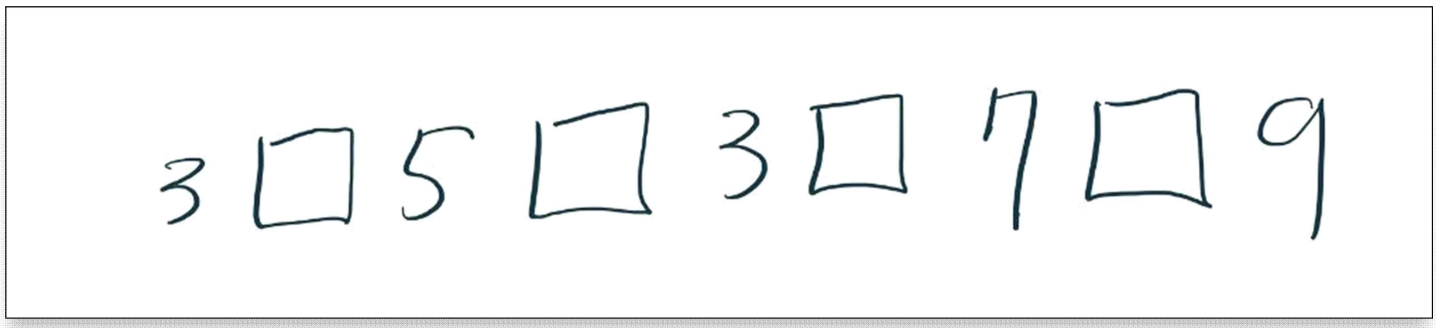
4008. 숫자 만들기 풀이

(두둥탁!!)



1. 우린 빈칸(ㅁ)에 연산자(+, −, * /)를 넣어 **최댓값**과 **최솟값**을 구해야 한다!
(하나의 빈칸에는 4개의 연산자 중에서 하나만 들어 갈 수 있다.)

(ㅁ를 보면 스폰지밥이 생각난다,,,)

재귀호출의 lv은 빈칸(ㅁ)의 개수와 같다!!

우린

lv=1에서 1번째 빈칸(ㅁ)을 채우고,

lv=2에서 2번째 빈칸(ㅁ)을 채우고,

lv=3에서 3번째 빈칸(ㅁ)을 채우고,

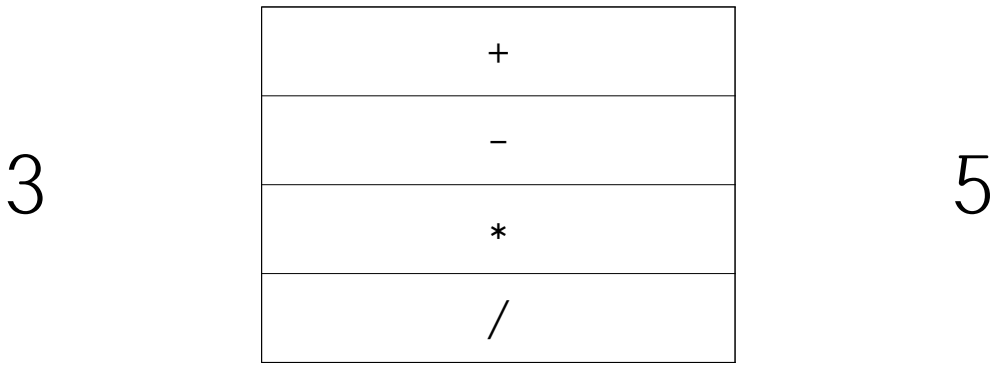
.

.

.

lv=N에서 N번째 빈칸(ㅁ)을 채워야 한다.

