2022.12.23 스터디 자료

❄️ 내가 푼 문제

- LeetCode 008. String to Integer (atoi)

❄️ 문제 설명

문자열을 부호 있는 32비트 정수로 변환하는 myAtoi(string s) 함수를 구현합니다.

알고리즘은 다음과 같다 :

1️⃣ 선행 공백은 읽고 무시.

2️⃣ 문자열이 끝나지 않은 경우, 다음 문자가 '-' 또는 '+'인지 확인, 둘 중 하나인 경우 이 문자를 읽음.

이것은 음수인지 양수인지를 결정. 둘 다 존재하지 않으면 결과가 양성이라고 가정.

3️⃣ 숫자가 아닌 다음 문자 또는 끝에 도달할 때까지 다음 문자를 읽음. 나머지 문자열은 무시.

4️⃣ 정수로 변환(예시: "123" -> 123, "0032" -> 32). 숫자를 읽지 못한 경우, 0.

필요에 따라 부호를 변경(2번 참조).

5️⃣ 값이 정수 범위[-2^31, 2^31 - 1]를 벗어나는 경우 범위에 남도록 설정.

(예시 : -2^31(Integer.MIN_VALUE)보다 작은 수 -> -2^31(Integer.MIN_VALUE),

2^31 - 1(Integer.MAX_VALUE)보다 큰 수 -> 2^31 - 1(Integer.MAX_VALUE))

6️⃣ 정수로 결과 반환.

Note:

공백 문자 ' '만 공백 문자.

선행 공백 혹은 숫자 뒤의 나머지 문자열 이외의 문자는 무시하지 말 것.

❄️ 방법

- 처음 : 13 ms
- 이후 : 5 ms

```
strip(): 문자열 앞, 뒤의 공백 제거.
stripLeading(): 문자열 앞의 공백 제거.
stripTrailing(): 문자열 뒤의 공백 제거.
```

```
public static int myAtoi(String s) {
  /* Java 11부터 생긴 stripLeading()을 이용하여 문자열 앞의 공백을 제거 */
  s = s.stripLeading().split(" ")[0];

  /*문자열의 길이가 0이면, 0을 리턴 */
  if (s.length() == 0) {return 0;}

  /*반환에 도움을 줄 결과 문자열 선언 및 초기화 */
  String result = "";

  /*
   * 문자열 처음에 -나 +가 오는 경우,음수 양수 판별을 위해
   * 문자열 맨 처음 문자에만 판단
   * 그 이후 문자에서는 무시
   * isPositive ->양수면 true,음수면 false
   * 만약 맨 처음 문자가 +,-그리고 숫자가 아니면 0을 리턴
   */
  char ch = s.charAt(0);
  boolean isPositive = true;
  if (ch == '-') {
      isPositive = false;
      s = s.substring(1);
   } else if (ch == '+') {
      s = s.substring(1);
   } else if(!Character.isDigit(ch)) {
      return 0;
   }

  /* 반복문을 돌면서 문자열에서 숫자를 읽기,조건식에 문자가 숫자일 때를 추가 */
  for (int i = 0; i < s.length() && Character.isDigit(s.charAt(i)); i++) {
      result += String.valueOf(s.charAt(i));

      /* 정수형의 범위를 넘어설 경우,음수이면 최소 정수를 양수이면 최대 정수를 리턴 */
      if (Long.valueOf(result) > Integer.MAX_VALUE||
          Long.valueOf(result) < Integer.MIN_VALUE) {
         return isPositive? Integer.MAX_VALUE: Integer.MIN_VALUE;
      }
   }

  /*만약에 결과를 반환할 문자열의 길이가 0이면, 0을 리턴 */
  if (result.length() == 0) {
      return 0;
   }
```

```
  /*양수면 그대로 정수 값 출력,음수이면 -1을 곱해 음수 출력 */
  return isPositive? Integer.valueOf(result) : Integer.valueOf(result) * -1;
}
```

❄️ 다른 사람이 푼 방법

☝🏻 13ms

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
class Solution {
   public int myAtoi(String s) {
      while(s.startsWith(" ")){
            s = s.substring(1);
        }

      String pattern = "^[+-]?[0-9]+";
      int answer = 0;
      Pattern pattern1 = Pattern.compile(pattern);
      Matcher m = pattern1.matcher(s);

      if(m.find()){
          try{
              answer = Integer.parseInt(m.group());
          }catch(Exception e){
              if(s.startsWith("-")){
                  return Integer.MIN_VALUE;
              }else{
                  return Integer.MAX_VALUE;
              }
          }
      }
      return answer;
  }
}
```

- 23 ms

```
class Solution {
    public int myAtoi(String s) {
      String str = s.stripLeading().split("\\s+")[0];

        if (str.length() == 0) {
            return 0;
        } else if (str.charAt(0) == '+') {
            return convert(str.substring(1), false);
        } else if (str.charAt(0) == '-') {
            return convert(str.substring(1), true);
```

```
        } else if (Character.isDigit(str.charAt(0))) {
            return convert(str, false);
        } else {
            return 0;
        }
    }
    public static int convert(String s, boolean isNegative) {
        long value = 0;

        for (int idx = 0; idx < s.length() && Character.isDigit(s.charAt(idx)); idx++) {
            value = value * 10 + Character.getNumericValue(s.charAt(idx));

            if (value > Integer.MAX_VALUE) {
                return isNegative ? Integer.MIN_VALUE : Integer.MAX_VALUE;
            }
        }

        return (int) (isNegative ? -value : value);
    }
}
```