값을 찾기
      int idx;
      if(k < N) idx = k;              // N보다 작으면 인덱스는 K로
      else {                    // N과 같거나 크면 달팽이 사이클을 돌음
        int snailSize = N - V + 1;        // 달팽이 사이클의 크기
        idx = N - snailSize + (k - N + snailSize) % snailSize;  // 달팽이 사이클에 연결되지 않은 노드 개수 + 달팽이 사이클에서 K의 위치 -> 전체 리스트
      }
      sb.append(list[idx] + "\n");  // 결과를 stringbuilder에 추가
    }
    System.out.println(sb);       // 결과 출력
  }
}
==> 시간초과의 우려가 있어서 달팽이 사이클 내의 인덱스를 계산해서 결과 도출
```

```
package algo;

import java.io.*;
```

```
import java.util.*;

public class Baekjoon17827Ver3 {
    static int N, M, V;
    static int[] snailList;

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());
        N = Integer.parseInt(st.nextToken());
        M = Integer.parseInt(st.nextToken());
        V = Integer.parseInt(st.nextToken());

        snailList = new int[N];
        st = new StringTokenizer(br.readLine());
        for (int i = 0; i < N; i++) {
            snailList[i] = Integer.parseInt(st.nextToken());//달팽이리스트 연결안된상태
        }

        for (int i = 0; i < M; i++) {
            int question = Integer.parseInt(br.readLine());//질문거리들

            if (question < N) {//질문거리 N 미만이면 그냥 진행
                System.out.println(snailList[question]);
            } else {//이상이면 계산으로 진행
                int realIdx = ((question - (V - 1)) % (N - (V - 1)) + (V - 1));
                System.out.println(snailList[realIdx]);
            }
        }
    }
}
```

1406 에디터

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.LinkedList;


public class Baekjoon_1406 {

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

    char[] str = br.readLine().toCharArray();
    int TC = Integer.parseInt(br.readLine());

    LinkedList<Character> list = new LinkedList<>();

    for (int i = 0; i < str.length; i++) {
      list.add(str[i]);
    }

    int cursorIdx = list.size();
    for (int i = 0; i < TC; i++) {
      char[] cmd = br.readLine().toCharArray();

      if (cmd[0] == 'L') { //  L이 입력되면
        if (cursorIdx == 0)
          continue;
        cursorIdx = cursorIdx - 1; //커서를 왼쪽으로 이동시킨다.
      } else if (cmd[0] == 'D') { // D가 입력되면
        if (cursorIdx == list.size())
          continue;
        cursorIdx = cursorIdx + 1; // 커서를 오른쪽으로 이동시킨다.
      } else if (cmd[0] == 'B') {// B가 입력되면
        if (cursorIdx == 0)
          continue;
        int moveIdx = cursorIdx - 1; // 현재 커서의 위치의 왼쪽 값을 삭제한다.
        list.remove(moveIdx);
        cursorIdx = cursorIdx - 1;
      }
      else if(cmd[0] == 'P'){ // p가 입력되면
        list.add(cursorIdx,cmd[2]);
        cursorIdx = cursorIdx+1; // 값을 삽입시킨다.
      }
    }
    for(int i = 0 ; i < list.size() ; i++) {
      System.out.print(list.get(i));
    }
  }
```

```
}

==> linkedlist로 구현한 코드. 이렇게 구현하면 시간초과가 발생하고, ListIterator를 사용하여
양방향 탐색이 가능하도록 코드를 수정해야 시간초과가 나지 않음.
```

```
package day0819;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.LinkedList;
import java.util.StringTokenizer;

public class BJ1406_Editor {
  public static void main(String[] args) throws IOException {
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    StringBuilder sb = new StringBuilder();
    StringTokenizer st;
    String s = in.readLine();
    List list = new List();
    list.insert(list.top, new Node(s.charAt(0)));
    Node point = list.top;

    for(int i = 1 ; i < s.length() ; i++) {
      list.insert(point, new Node(s.charAt(i)));
      point = point.right;
    }
    list.insert(point, new Node('0'));    // 마지막 노드 추가(이전 노드를 삭제 또는 삽입하므로)
    point = point.right;          // 마지막 노드를 가리키도록
    int M = Integer.parseInt(in.readLine());
    for(int i = 0 ; i < M ; i++) {
      st = new StringTokenizer(in.readLine());
      char command = st.nextToken().charAt(0);
      switch(command) {
        case 'L' :
          if(point.left != null) point = point.left;
          break;
        case 'D' :
          if(point.right != null) point = point.right;
          break;
        case 'B' :
          if(point != list.top) {
            list.remove(point.left);
          }
          break;
        case 'P' :
          char c = st.nextToken().charAt(0);
          list.insert(point.left, new Node(c));
          break;
      }
//      System.out.println("complete: " + command + " " + point.c);
    }
    for(Node n = list.top ; n != null ; n = n.right) {
      //System.out.println(n.c);
      sb.append(n.c);
    }
    if(list.top != null) sb = sb.deleteCharAt(sb.length() - 1);   // 잘못 들어간 마지막 노드 삭제
    System.out.println(sb);
  }
  static class Node {
    Node left;
    Node right;
    char c;
    public Node(char c) {
      this.c = c;
    }
  }
  static class List {
    Node top;
    List() {
      top = null;
    }
    void insert(Node last, Node n) {  // 이전 노드가 주어졌을 때, 다음 노드 삭제
      if(top == null) top = n;
      else if(last == null) {

        n.right = top;
        top.left = n;
        top = n;
      }
      else if(last.right == null) {
        last.right = n;
        n.left = last;
      }
      else {
```

```
        n.left = last;
        n.right = last.right;
        last.right.left = n;
        last.right = n;
      }
    }
    void remove(Node n) { // 해당 요소 삭제
      Node now = n;
      if(now.left == null) {
        top = n.right;
        if(top != null)
          top.left = null;
      }
      else if(n.right == null){
        now.left.right = null;
      } else {
        now.left.right = n.right;
        now.right.left = n.left;
      }
      n = null;
    }
    void print() {
      for(Node n = top ; n != null ; n = n.right) {
        //System.out.println(n.c);
        System.out.print(n.c + " ");
      }
      System.out.println();
    }
  }
}
```

