1654. 랜선 자르기

출처	BoJ
실습/과제/자율	자율
문제유형	이분 탐색
날짜(처음 풀이)	@2024년 9월 25일
풀이 언어	Java

https://www.acmicpc.net/problem/1654

https://www.acmicpc.net/problem/1654

🌟 1. 문제 요약

- 현재 K개의 랜선을 가지고 있고, K개의 랜선은 길이가 제각각이다.
- K개의 랜선을 잘라서 모두 같은 길이인 랜선 N개를 만들어야 한다.
  - 편의를 위해 랜선을 자르거나 만들 때 손실되는 길이는 없다고 가정한다.
  - 기존의 K개의 랜선으로 N개의 랜선을 만들 수 없는 경우는 없다고 가정한다.
  - 자를 때는 항상 센티미터 단위로 정수길이만큼 자른다고 가정한다.
  - N개보다 많이 만드는 것도 N개를 만드는 것에 포함된다.
  - 예시

    300cm 짜리 랜선에서 140cm 짜리 랜선을 두 개 잘라내면 20cm는 버려야 한다.

⇒ 만들 수 있는 최대 랜선의 길이를 구하는 프로그램을 작성해라.

➡️ 2. 문제 입출력

입력

- 첫째 줄에는 랜선의 개수 K, 필요한 랜선의 개수 N이 입력된다.
- 그 후 K줄에 걸쳐 이미 가지고 있는 각 랜선의 길이가 센티미터 단위의 정수로 입력된다.

입력예시

```
4 11
802
743
457
539
```

출력

- 첫째 줄에 N개를 만들 수 있는 랜선의 최대 길이를 센티미터 단위의 정수로 출력한다.

출력예시

```
200
```

📣 3. 제한사항

- **시간** : 2초
- **메모리** : 128MB

- K는 1이상 10,000이하의 정수이다.
- N은 1이상 1,000,000이하의 정수이다.’
- 항상 K ≦ N 이다.
- 랜선의 길이는 231-1보다 작거나 같은 자연수이다.

🧐 4. 접근법

▼ 초기 접근법 - 단순 비교 (이분 탐색을 몰랐다고 가정)

1. 주어진 K개의 랜선의 길이를 모두 더하고, N으로 나눈다.

   ⇒ N개의 랜선을 만들 수 있는 랜선의 최대 길이는 나누었을 때의 몫(maxnum)이 된다.

2. while(true)
   - K개 만큼 반복문을 돈다.
     - cnt += K[i]을 maxnum 으로 나누었을 때의 몫
   - cnt ≥ N이면 break, 아니면 maxnum -= 1

- left = 1, right = 가장 긴 랜선의 길이
- while (left <= right)
  - mid = (left + right) / 2;
  - 각 랜선을 mid 길이로 잘랐을 때 만들 수 있는 랜선의 개수를 cnt에 저장한다.
  - cnt가 N개 이상이면, 현재 mid 길이로 N개 이상의 랜선을 만들 수 있다는 뜻이므로, 가능한 길이를 저장하고 더 긴 길이에서도 시도하기 위해 left = mid + 1로 설정한다.
  - cnt가 N개 미만이면, 더 짧은 길이에서 시도하기 위해 right = mid - 1로 설정한다.

💻 5. 코드

▼ 이전 코드 (1차)

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

public class BoJ_1654 {
    static int K, N;
    static int maxnum;
    static int[] kNums;

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

        st = new StringTokenizer(br.readLine());
        K = Integer.parseInt(st.nextToken());
        N = Integer.parseInt(st.nextToken());

        kNums = new int[K];
        int sum = 0;

        for(int i=0; i<K; i++){
            kNums[i] = Integer.parseInt(br.readLine());
            sum += kNums[i];
        }

        maxnum = sum / N;
        int cnt;

        while (true) {
            cnt = 0;
            for (int i = 0; i < K; i++) {
                cnt += kNums[i] / maxnum;
            }

            if (cnt >= N) {
                break;
            }

            maxnum -= 1;
        }

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

▼ 이전 코드 (2차)

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

public class Main {
    static int K, N;
    static int[] kNums;

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

```
        StringTokenizer st;

        st = new StringTokenizer(br.readLine());
        K = Integer.parseInt(st.nextToken());
        N = Integer.parseInt(st.nextToken());

        kNums = new int[K];
        int right = 0; // 가장 긴 랜선 길이 (기존 maxnum)
        for (int i = 0; i < K; i++) {
            kNums[i] = Integer.parseInt(br.readLine());
            if (kNums[i] > right) {
                right = kNums[i];
            }
        }

        int left = 1; // 랜선 최소 길이 1
        int result = 0;

        while (left <= right) {
            int mid = (left + right) / 2;
            int cnt = 0;

            for (int i = 0; i < K; i++) {
                cnt += kNums[i] / mid;
            }

            if (cnt >= N) {
                result = mid; // 가능한 길이를 result에 저장
                left = mid + 1; // 더 긴 랜선 시도
            } else {
                right = mid - 1; // 길이 줄임
            }
        }

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

▼ 최종 코드

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

public class Main {
    static int K, N;
    static int[] kNums;

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

        st = new StringTokenizer(br.readLine());
        K = Integer.parseInt(st.nextToken());
        N = Integer.parseInt(st.nextToken());

        kNums = new int[K];
        long right = 0; // 가장 긴 막대의 길이로 설정
```

```
        long left = 1; // 최소 길이 1

        for (int i = 0; i < K; i++) {
            kNums[i] = Integer.parseInt(br.readLine());
            if (kNums[i] > right) {
                right = kNums[i]; // 최대 길이 설정
            }
        }

        long result = 0;

        // 이분 탐색 시작
        while (left <= right) {
            long mid = (left + right) / 2;
            int cnt = 0;

            // 각 막대를 mid 길이로 잘라서 몇 개가 나오는지 계산
            for (int i = 0; i < K; i++) {
                cnt += kNums[i] / mid;
            }

            if (cnt >= N) { // mid 길이로 N개 이상 만들 수 있는 경우
                result = mid; // 가능한 길이를 result에 저장
                left = mid + 1; // 더 긴 막대로 시도
            } else { // N개 미만인 경우
                right = mid - 1; // 길이를 줄임
            }
        }

        System.out.println(result); // 결과 출력
    }
}
```