boj_2805_유서현

수업일	@2024/02/14
작성일시	@2024년 2월 14일 오후 5:18
최종 편집 일시	@2024년 2월 14일 오후 5:18
복습?	☐

boj_2805_나무자르기

- 느낀점 (인 척 하는 풀이과정)
  - 처음에 그냥 무지성으로 H-1부터 1씩 잘라나가면서 풀면 완전 쉬운 문제 아닌가? 해서 구해보았는데 바로 시간초과가 떠서, 정신차리고 입력값의 범위를 보니 2억까지라서 식겁했다. ~~아니 나무가 어떻게 2억미터까지 자라~~
  - 나와 같은 케이스를 겪는 질문게시판의 글을 참고했고, 수업 때 배웠던 이분 탐색을 복습하고 적용하기로 했다.

    https://www.acmicpc.net/board/view/55886

    이 경우 시간 복잡도는 O(NM)이 되어 최악의 경우 1,000,000,000,000,000번의 연산을 수행해야 할 수도 있습니다. 이분 탐색에 대해 찾아보셔서 이 부분을 최적화 해보시는 것을 추천드립니다.
  - 이분탐색이 아직 헷갈리는 것 같아서 구현을 자주 해봐야 할 것 같다

```
start = mid + 1; // start지점을 11으로 수정 -> 11~19 탐색 (
                 // 이부분을 mid로했었는데, 답이 안 나오고 무한루
                 // mid+1로 수정하니 잘 돌아간다
                 // 아직 이분탐색 이해가 부족한듯
```

  - cutTree 변수가 int범위를 초과할 수도 있어서 long으로 바꿔줘야겠다고 생각해서 바꾸었는데, 전체 변수를 다 long으로 해줘야 하는 건지 아니면 cutTree만 long으로 해줘도 되는건지 헷갈렸다. 일단 후자로 코드를 작성했다.

- 설계 시간: 10분

💡 설계 아이디어

```
//    나무의 수 N 4
//    가져갈 길이 M 7
//    배열에다가 나무길이 N개사이즈로 20 15 10 17
//
//    높이 H설정 : N개 중 최대값 -1 19
//
//    - H를 -- 해가면서 (최초는 19)
//    - 가져갈 나무 길이 cutTree >M이 되면 종료
//    - 배열 반복문에서
//          배열을 돌면서
//          20 - H 해서 그 값을 cutTree에 +=
// 그 때의 H를 출력
```

▼ 수정 전 코드 (2% 시간초과)

```
package boj_2805_나무자르기;

import java.util.Scanner;

public class Main {

//  나무의 수 N 4
//  가져갈 길이 M 7
//  배열에다가 나무길이 N개사이즈로 20 15 10 17
//
//  높이 H설정 : N개 중 최대값 -1 19
//
//  - H를 -- 해가면서 (최초는 19)
//  - 가져갈 나무 길이 cutTree >M이 되면 종료
//  - 배열 반복문에서
//       배열을 돌면서
//       20 - H 해서 그 값을 cutTree에 +=
    // 그 때의 H를 출력
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int N = sc.nextInt(); // 나무의 수 N 4
        int M = sc.nextInt(); // 가져갈 길이 M 7
        int[] trees = new int[N]; // 배열에다가 나무길이 N개사이
        int H = Integer.MIN_VALUE; // 최대값 찾기
        // 높이 H설정 : N개 중 최대값 -1 19
        for (int i = 0; i < N; i++) {
            trees[i] = sc.nextInt();

            if (trees[i] > H) {
                H = trees[i];
            }
        }

        sc.close();

//      - H를 -- 해가면서 (최초는 19)
//      - 가져갈 나무 길이 cutTree >M이 되면 종료
//      - 배열 반복문에서
//          배열을 돌면서
//          20 - H 해서 그 값을 cutTree에 +=
        for (int h = H - 1; h >= 0; h--) {
            long cutTree = 0;

            for (int i = 0; i < N; i++) {
                if (trees[i] - h >= 0) {
                    cutTree += trees[i] - h;
                } // 음수는 안더함
            }

            if (cutTree >= M) {
                System.out.println(h); // 그 때의 H를 출력
                break;
            }
        }
```

```
    } // main

}
```

- 구현 시간: 최초 20분 + 삽질이분탐색 공부 20분 + 최종 15분

```
package boj_2805_나무자르기;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int N = sc.nextInt(); // 나무의 수 N 4
        int M = sc.nextInt(); // 가져갈 길이 M 7
        int[] trees = new int[N]; // 배열에다가 나무길이 N개사이
        int H = Integer.MIN_VALUE; // 최대값 찾기
        // 높이 H설정 : N개 중 최대값 -1 19
        for (int i = 0; i < N; i++) {
            trees[i] = sc.nextInt();

            if (trees[i] > H) {
                H = trees[i];
            }
        }

        sc.close();

        // 이분 탐색 시작
        // 시작범위: 1~H
        int start = 1;
        int end = H - 1; // 19
```

```
        int mid; // 중간값
        long cutTree; // 가져갈 나무

        while (start <= end) { //

            mid = (start + end) / 2; // 10
            // 이 부분 코드를 while문안에서 안하고 int mid; 여기서

            cutTree = 0;

            for (int i = 0; i < N; i++) {
                if (trees[i] > mid)
                    cutTree += (trees[i] - mid);
            }

            if (cutTree < M) { // cutTree가 M보다 모자르면
                end = mid - 1; // end지점을 9로 수정 -> 1~9 탐
            } else { // cutTree가 M보다 같거나 크면
                start = mid + 1; // start지점을 11으로 수정 -
                // 이부분을 mid로했었는데, 답이 안 나오고 무한루프
                // 아직 이분탐색 이해가 부족한듯
            }
        }

        System.out.println(end);
    } // main

}
```