조이스틱

2022-02-18

<https://programmers.co.kr/learn/courses/30/lessons/42860>

키워드: 그리디 알고리즘

## 풀이

"AAA"를 "JAN"로 바꾸기 위해 필요한 조작 횟수와

"JAN"를 "AAA"로 바꾸기 위해 필요한 조작 횟수가 동일하다.

따라서 입력 문자열의 문자를 전부 "A"로 바꾸기 위해 필요한 조작 횟수를 구하자.

### 알파벳 바꾸기 조작 횟수

조이스틱을 상하로 조작하여 알파벳을 바꾸는 회수 계산은 쉽다.

위로 조작하는 경우 (ch - 'A') 이거나

아래로 조작하는 경우 ('Z' - ch + 1) 이다.

위 두 식중의 최소값을 선택하면 된다.

입력 문자열의 문자 각각에 대해서 위 최소값의 합을 구하자.

### 이동 횟수

"JAAN" 문자열에서 오른쪽 방향으로 이동하면 J에서 N까지 3번 이동해야 하지만

왼쪽으로 이동하면 1번 이동하면 된다.

오른쪽으로 이동하는 경우 어디까지 가야하는가?

"ZZZAA"

노란색으로 표시한 부분까지만 가면 된다.

오른쪽으로 두 칸 이동하면 된다.

연두색 문자의 위치를 찾아야 한다.

왼쪽으로 이동하는 경우 어디까지 가야 하나

"ZAAZZ"

왼쪽으로 두 칸 이동하면 된다.

연두색 문자의 위치를 찾아야 한다.

## 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  27  28 | package programmers.e42860;  public class Main1 {  static class Solution {  public int solution(String name) {  final int N = name.length();  int count = 0;  for (char ch : name.toCharArray())  count += Math.min(ch - 'A', 'Z' - ch + 1);  int index1 = N - 1;  while (index1 > 0 && name.charAt(index1) == 'A')  --index1;  int index2 = 1;  while (index2 < N && name.charAt(index2) == 'A')  ++index2;  count += Math.min(index1, N - index2);  return count;  }  }  public static void main(String[] args) {  var sol = new Solution();  System.out.println(sol.solution("JEROEN"));  System.out.println(sol.solution("JAN"));  System.out.println(sol.solution("ZZAAZ"));  }  } |

(줄11~13)

오른쪽으로 이동하는 경우 어디까지 가야하는가?

"ZZZAA"

노란색으로 표시한 부분까지만 가면 된다.

오른쪽으로 두 칸 이동하면 된다.

연두색 문자의 위치 찾기

오른쪽 이동 횟수 == index1

(줄14~16)

왼쪽으로 이동하는 경우 어디까지 가야 하나

"ZAAZZ"

왼쪽으로 두 칸 이동하면 된다.

연두색 문자의 위치 찾기

왼쪽 이동 횟수 == N - index2

위의 두 이동 횟수 중에서 작은 값을 선택하면 된다.

count += Math.min(index1, N - index2);

### 오류!

몇몇 테스트 통과 못함.

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

"ZZAAZ" 에서 오류가 발견되었다.

세 개의 Z 문자를 A로 만들기 위한 조작 = 3회

왼쪽으로 한 칸 갔다가, 다시 오른쪽으로 두 칸 이동 = 3회

답은 6회이어야 한다.

그런데 좌우로 왔다 갔다 하는 이동을 구현하지 않아서, 오른쪽으로 4칸 이동을 선택해서 7회가 출력된다.

## 답 #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  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | package programmers.e42860;  public class Main2 {  static class Solution {  public int solution(String name) {  final int N = name.length();  int count1 = 0;  for (char ch : name.toCharArray())  count1 += Math.min(ch - 'A', 'Z' - ch + 1);  int index1 = N - 1;  while (index1 > 0 && name.charAt(index1) == 'A')  --index1;  int index2 = 1;  while (index2 < N && name.charAt(index2) == 'A')  ++index2;  int count2 = Math.min(index1, N - index2);  for (int i = 0; i < N; ++i) {  int index3 = i + 1;  while (index3 < N && name.charAt(index3) == 'A')  ++index3;  count2 = Math.min(count2, i \* 2 + (N - index3));  count2 = Math.min(count2, i + (N - index3) \* 2);  }  return count1 + count2;  }  }  public static void main(String[] args) {  var sol = new Solution();  System.out.println(sol.solution("JEROEN"));  System.out.println(sol.solution("JAN"));  System.out.println(sol.solution("A"));  System.out.println(sol.solution("B"));  System.out.println(sol.solution("Z"));  System.out.println(sol.solution("ZAZ"));  System.out.println(sol.solution("ZZAAZ"));  }  } |

다음 페이지...

