2024 겨울학기 동국대학교SW역량강화캠프

1일차. 시간 복잡도 & 문제 접근 방법

알고리즘 문제해결을 위한 과정

- 알고리즘들의 난이도는 그렇게 높지 않지만, 그 알고리즘들을 이용해 풀이를 구상하는 논리력이 중요
- 알고리즘 문제를 푸는 과정을 크게 4단계로 나누면 다음과 같이 나눌 수 있음

문제 상황, 문제의 조건과 같은 내용에 대한 이해력이 요구

문제에 대한 다양한 접근 시각, 논리력

구상한 풀이를 단계별로 구체화 시키는 능력과, 프로그래밍 언어에 대한 이해, 실제 구현 능력

자신의 코드에 대한 통찰, 예외 케이스를 찾는 능력을 필요로 합니다.

● 시간복잡도

- ● 모든 코딩테스트들은 문제에 “시간 제한”과 “메모리 제한” 이 존재
- ● 대부분의 경우 시간 제한은 1초 내외, 메모리 제한은 256MB 내외
- ● 본인이 작성할 프로그램의 예상 실행 시간과 사용하는 메모리 양을 확인한 후 코드 작성

▶ 1초 = 연산 1억회

● 구간의 합들 (5302, 사전테스트 1번 문항)

문제

자연수가 N개로 이루어진 수열 $A[1]$ ~ $A[N]$ 이 주어질 때, 아래 조건을 만족하는 구간 $[i, j]$의 수를 출력하여라

$$A[i] + A[i+1] + \ldots + A[j-1] + A[j] = M$$

첫 번째 예제에서는 $[2, 4], [3, 6]$ 구간이 조건을 만족하므로 답이 2이다.

입력

첫 줄에는 수의 개수를 나타내는 정수 N과 구간의 합을 의미하는 정수 M이 주어진다. $1 \leq N \leq 500, 1 \leq M \leq 10^8$
두 번째 줄에는 배열을 구성하는 $3 * 10^4$ 이하의 자연수가 N개 주어진다.

출력

문제의 조건을 만족하는 구간의 수를 출력하자.

●코드를 짜기 전에..

문제를 정확하게 이해했는지 확인

예시 입력과 예시 출력을 활용

직접 계산해서, 정답과 일치하는 지 확인.

예제1 입력 복사

```
6 8
1 3 3 2 2 1
```

예제1 출력 복사

```
2
```

● 문제의 제한 조건 확인

주어진 입력의 범위

출력 조건

주어진 시간 제한(1초)

입력

첫 줄에는 수의 개수를 나타내는 정수 N과 구간의 합을 의미하는 정수 M이 주어진다. $1 \leq N \leq 500, 1 \leq M \leq 10^8$
두 번째 줄에는 배열을 구성하는 $3 * 10^4$ 이하의 자연수가 N개 주어진다.

● 2가지 문제 해결 방법

▶ 1. 구간의 시작점과 끝점을 정하고, 시작점부터 끝점까지의 합을 구해 M과 비교

▶ 2. 구간의 시작점을 정하고, 합이 M 이상이 될 때까지 다음 수를 더하기

핵심 코드

```
int count = 0;
for (int i=0; i<N; i++){ // 시작점 i
    for(int j=i; j<N; j++){ // 끝점 j
        // 구간 [i,j] 검사
        int sum = 0;
        for (int k=i; k<=j; k++){ // sum에 arr[i~j] 의 합 저장
            sum += arr[k];
        }
        if (sum == M){ // 구간 [i,j]의 합이 M과 같다면 경우의 수에 1 추가
            count++;
        }
    }
}
```

핵심 코드

```
int count = 0;
for (int i=0; i<N; i++){ // 시작점 i
    for(int j=i; j<N; j++){ // 끝점 j
        // 구간 [i,j] 검사
        int sum = 0;
        for (int k=i; k<=j; k++){ // sum에 arr[i~j] 의 합 저장
            sum += arr[k];
        }
        if (sum == M){ // 구간 [i,j]의 합이 M과 같다면 경우의 수에 1 추가
            count++;
        }
    }
}
```

