3079. 입국심사

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

https://www.acmicpc.net/problem/3079

https://www.acmicpc.net/problem/3079

1. 문제 요약

상근이와 친구들은 오스트레일리아로 여행을 떠났다. 상근이와 친구들은 총 M명이고, 지금 공항에서 한 줄로 서서 입국심사를 기다리고 있다. 입국심사대는 총 N개가 있다. 각 입국심사관이 심사를 하는데 걸리는 시간은 사람마다 모두 다르다. k번 심사대에 앉아있는 심사관이 한 명을 심사를 하는데 드는 시간은 Tk이다.

가장 처음에 모든 심사대는 비어있고, 심사를 할 준비를 모두 끝냈다. 상근이와 친구들은 비행기 하나를 전세내고 놀러갔기 때문에, 지금 심사를 기다리고 있는 사람은 모두 상근이와 친구들이다. 한 심사대에서는 한 번에 한 사람만 심사를 할 수 있다. 가장 앞에 서 있는 사람은 비어있는 심사대가 보이면 거기로 가서 심사를 받을 수 있다. 하지만 항상 이동을 해야 하는 것은 아니다. 더 빠른 심사대의 심사가 끝나길 기다린 다음에 그 곳으로 가서 심사를 받아도 된다.

상근이와 친구들은 모두 컴퓨터 공학과 학생이기 때문에, 어떻게 심사를 받으면 모든 사람이 심사를 받는데 걸리는 시간이 최소가 될지 궁금해졌다.

예를 들어, 두 심사대가 있고, 심사를 하는데 걸리는 시간이 각각 7초와 10초라고 하자. 줄에 서 있는 사람이 6명이라면, 가장 첫 두 사람은 즉시 심사를 받으러 가게 된다. 7초가 되었을 때, 첫 번째 심사대는 비어있게 되고, 세 번째 사람이 그곳으로 이동해서 심사를 받으면 된다. 10초가 되는 순간, 네 번째 사람이 이곳으로 이동해서 심사를 받으면 되고, 14초가 되었을 때는 다섯 번째 사람이 첫 번째 심사대로 이동해서 심사를 받으면 된다. 20초가 되었을 때, 두 번째 심사대가 비어있게 된다. 하지만, 여섯 번째 사람이 그 곳으로 이동하지 않고, 1초를 더 기다린 다음에 첫 번째 심사대로 이동해서 심사를 받으면, 모든 사람이 심사를 받는데 걸리는 시간이 28초가 된다. 만약, 마지막 사람이 1초를 더 기다리지않고, 첫 번째 심사대로 이동하지 않았다면, 모든 사람이 심사를 받는데 걸리는 시간이 30초가 되게 된다.

상근이와 친구들이 심사를 받는데 걸리는 시간의 최솟값을 구하는 프로그램을 작성하시오.

2. 문제 입출력

입력

- 첫째 줄에 N과 M이 주어진다.
- 다음 N개 줄에는 각 심사대에서 심사를 하는데 걸리는 시간인 Tk가 주어진다.

입력예시

```
2 6
7
10
```

출력

- 첫째 줄에 상근이와 친구들이 심사를 마치는데 걸리는 시간의 최솟값을 출력한다.

출력예시

```
28
```

📣 3. 제한사항

- **시간** : 1초
- **메모리** : 128MB

- 1 ≤ N ≤ 100,000, 1 ≤ M ≤ 1,000,000,000
- 1 ≤ Tk ≤ 109

🧐 4. 접근법

▼ 초기 접근법 - 단순 비교 (이분 탐색을 몰랐다고 가정)

- 각 심사대가 심사 진행 중인지 아닌지 알 수 있는 배열 생성, 초기화
- 일단 처음에는 모든 심사대가 비어있기 때문에,
  - M ≤ N 이면 N[0]부터 N[M-1]까지의 합을 구해 출력
  - M > N 이면 N[0]부터 N[M-1]까지의 합 구하고 각 심사대 상태 true로 바꾸고 M-N 한 상태에서 시작
    - M이 0이 될때까지 각 심사대 반복문 돌며 심사 상태 확인
      - 심사가 진행중이면 남은 시간 + 해당 심사대 심사 시간, 끝나서 대기 상태이면 해당 심사대 심사 시간 이 값들을 비교하며 min 값을 찾는다.
      - 심사대를 다 돌았으면 min값을 결과값에 더하고, M -= 1

- left = 0, right = maxTime * M (최악의 경우: 가장 느린 심사관이 모든 사람을 심사할 때)
- while (left <= right)
  - mid = (left + right) / 2;
  - total += mid / times[i];
  - total >= M 이면
    - result = mid; // 가능한 시간 중 최소를 찾음
    - right = mid - 1; // 더 짧은 시간에서 시도
  - 아니면
    - left = mid + 1; // 더 많은 시간이 필요함

5. 코드

▼ 이전 코드 (작성하다가 말았음)

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

public class BoJ_3079 {
    static int N, M;
    static boolean[] isWork;
    static int[] T, aT;
    static int result;
    static int time, mintime;
    static int cnt;

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

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

        isWork = new boolean[N];
        T = new int[M];
        aT = new int[M];

        for (int i = 0; i < N; i++) {
            T[i] = Integer.parseInt(br.readLine());
        }

        result = 0;

        if (M <= N) {
            for (int i = 0; i < M; i++) {
                result += T[i];
            }
        } else {
            for (int i = 0; i < N; i++) {
                result += T[i];
                isWork[i] = true;
            }
            M -= N;

            cnt = 0;

            while (M > 0) {
                int mintime = Integer.MAX_VALUE;
                int time;
                int t;
                for (int i = 0; i < N; i++) {
                    aT[i] -= cnt;
```

```
                    if (aT[i] == 0) {
                        isWork[i] = false;
                    } else {
                        isWork[i] = true;
                    }

                    if (isWork[i]) {
                        if (aT[i] > cnt) {
                            t = aT[i] - cnt;
                        } else {
                            t = aT[i] - cnt % aT[i];
                        }

                        time = t + T[i];
                    } else {
                        time = T[i];
                    }

                    if (time < mintime) {
                        mintime = time;

                    }
                }

                result += mintime;
                System.out.println("mintime : " + mintime);
                M -= 1;
                cnt++;
            }
        }

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

▼ 최종 코드

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

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());
        long M = Long.parseLong(st.nextToken());

        long[] times = new long[N]; // 각 심사대에서 심사하는데 걸리는 시간
        long maxTime = 0;

        for (int i = 0; i < N; i++) {
            times[i] = Long.parseLong(br.readLine());
            if (times[i] > maxTime) {
                maxTime = times[i]; // 가장 긴 심사 시간을 저장
            }
        }
```

```java
        // 이분 탐색 범위 설정
        long left = 0;
        long right = maxTime * M; // 최악의 경우: 가장 느린 심사관이 모든 사람을 심사할 때

        long result = right; // 최소 시간을 저장할 변수

        while (left <= right) {
            long mid = (left + right) / 2; // 중간 시간
            long total = 0;

            // mid 시간 동안 각 심사대에서 처리할 수 있는 사람의 수 계산
            for (int i = 0; i < N; i++) {
                total += mid / times[i];
                if (total >= M) break; // 이미 M명을 넘으면 더 계산할 필요 없음
            }

            if (total >= M) {
                result = mid; // 가능한 시간 중 최소를 찾음
                right = mid - 1; // 더 짧은 시간에서 시도
            } else {
                left = mid + 1; // 더 많은 시간이 필요함
            }
        }

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