백준 17427 약수합 2

문제: 두 자연수 A와 B가 있을 때, A = BC를 만족하는 자연수 C를 A의 약수라고 한다. 예를 들어, 2의 약수는 1, 2가 있고, 24의 약수는 1, 2, 3, 4, 6, 8, 12, 24가 있다. 자연수 A의 약수의 합은 A의 모든 약수를 더한 값이고, f(A)로 표현한다. x보다 작거나 같은 모든 자연수 y의 f(y)값을 더한 값은 g(x)로 표현한다.

자연수 N이 주어졌을 때, g(N)을 구해보자.

입력: 첫째 줄에 자연수 N(1 ≤ N ≤ 1,000,000)이 주어진다.

출력:

출력

첫째 줄에 g(N)를 출력한다.

예제 입력	예제 출력
예제 입력 1 복사	예제 출력 1 복사
1	1
예제 입력 2 복사	예제 출력 2 복사
2	4
예제 입력 3 복사	예제 출력 3 복사
10	87
예제 입력 4 복사	예제 출력 4 복사
70	4065
예제 입력 5 복사	예제 출력 5 복사
10000	82256014

처음에 문제 이해부터 되지 않아 구글링을 해본 결과 다음과 같았다. 3을 입력할 경우 1의 약수 {1} + 2의 약수 {1,2} + 3의 약수{1+2+3}= 1+3+5가되어 9를 출력하는 것이다. N이 100이면 1부터 100까지 반복이다. 즉 1부터 100까지 약수의 개수 합을 모두 더해야 한다. 하지만 이 과정에서도 만약 5의 약수의 총합을 구할 때 1부터 5까지 전부 검사하

며 약수인지 확인하면 시간 복잡도가 좋지 않을 것이라고 생각했다. 따라서 처음에 36이라면 36/2인 2부터 18까지만 검사해 약수인지 확인하고 1과 36을 더해주는 방식으로 구했다. 이렇게 구해도 되는 이유는 약수는 서로 처음과 끝 부분들이 항상 짝을 이루기에 1xn =2x(n/2)가 짝이 되는 것이다. 하지만 이 과정에서 n이 무수히 크면 시간 복잡도가 좋지 않기 때문에 시간초과가 나왔다. (시간 복잡도 n^2)

```
    }
    else {
        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                sum += i;
            }
        }
        sum += num + 1; // 자기 자신과 1을 더함
    }
```

지피티의 영감을 통해 다음과 같은 방법을 알게 되었다. 배수를 이용하여 1~n까지의 배열에 다 더해주는 것이다. 예를 들어 입력이 6이라 해보자. 1일 때 {1,2,3,4,5,6}인덱스에 모두 1을 더하는 것이다. 1은 모든 수의 약수이기 때문이다. 2일때는 {2,4,6} 즉 2의 배수 인덱스에 2를 더하는 것이다. 3도 마찬가지로 {3,6} 인덱스에 3을 더한다. 4는 {4}에만 4를 더하고 5는 {5}에 6은{6}에 다 더하는 것이다. 마지막에 모든 배열 인덱스 안에 수를 다 더하면 답이 되는 것이다.

```
// 1부터 num까지의 모든 수에 대해 약수의 합을 미리 계산
for (int i = 1; i <= num; i++) {
    for (int j = i; j <= num; j += i) {
        divisorSum[j] += i;
    }
}
```

즉 이는 O(nlogn)이 된다.

이를 구현하고 앞으로의 약수의 개수 문제가 나온다면 배수의 성질을 이용해야 겠다 생각했다. 배수 부분만 확인을 하면 되기에 효율적이기에 꼭 기억해야겠다.