이분탐색 : 나무 자르기

문제

상근이는 나무 M미터가 필요하다. 근처에 나무를 구입할 곳이 모두 망해버렸기 때문에, 정부에 벌목 허가를 요청했다. 정부는 상근이네 집 근처의 나무 한 줄에 대한 벌목 허가를 내주었고, 상근이는 새로 구입한 목재절단기를 이용해서 나무를 구할것이다.

목재절단기는 다음과 같이 동작한다. 먼저, 상근이는 절단기에 높이 H를 지정해야 한다. 높이를 지정하면 톱날이 땅으로부터 H미터 위로 올라간다. 그 다음, 한 줄에 연속해있는 나무를 모두 절단해버린다. 따라서, 높이가 H보다 큰 나무는 H 위의 부분이 잘릴 것이고, 낮은 나무는 잘리지 않을 것이다. 예를 들어, 한 줄에 연속해있는 나무의 높이가 20, 15, 10, 17이라고 하자. 상근이가 높이를 15로 지정했다면, 나무를 자른 뒤의 높이는 15, 15, 10, 15가 될 것이고, 상근이는 길이가 5인 나무와 2인 나무를 들고 집에 갈 것이다. (총 7미터를 집에 들고 간다) 절단기에 설정할 수 있는 높이는 양의 정수 또는 0이다.

상근이는 환경에 매우 관심이 많기 때문에, 나무를 필요한 만큼만 집으로 가져가려고 한다. 이때, 적어도 M미터의 나무를 집에 가져가기 위해서 절단기에 설정할 수 있는 높이의 최댓값을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 나무의 수 N과 상근이가 집으로 가져가려고 하는 나무의 길이 M이 주어진다. (1 ≤ N ≤ 1,000,000, 1 ≤ M ≤ 2,000,000,000)

둘째 줄에는 나무의 높이가 주어진다. 나무의 높이의 합은 항상 M보다 크거나 같기 때문에, 상근이는 집에 필요한 나무를 항상 가져갈 수 있다. 높이는 1,000,000,000보다 작거나 같은 양의 정수 또는 0이다.

출력

적어도 M미터의 나무를 집에 가져가기 위해서 절단기에 설정할 수 있는 높이의 최댓값을 출력한다.

```
예제 입력 1
4 7
20 15 10 17
```

```
예제 출력 1
15
```

```
예제 입력2
5 20
4 42 40 26 46

예제 출력 2
36
```

1. 시간 초과 오류

입력값의 범위가 크기 때문에 완전탐색으로 접근하면 시간 초과 → long long 자료형 사용!!

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{
   int N=0;
   int M;
   cin>>N>>M;

   vector<int> tree;
   int ptr;
   for(int i=0; i<N;i++){
       cin>>ptr;
       tree.push_back(ptr);
   }
   sort(tree.begin(),tree.end());

  int sum;
  int max=tree[N-1];

  for(int i= max;i>=0;i--){
```

```
      sum=0;
      for(int j=0; j<N;j++){
          if(tree[j]-i>0){
            sum+=tree[j]-i;
          }
      }
      if(sum==M){
          cout<<i;
          break;
      }
  }
    return 0;
}
```

2. 이분 탐색으로 풀기

코드상으로는 돌아가지만

But, 제출 시 틀렸다고 나옴..ㅠ

→ begin 값 잘못 설정 (0~최댓값 사이에서 줄여나가야함)

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

void binarySearch(vector<int> tree, int b, int e, long long M
    long long sum=0;
    int begin=b;
    int end= e;
    int middle= (begin+end)/2;

    for(int j=0; j<tree.size();j++){
          if(tree[j]-middle>0){
            sum+=tree[j]-middle;}
      }
    if(begin==end){
```

```
        if(M==sum){
            cout<<middle;
        }
        else{}
        return;
    }

    if(M==sum) cout<<middle;
    else if(M>sum && (middle-1)>=begin){ binarySearch(tree,be
    else if(M<sum && (middle+1)<=end){binarySearch(tree,middl
    else{}
}

int main()
{
   int N=0;
   long long M; // key값값
   cin>>N>>M;

   vector<int> tree;
   int ptr;
   for(int i=0; i<N;i++){
       cin>>ptr;
       tree.push_back(ptr);
   }
   sort(tree.begin(),tree.end());
  binarySearch(tree,tree[0],tree[N-1],M);
    return 0;
}
```

3. 이분 탐색을 while문으로 좀 더 간략화 시켜보자

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
```

```
int n,m;
vector<int> tree;

int main() {
    cin >> n >> m;

    for(auto i =0; i<n;i++) {
        int x;
        cin >> x;
        tree.push_back(x);
    }
    sort(tree.begin(),tree.end());

    int start =0;
    int end= tree[n-1];
    int result=0;

    while(start<=end) {
        long long int total = 0;
        int mid = (start+end) / 2;
        for(auto i =0; i<n;i++) {
            if (tree[i]>mid) total += tree[i] - mid;
        }
        if(total<m) {
            end = mid -1;
        } else {
            result = mid;
            start = mid +1;
        }
    }
    cout << result;
}
```