Asymptotic Analysis

در برنامه برای محاسبه یک عبارت (بدون چاپ مراحل آن) ما دو تابع داریم . یکی تبدیل infix به postfix و یکی محاسبه postfix . برای بدست آوردن پیچیدگی محاسبه یک عبارت باید پیچیدگی این دو تابع را بدست آورده و با هم جمع کنیم .

1- infixtoPostfix Method :

```
174  list < string > infixtoPostfix(string infix) {
175      /*----------------------------Main Function----------------------------*/
176
177      Stack < char > S(infix.length());
178      list < string > postfix;
179      string temp;
180
181      S.push('(');
182      infix += ')';
183
184      for (int i = 0; i ≤ infix.length(); i++) {     comparison of integers of
185
186          if (infix[i] == '(') { ... }
189          } else if (isOperand(infix[i]) || infix[i] == '.') { ... }
192          } else if (isOperator(infix[i])) {
193              if (infix[i] == '-') { ...} }
214
215              temp = "";
216              while (!S.isEmpty() && checkTwoOperators(S.top(), infix[i])) { .
222              S.push(infix[i]);
223          } else if (infix[i] == ')') {
224              temp = "";
225              while (!S.isEmpty() && S.top() != '(') {
226                  string tempChar = "";
227                  tempChar += S.top();
228                  postfix.push_back(tempChar);
229                  S.pop();
230              }
231              S.pop();
232          }
233      }
234      return postfix;                    Potential leak of memory pointed
235
236  }
```

در این تابع ما یک حلقه for داریم که کل رشته infix را پیمایش میکند . سپس در درون این حلقه یک حلقه while داریم که برای قرار دادن اپراتور در درون استک استفاده میشود .

اگر بدترین حالت ممکن را در نظر بگیریم با توجه به این حلقه های تو در تو ، پیچیدگی زمانی این تابع برابر $O(N^2)$ خواهد شد . این به ندرت پیش می آید زیرا در اکثر مواقع سایز استک و میزان پیمایش حلقه while آنقد کم است که میتوان آن را $O(1)$ در نظر گرفت که در این صورت پیچیدگی کل تابع برابر $\mathbf{O(N)}$ خواهد شد

2- evaluatePostfix Method :

```
255 double evaluatePos(list < string > postfix) {
256     /*---------------------------Main Function----------------------------*/
257     Stack < double > stack = Stack < double > (postfix.size());
258
259     for (const auto & x: postfix) {
260         if (isOperator(x.c_str())) {
261             double val1 = stack.top();
262             stack.pop();
263             double val2 = stack.top();
264             stack.pop();
265             try {
266                 stack.push(operation(val1, val2, * x.c_str()));
267             } catch (string a) {
268                 throw std::move(a);
269             }
270         } else {
271             stack.push(stod(x));
272         }
273     }
274     return stack.top();                    Potential leak of memory pointed to by '
275 }
```

در این تابع با توجه به اینکه یک حلقه for داریم که کل رشته postfix را پیمایش میکند و همچنین توابع مربوط به استک نیز دارای پیچیدگی $O(1)$ هستند ، پس پیچیدگی کل این تابع برابر با $\mathbf{O(N)}$ میشود .

در نهایت با توجه به این دو تابع ، پیچیدگی کل محاسبه عبارت برابر $O(N) + O(N)$ میشود که در تحلیل کلان ، همان $\boldsymbol{O(N)}$ در نظر میگیریم.