// 줄17까지 실행하면, count2에는 한 방향으로만 이동할 경우의 최소 이동 횟수가 들어있다.

// 좌우로 왔다 갔다하는 이동 횟수를 구한다.

// 오른쪽으로 i 위치까지 이동, 왼쪽으로 돌아서 index3 위치까지 이동

for (int i = 0; i < N; ++i) { // 가능한 모든 i 위치에 대해서

// 왼쪽으로 돌아서 거꾸로 이동한 index3 위치를 찾자

// index3의 기본 값은 거꾸로 한 바퀴 돌아서 i 바로 다음 칸까지 이동해야 한다.

int index3 = i + 1;

// 그런데 만약 i 다음 칸이 'A' 문자이면, 'A' 문자 칸까지 올 필요는 없고

// 'A' 문자가 아닌 칸 까지만 오면 된다.

// 그 index3을 찾자.

while (index3 < N && name.charAt(index3) == 'A')

++index3;

// 오른쪽으로 i 위치까지 왔다가, 다시 왼쪽으로 한 바퀴 돌아서 index3 까지 가자

// i + i + N - index3

// 이 이동 횟수가 더 작다면, 이 이동 횟수를 count2 값에 대입한다.

count2 = Math.min(count2, i \* 2 + (N - index3));

// 왼쪽으로 한 바퀴 돌아서 index3 까지 갔다가 다시 오른쪽으로 i까지 가자.

// N - index3 + N - index + i

// 이 이동 횟수가 더 작다면, 이 이동 횟수를 count2 값에 대입한다.

count2 = Math.min(count2, i + (N - index3) \* 2);

}

### 군더더기 코드 #1

int index1 = N - 1;

while (index1 > 0 && name.charAt(index1) == 'A')

--index1;

index1 횟수 이동

위 이동 횟수 계산은 아래 계산과 결과가 같다.

"ZZZZAAAA"

i 값이 3일때, index3은 N 이다.

count2 = Math.min(count2, i + (N - index3) \* 2);

이동 횟수

### 군더더기 코드 #2

int index2 = 1;

while (index2 < N && name.charAt(index2) == 'A')

++index2;

N - index2 횟수 이동

위 이동 횟수 계산은 아래 계산과 결과가 같다.

"ZAAAAZZZ"

i 값이 0일때, index3은 5 이다.

count2 = Math.min(count2, i \* 2 + (N - index3));

이동 횟수

### 요약

int index1 = N - 1;

while (index1 > 0 && name.charAt(index1) == 'A')

--index1;

int index2 = 1;

while (index2 < N && name.charAt(index2) == 'A')

++index2;

int count2 = Math.min(index1, N - index2);

위 계산은 아래 계산에 포함된다. 따라서 위 계산은 군더더기 코드이다.

for (int i = 0; i < N; ++i) {

int index3 = i + 1;

while (index3 < N && name.charAt(index3) == 'A')

++index3;

count2 = Math.min(count2, i \* 2 + (N - index3));

count2 = Math.min(count2, i + (N - index3) \* 2);

}

## 답 #2

|  |  |
| --- | --- |
| 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  29  30  31  32  33  34 | package programmers.e42860;  public class Main3 {  static class Solution {  public int solution(String name) {  final int N = name.length();  int count1 = 0;  for (char ch : name.toCharArray())  count1 += Math.min(ch - 'A', 'Z' - ch + 1);  int count2 = N;  for (int i = 0; i < N; ++i) {  int index = i + 1;  while (index < N && name.charAt(index) == 'A')  ++index;  count2 = Math.min(count2, i \* 2 + (N - index));  count2 = Math.min(count2, i + (N - index) \* 2);  }  return count1 + count2;  }  }  public static void main(String[] args) {  var sol = new Solution();  System.out.println(sol.solution("JEROEN"));  System.out.println(sol.solution("JAN"));  System.out.println(sol.solution("A"));  System.out.println(sol.solution("B"));  System.out.println(sol.solution("Z"));  System.out.println(sol.solution("ZAZ"));  System.out.println(sol.solution("ZAAAAZ"));  }  } |