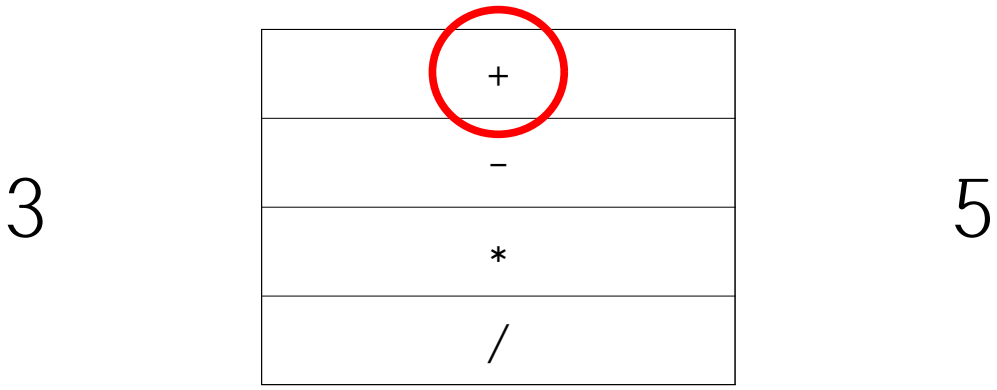
2. 먼저 3ㅁ5를 먼저 보도록 하자.



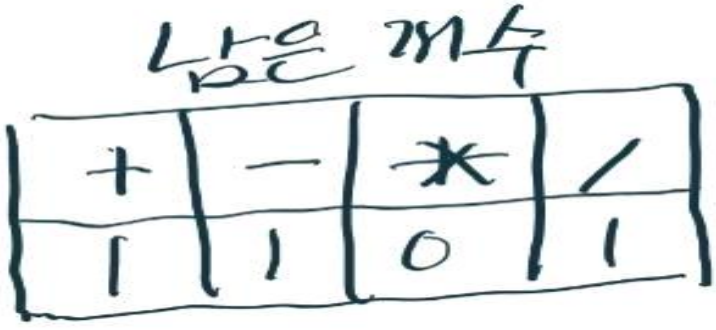
우린 ㅁ에 4가지의 연산자를 넣을 수 있다.
근데 왜 +가 먼저냐고 생각할 수 있다.
단순하다! 입력을 +, −, *, / 순으로 받았기 때문이다.

만약 테스트 케이스가 엄청나게 많다면
연산자의 배치 순서를 바꾸어 백트래킹 하면 좋다!
(최댓값 혹은 최솟값에 빨리 도달 할 수 있다.)

3. 우린 4가지 중에서 한 가지를 선택해야만 한다. +를 선택했다면 +의 개수에 -1 해준다.

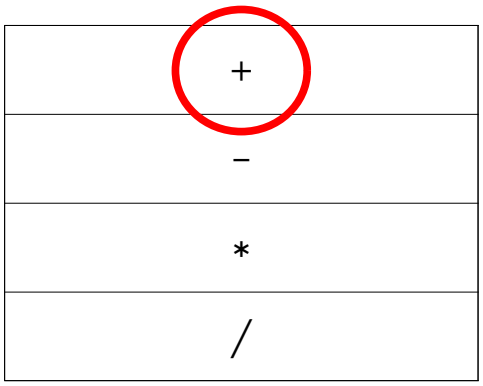


결과 값은 8이다!!
첫 번째 빈칸(ㅁ)을 구했다. lv은 1이다!!



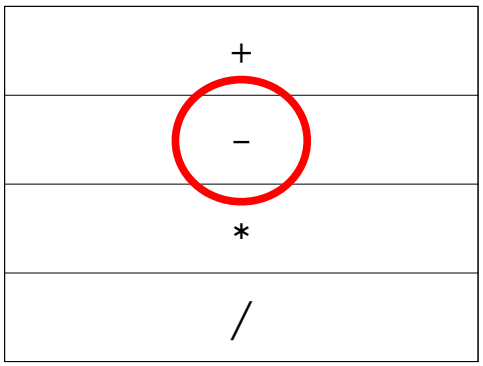
+를 선택했으니 남은 연산자의 개수는 이렇다.
(잘 그렸다!)

5. 이어서 8ㅁ3을 보도록 하자. +의 개수가 남았으니 이어서 +를 선택한다.

8  3

결과 값은 11,,
lv은 2,,

6. 이어서 11ㅁ7을 보자. +의 개수는 0이므로 다음 연산자인 -를 선택한다.