==> 먼저 링크드리스트를 사용해 접근함. 처음 요소부터 차례대로 접근해 인덱스에 해당하는 요소에 접근하므로 시간 초과 발생. 그 후 노드와 리스트 클래스를 구현해 노드로 커서 역할을 하도록 함

```
package algo;

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

public class Baekjoon1406 {
    static String input;

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        input = br.readLine();
        int commandNum = Integer.parseInt(br.readLine());//커맨드 갯수
        char command;//커맨드
        StringTokenizer st;
        int cursor = input.length();
        for (int i = 0; i < commandNum; i++) {
            st = new StringTokenizer(br.readLine());
            command = st.nextToken().charAt(0);
            switch (command) {//입력받은 커맨드에따라 동작
                case 'L':
                    if (cursor != 0) {
                        cursor -= 1;
                    }
                    break;
                case 'D':
                    if (cursor != input.length()) {
                        cursor += 1;
                    }
                    break;
                case 'B':
                    if (cursor == 0) {
                        break;
                    } else if (cursor == input.length()) {
                        input = input.substring(0, cursor - 1);//스트링 자르는형식으로 진행
                        cursor -= 1;
                    } else {
                        String temp1 = input.substring(0, cursor - 1);
                        String temp2 = input.substring(cursor);
                        input = temp1 + temp2;
                        cursor -= 1;
                    }
                    break;
                case 'P':
                    if (cursor == 0) {
                        input = st.nextToken() + input;
                        cursor += 1;
                    } else if (cursor == input.length()) {
                        input = input + st.nextToken();
                        cursor += 1;
                    } else {
```

```
                    String temp1 = input.substring(0, cursor);
                    String temp2 = input.substring(cursor);
                    input = temp1 + st.nextToken() + temp2;
                    cursor += 1;
                }
                break;

            default:
                break;
        }

    }
    System.out.println(input);
}//시간초과 남, 시간초과 해결 못함 다른 여러가지 방법으로 진행해볼것
}
```