# BASIC CALCULATOR FROM LEETCODE 

CS3201 Algorithm Design Term Project
Semester 2/2020


## PROBLEM

Given a string s representing an expression, implement a basic calculator to evaluate it.

```
Example 1:
Input: s = "1 + 1"
Output: 2
```

Example 2:
Input: $s=" 2-1+2$ "
Output: 3

Example 3:
Input: $s="(1+(4+5+2)-3)+(6+8) "$
Output: 23

## ANALYSIS

According to the title information, s contains the following types of data:

- Left bracket: (
- Right bracket: )
- Plus sign: +
- Minus sign: -
- Non-negative integer
- Space



## ANALYSIS

## Idea

Using the example of this topic " $(1+(4+5+2)-3)+(6+8)$ ", we can split it into:

- $6+8=14$
- $4+5+2=11$
- $11-3=8$
- $1+8=9$
- $9+14=23$


## ANALYSIS

## Idea

For example, first calculate $6+8$ equal to 14 according to the above algorithm, then replace 14 with the original $6+8$, then the original question becomes $(1+(4+5+2)-3)+14$.
$(1+(4+5+2)-3)+(6+8)$
$(1+(4+5+2)-3)+(14)$
$(1+(11)-3)+(14)$
$(1+8)+(14)$
$(9)+(14)$
23


## CODE

class Solution:
def calculate(self, s: str) -> int:
def calculate2(s, start):
stack = [ ]
s += '\$'
position = '+'
number $=0$
i = start
while i < len(s):
$\mathrm{j}=\mathrm{s}[\mathrm{i}]$
if $j==$ ' ':
i += 1
continue
elif j == '(':
i, number $=$ calculate2 $(\mathrm{s}, \mathrm{i}+1)$
elif j.isdigit():
number $=$ number $* 10+i n t(j)$
else:
if position == '+':
stack.append(number)
elif position == '-':
stack.append(-number)
if $\mathrm{j}=$ ' $^{\prime}$ ':
break
position $=j$
number $=0$
i $+=1$
print(sum(stack))
return i, sum(stack)
return calculate2(s, 0)[1]

## COMPLEXITY

Time complexity: $\mathrm{O}(\mathrm{n})$ Where n is the length of the given string expression

Space complexity: O(n)

## References

https://leetcode.com/problems/basic-calculator/


