큰수 만들기

2022-02-18

<https://programmers.co.kr/learn/courses/30/lessons/42883>

키워드: 그리디 알고리즘

## 풀이

그리디 알고리즘 문제는, 주어진 입력을 순서대로 읽어가며 문제는 풀어야 한다.

O(N) 시간으로 문제는 푸는 규칙을 고안해 내야 한다.

예로 주어진 입력을 보면서 규칙을 고안하자.

"1924"

1을 읽은 후 9를 읽을 때, 앞의 1을 버려야 한다.

2를 읽은 후 4를 읽을 때, 앞의 2를 버려야 한다.

즉 큰 수를 읽으면, 앞에서 읽은 작은 수를 버려야 한다.

"1231234"

2를 읽은 후 앞의 1을 버리고

3을 읽은 후 앞의 2를 버린다.

1을 읽고

그 다음 2를 읽은 후 앞의 1을 버린다.

3개를 버렸으니, 남은 "3234"를 리턴

"4177252841"

4를 읽고, 1을 읽고,

4을 읽은 후, 앞의 1을 버린고, 또 그 앞의 4도 버려야 한다.

즉 큰 수를 읽으면,

앞에서 읽은 작은 수들을 계속 버려야 한다.

버린 갯수가 아직 k 보다 작을 때까지

위 규칙이면 충분한 듯.

구현하자.

## 구현 #1

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | package programmers.e4883;  public class Main1 {  static class Solution {  public String solution(String number, int k) {  char[] A = new char[number.length()];  int index = 0;  for (int i = 0; i < number.length(); ++i) {  char ch = number.charAt(i);  while (k > 0 && index > 0 && A[index - 1] < ch) {  --index;  --k;  }  A[index++] = ch;  }  return new String(A, 0, index);  }  }  public static void main(String[] args) {  var sol = new Solution();  System.out.println(sol.solution("1924", 2));  System.out.println(sol.solution("1231234", 3));  System.out.println(sol.solution("4177252841", 4));  }  } |

// 버리고 남은 문자들을 담을 배열

char[] A = new char[number.length()];

// A 배열에 채워진 문자들 뒤 빈칸의 인덱스

int index = 0;

// number 문자열의 문자 각각에 대해서

for (int i = 0; i < number.length(); ++i) {

char ch = number.charAt(i);

// 아직 k 개의 문자를 제거하지 못했고

// 앞에서 읽은 문자가 아직 남아있고

// 그 문자가 ch 문자보다 작을 때까지

while (k > 0 && index > 0 && A[index - 1] < ch) {

--index; // 앞의 문자를 버린다

--k; // 더 버려야 할 문자수 1 감소

}

A[index++] = ch; // ch 문제를 A 배열에 채운다.

}

// A 배열의 선두부터, index 개의 문자로 구성된 문자열 생성하여 리턴

return new String(A, 0, index);

### 오류!

몇몇 테스트 통과 못함.

## 답

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | package programmers.e4883;  public class Main2 {  static class Solution {  public String solution(String number, int k) {  char[] A = new char[number.length()];  int index = 0;  for (int i = 0; i < number.length(); ++i) {  char ch = number.charAt(i);  while (k > 0 && index > 0 && A[index - 1] < ch) {  --index;  --k;  }  A[index++] = ch;  }  if (k > 0) index -= k;  return new String(A, 0, index);  }  }  public static void main(String[] args) {  var sol = new Solution();  System.out.println(sol.solution("1924", 2));  System.out.println(sol.solution("1231234", 3));  System.out.println(sol.solution("4177252841", 4));  System.out.println(sol.solution("11", 1));  }  } |

디버깅 하는 좋은 방법은, 오답이 나올 때까지 간단한 테스트 케이스를 추가해서 돌려 보는 것이다.

"11" 문자열에서 1 개를 제거하는 테스트 케이스를 시도하자 마자

바로 오답이 출력되었다.

정답은 "1" 인데, "11"이 출력되었다.

이 버그를 해결하기 위해서, (줄16) 코드를 추가함.

마지막에 리턴하기 전에, k 값이 양수이면, 마지막 k개의 문제를 버려야 한다.

"111111", "22222" 이런 패턴의 입력일 경우에, (줄16)에서 k가 양수가 된다.