백준 16639번 - 괄호 추가하기3

시간제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1.5초	512MB	768	285	221	41.777%

문제

길이가 N인 수식이 있다. 수식은 0보다 크거나 같고, 9보다 작거나 같은 정수와 연산자(+, -, ×)로 이루어져 있다. 곱하기의 연산자 우선순위가 더하기와 빼기보다 높기 때문에, 곱하기를 먼저 계산 해야 한다. 수식을 계산할 때는 왼쪽에서부터 순서대로 계산해야 한다. 예를 들어, 3+8×7-9×2의 결과는 41이다.

수식에 괄호를 추가하면, 괄호 안에 들어있는 식은 먼저 계산해야 한다. 예를 들어, 3+8×7-9×2에 괄호를 (3+8)×7-(9×2)와 같이 추가했으면, 식의 결과는 59가 된다. 중첩된 괄호도 사용할 수 있고, 괄호 안에 여러 개의 연산자가 들어 있어도 된다. 즉, 3+((8×7)-9)×2, 3+((8×7)-(9×2)), (3+8)×(7-9×2) 모두 올바른 식이고, 결과는 97, 41, -121이다.

수식이 주어졌을 때, 괄호를 적절히 추가해 만들 수 있는 식의 결과의 최댓값을 구하는 프로그램을 작성하시오. 추가하는 괄호 개수의 제한은 없으며, 추가하지 않아도 된다.

입력

첫째 줄에 수식의 길이 N(1 ≤ N ≤ 19)가 주어진다. 둘째 줄에는 수식이 주어진다. 수식에 포함된 정수는 모두 0보다 크거나 같고, 9보다 작거나 같다. 문자열은 정수로 시작하고, 연산자와 정수가 번갈아가면서 나온다. 연산자는 +, -, * 중 하나이다. 여기서 *는 곱하기 연산을 나타내는 × 연산이다. 항상 올바른 수식만 주어지기 때문에, N은 홀수이다.

출력

첫째 줄에 괄호를 적절히 추가해서 얻을 수 있는 결과의 최댓값을 출력한다. 정답은 2^{31}보다 작고, -2^{31}보다 크다.

예제 입력1

```
9
3+8*7-9*2
```

예제 출력1

```
136
```

(이하 생략)

출처

출처

알고리즘 분류

- 다이나믹 프로그래밍

접근 방법

문제에서 필요한 정보는 `전체 수식에 괄호를 추가하였을 경우 얻을 수 있는 결과의 최댓값`이므로, 전체의 최적해를 구하기 위하여 어떠한 방식으로 부분 문제들이 최적 부분 구조를 이루는 지 알아본다.

문제에서 주어진 operation은 `+, -, *`로 총 3가지이다.

`[1, n]`까지의 전체 문제는 $[1, k-1] \bullet [k+1, n]$ 의 operation으로 구성된다는 것을 쉽게 알 수 있다. (k는 operator)

각각의 부분 문제를 아래와 같이 정의한다음 메모이제이션하는 것을 통하여 문제를 해결할 수 있다.

`mx[i][j] := 구간 [i][j] 까지 연산에서 가능한 최댓값`

`mn[i][j] := 구간[i][j] 까지 연산에서 가능한 최솟값`

이 때, 최솟값까지 고려하는 이유는 `*`라는 operation이 존재하기 때문이다.

(최소 * 최소 = 최대값의 가능성)

메모이제이션 테이블을 채우기 위하여, 구간의 크기를 $3, 5, \ldots, 2n+1 (n \geq 1)$ 로 잡아준다.

(그 이유는, 홀수 개의 chunk가 operator를 포함한 의미있는 단위이기 때문이다.)

이 후, 구간의 크기에 맞게 i와 j 를 설정한다.

i를 $[0, n-1]$에 존재하는 짝수번째 인덱스로 설정한다면, 구간의 크기에 맞게 j도 설정된다.

k를 홀수번째 인덱스로 설정한다면, k는 항상 수식에서 operator를 가리키게 된다.

여기까지 해결했다면, 아래의 점화식을 통하여 문제를 해결할 수 있다.

> $mx[i][j] := mx[i][k-1] \bullet mx[k+1][j]$, $\bullet$은 연산자, k는 연산자의 인덱스
>
> $mn[i][j] := mn[i][k-1] \bullet mn[k+1][j]$, $\bullet$은 연산자, k는 연산자의 인덱스

소스코드

```cpp
#define FASTIO cin.tie(0)->sync_with_stdio(false), cout.tie(0)
////////////////////////////////////////////////////////////////////
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int N;
ll mx[25][25], mn[25][25];
string s;
ll calc(ll a, char op, ll b){
    if(op == '+') return a + b;
    if(op == '-') return a - b;
    if(op == '*') return a * b;
}
int main(void){
    FASTIO;
////////////////////////////////////////////////////////////////////
    for(int i=0; i<25; i++){
        for(int j=0; j<25; j++){
            mx[i][j] = -3e9;
            mn[i][j] = 3e9;
        }
    }

    cin >> N >> s;

    for(int i=0; i<N; i++) {
        if(i % 2 == 0)
            mx[i][i] = mn[i][i] = s[i] - '0';
    }

    // size = 3, 5, ... 2n + 1
    for(int sz=3; sz <= N; sz += 2){
        for(int i=0; i<N; i+=2){
            int j = i + sz - 1;
            if(j >= N) continue;
            // partition [i, k-1], [k+1, j]
            // k is operator
            for(int k=i+1; k<j; k+=2){
                mx[i][j] = max(mx[i][j], calc(mx[i][k-1], s[k], mx[k+1][j]));
                mx[i][j] = max(mx[i][j], calc(mx[i][k-1], s[k], mn[k+1][j]));
                mx[i][j] = max(mx[i][j], calc(mn[i][k-1], s[k], mx[k+1][j]));
                mx[i][j] = max(mx[i][j], calc(mn[i][k-1], s[k], mn[k+1][j]));
```

```
43
44                  mn[i][j] = min(mn[i][j], calc(mx[i][k-1], s[k], mx[k+1][j]));
45                  mn[i][j] = min(mn[i][j], calc(mx[i][k-1], s[k], mn[k+1][j]));
46                  mn[i][j] = min(mn[i][j], calc(mn[i][k-1], s[k], mx[k+1][j]));
47                  mn[i][j] = min(mn[i][j], calc(mn[i][k-1], s[k], mn[k+1][j]));
48              }
49          }
50      }
51
52      cout << mx[0][N-1] << '\n';
53      return 0;
54  }
55
```