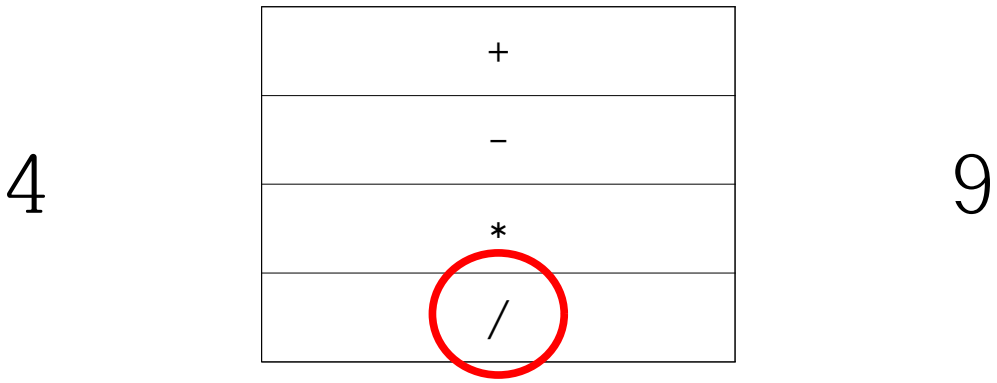
11  7

+의 남은 개수가 0이므로
우린 재귀 함수 for문 안에 if(operList[i] == 0)일 때
continue해줘야 한다.

바로 이 부분이라능~!

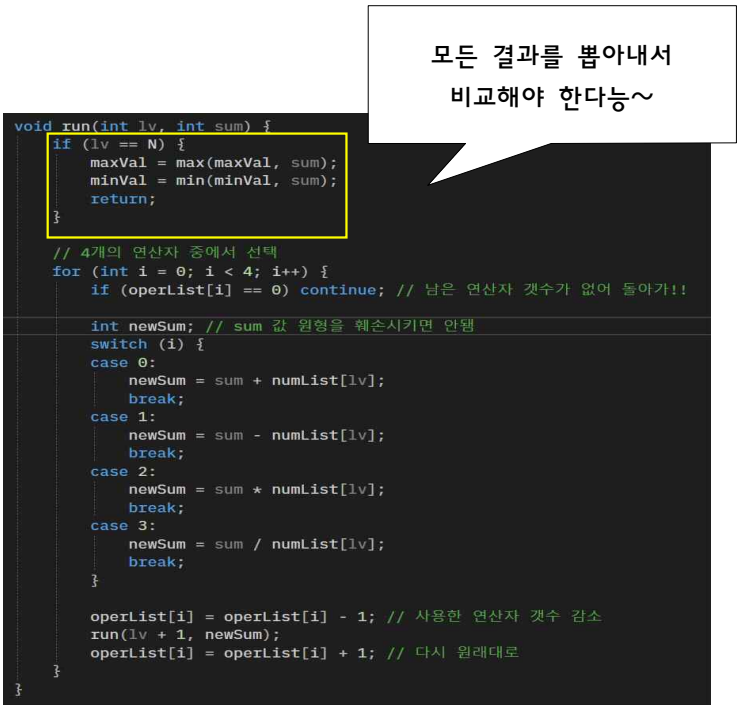
```
void run(int lv, int sum) {
    if (lv == N) {
        maxVal = max(maxVal, sum);
        minVal = min(minVal, sum);
        return;
    }

    // 4개의 연산자 중에서 선택
    for (int i = 0; i < 4; i++) {
        if (operList[i] == 0) continue; // 남은 연산자 갯수가 없어 돌아가!!

        int newSum; // sum 값 원형을 훼손시키면 안됨
        switch (i) {
        case 0:
            newSum = sum + numList[lv];
            break;
        case 1:
            newSum = sum - numList[lv];
            break;
        case 2:
            newSum = sum * numList[lv];
            break;
        case 3:
            newSum = sum / numList[lv];
            break;
        }

        operList[i] = operList[i] - 1; // 사용한 연산자 갯수 감소
        run(lv + 1, newSum);
        operList[i] = operList[i] + 1; // 다시 원래대로
    }
}
```

6. 마지막으로 /를 제외한 나머지 연산자의 개수는 0이므로 다음 연산자로 /를 선택한다.



최종 결과값이 0이 나왔다!!

허나 이것이 최댓값인지 최솟값인지는 아직 알 수 없다..!



lv이 N일 때, 모든 결과값을 서로 비교해야 한다!

변수 maxVal와 minVal를 만들어 모든 결과값과 비교 해보자!

이런 식으로 재귀호출을 반복하면 최댓값과 최솟값을 얻을 수 있다!!

-끗-