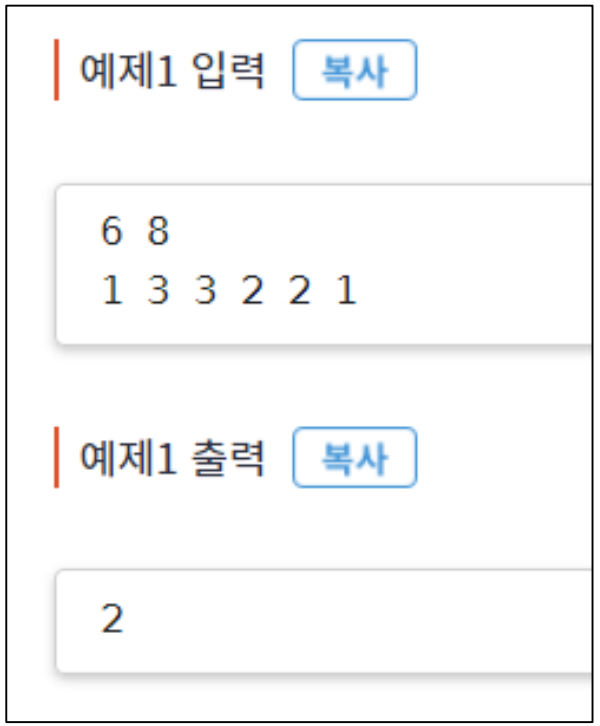
O(N^3) 번 실행된다. 500^3 = 1.25억

● 코드의 실행시간을 최적화하는 3가지 방법

▶ 1. 더 빠르게 처리할 수 있는 과정이 있는가?

▶ 2. 불필요한 정보를 구하기 위해 시간을 낭비하고 있는가?

▶ 3. 이미 알고 있는 정보를 다시 구하기 위해 시간을 낭비하고 있는가?



핵심 코드

```
int count = 0;
for (int i=0; i<N; i++){ // 시작점 i
    // i를 시작점으로 한 구간들 검사
    int sum = 0;
    for(int j=i; j<N; j++) {
        sum += arr[j]; // arr[i,j]의 합 sum
        if(sum == M) count++;
        if(sum > M) break;
    }
}
```

핵심 코드

```
int count = 0;
for (int i=0; i<N; i++){ // 시작점 i
    // i를 시작점으로 한 구간들 검사
    int sum = 0;
    for(int j=i; j<N; j++) {
        sum += arr[j]; // arr[i,j]의 합 sum
        if(sum == M) count++;
        if(sum > M) break;
    }
}
```

O(N^2) 번 실행된다. 500^2 = 0.0025억

● 약수의 합 (3079)

문제

한 정수 n을 입력받는다.
1부터 n의 자연수들 중 n 약수의 합을 구하는 프로그램을 작성하시오.
예를 들어 n이 10이라면 10의 약수는 1, 2, 5, 10이므로 구하고자 하는 값은 1 + 2 + 5 + 10을 더한 18 이 된다.

입력

첫 번째 줄에 정수 n이 입력된다.
(단, 1 <= n <= 10,000,000,000(100억))

출력

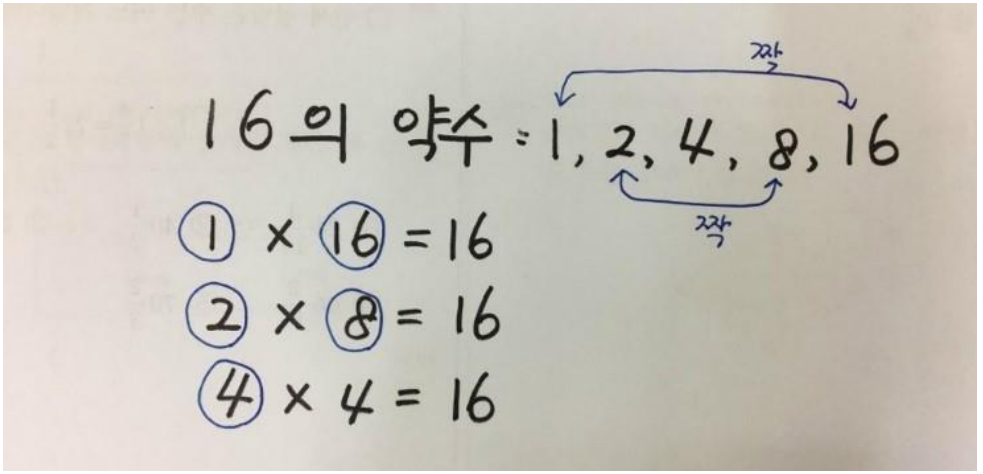
n의 약수의 합을 출력한다.

핵심 코드 O(N)

```
long ans = 0L;
for(long i=1L; i<=N; i++)
{
    if(N % i == 0) ans += i;
}
System.out.println(ans);
```

O(N) 번 실행된다. 100억 = 100초

● 코드의 실행시간을 최적화하는 3가지 방법



▶ 1. 더 빠르게 처리할 수 있는 과정이 있는가?

▶ 2. 불필요한 정보를 구하기 위해 시간을 낭비하고 있는가?

▶ 3. 이미 알고 있는 정보를 다시 구하기 위해 시간을 낭비하고 있는가?

