1461. 도서관

출처	BoJ
실습/과제/자율	자율
문제유형	그리디
날짜(처음 풀이)	@2024년 10월 10일
풀이 언어	Java

https://www.acmicpc.net/problem/1461

https://www.acmicpc.net/problem/1461

1. 문제 요약

세준이는 사람들이 마구 놓은 책을 다시 가져다 놓아야 한다.

각 책들의 원래 위치가 주어질 때, 책을 모두 제자리에 놔둘 때 드는 최소 걸음 수를 계산하는 프로그램을 작성하시오.

- 세준이는 현재 0에 있고, 사람들이 마구 놓은 책도 전부 0에 있다.
- 세준이는 한 걸음에 좌표 1칸씩 가며, 책의 원래 위치는 정수 좌표이다.
- 세준이는 한 번에 최대 M권의 책을 들 수 있다.
- 책을 모두 제자리에 놔둔 후에는 다시 0으로 돌아올 필요는 없다.

2. 문제 입출력

입력

- 첫째 줄에 책의 개수 N과, 세준이가 한 번에 들 수 있는 책의 개수 M이 주어진다.
- 둘째 줄에는 책의 위치가 주어진다.

입력예시

```
7 2
-37 2 -6 -39 -29 11 -28
```

출력

- 첫째 줄에 정답을 출력한다.

출력예시

```
131
```

📣 3. 제한사항

- **시간** : 2초
- **메모리** : 128MB

- N과 M은 50보다 작거나 같은 자연수이다.
- 책의 위치는 0이 아니며, 절댓값은 10,000보다 작거나 같은 정수이다.

🧐 4. 접근법 (시각적 자료 있으면 좋을듯)

- 최소 걸음 수를 계산해야 함 & 한 번에 최대 M권의 책을 들 수 있다

  ⇒ 가장 먼 위치에 있는 책들을 최대한 많이 운반해야 함

  ⇒ 현재 상황에서 가장 최선의 선택 필요

⏱️ 5. 시간복잡도

💻 6. 코드

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

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());

        int N = Integer.parseInt(st.nextToken()); // 책 개수
        int M = Integer.parseInt(st.nextToken()); // 한 번에 들 수 있는 책 개수

        st = new StringTokenizer(br.readLine());
        List<Integer> negative = new ArrayList<>();
        List<Integer> positive = new ArrayList<>();
```

```
        for (int i = 0; i < N; i++) {
            int position = Integer.parseInt(st.nextToken());
            if (position < 0) {
                negative.add(position);
            } else {
                positive.add(position);
            }
        }

        Collections.sort(negative);
        Collections.sort(positive, Collections.reverseOrder());

        int steps = 0; // 걸음 수
        int maxDistance = 0; // 가장 먼 거리

        for (int i = 0; i < negative.size(); i += M) {
            int distance = Math.abs(negative.get(i));
            steps += distance * 2;
            maxDistance = Math.max(maxDistance, distance);
        }

        for (int i = 0; i < positive.size(); i += M) {
            int distance = positive.get(i);
            steps += distance * 2;
            maxDistance = Math.max(maxDistance, distance);
        }

        // 마지막 가장 멀리 간 곳은 왕복 x
        steps -= maxDistance;

        System.out.println(steps);
    }
}
```