핵심 코드 O(N)

곱하여 N이 되는 두 약수 중 하나는 sqrt(N) 이하이다.

```
long ans = 0L;
for(long i=1L; i*i<=N; i++)
{
    if(N % i == 0) {
        if(i != N/i) ans += i + (N/i);
        else ans += i;
    }
}
```

O(sqrt(N)) 번 실행된다. 10만 = 0.001초

● 코딩테스트 프로그래밍 시 유의사항

- ● 코딩테스트를 위해 프로그래밍을 할 때 고려해야 하는 점이 몇 가지 존재
- ● 입출력이 대표적
- ● 많은 양을 입력 및 출력해야 하는 경우가 많은 코딩테스트 문제들

▶ BufferedReader, StringTokenizer를 이용한 입력, StringBuilder를 이용한 출력

●수 정렬하기 (4666)

문제

N개의 수가 주어질 때, 수들을 오름차순으로 정렬한 결과를 출력하라

입력

첫 줄에 수의 개수 N이 주어진다. $(1 \leq N \leq 1,000,000)$

둘째 줄부터 N개 줄에 걸쳐 수가 주어진다. 주어지는 수는 0 이상 1,000,000 이하의 정수이다.

출력

N개 줄에 걸쳐, 한 줄에 하나씩 수들을 오름차순으로 정렬한 결과를 출력한다.

- 수를 최대 100만개 입력받아 정렬하는 문제
- 빠른 입출력을 사용하지 않으면 입출력만으로 시간초과
- Collections.sort vs Arrays.sort

두 정렬 모두 평균 시간복잡도 O(NlgN)

Arrays.sort는 dual pivot quicksort. 최악의 경우 O(N^2)

핵심 코드 O(Collections.sort)

```
ArrayList<Integer> list = new ArrayList<>();

for(int i =0; i<n; i++){
    int num = Integer.parseInt(br.readLine());
    list.add(num);
}

Collections.sort(list);

for(int i : list) {
    sb.append(i).append('\n');
}
```

● N번째 피보나치 수 구하기 1(5333)

문제

피보나치 수 $Fibo(N)$는 다음과 같이 정의된다.

$$Fibo(1) = 1, Fibo(2) = 1$$
$$Fibo(i) = Fibo(i-1) + Fibo(i-2) \ (i \geq 3)$$

N을 입력받아 N번째 피보나치 수를 출력해보자

입력

첫째 줄에 N이 주어진다. ($N \leq 70$)

출력

$Fibo(N)$을 출력한다.

● 70번째 피보나치 수는?

190경 3924억 9070만 9135

● int 형 변수는 -2^31 ~2^31-1 (21억) 까지 저장 가능

2^63 – 1(900경) 까지 저장 가능한 long 형 변수 사용

핵심 코드

```
        int N = Integer.parseInt(br.readLine());
        long[] fibo = new long[505]; // 70번째 피보나치 수는 int 범위는 2^31 - 1 을 넘는다.
그러나 Long의 범위인 2^63 - 1 은 넘지 않기 때문에 Long으로 선언하여 문제를 해결할 수 있다.
        fibo[1] = 1L; fibo[2] = 1L;
        for(int i=3; i<=N; i++) fibo[i] = fibo[i-1] + fibo[i-2];
        System.out.println(fibo[N]);
```

● 분수 비교하기(5337)

문제

두 분수 a/b와 c/d가 주어졌을 때, 두 수 중 어느 값이 더 큰지 출력해보자.

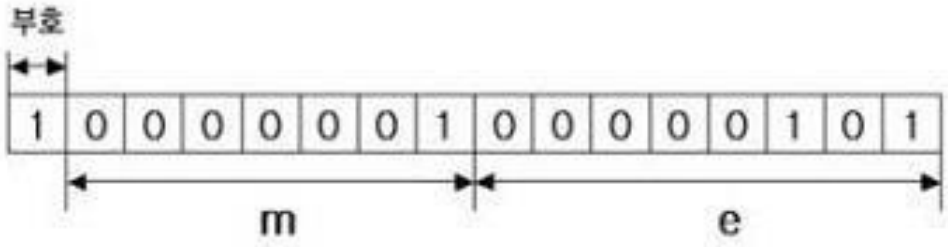
입력

네 자연수 a, b, c, d가 첫 줄에 공백을 사이에 두고 주어진다. $(1 \leq a, b, c, d \leq 10^8)$

출력

a/b가 더 크다면 "A/B" 를 c/d가 더 크다면 "C/D"를, 두 수가 같다면 "EQUALS" 를 첫 줄에 출력한다.

- 컴퓨터가 실수를 저장하는 방법은?



$$\pm(1.m)\times 2^{e-127}$$

- 정확한 값을 저장하지 않는다.

두 실수가 정확하게 같은지 비교하기 위해서는 최대한 정수 연산 활용

핵심 코드

```
long AD = (long)A*D;
long BC = (long)B*C;
if(AD>BC) System.out.println("A/B");
else if(AD<BC) System.out.println("C/D");
else System.out.println("EQUALS");
```