스타수열

2022-01-11

Level 3

<https://programmers.co.kr/learn/courses/30/lessons/70130>

## 설계

### 가능한 모든 부분수열 각각에 대해

가능한 모든 부분수열 각각에 대해서

그 부분수열이 스타수열인지 검사하고

스타수열이라면 그 길이의 최대값을 구한다.

이렇게 풀어야 하는 문제가 자주 나온다.

그렇데 이 문제는 이렇게 풀 수 없다.

주어진 수열의 길이가 500,000 이므로, 가능한 모든 부분수열의 수가 너무나 많다.

### 공통 수

스타 수열의 각각의 두 쌍의 수 집합에는 공통 수가 한 개 이상이어야 한다.

### 가능한 공통 수 각각에 대해

주어진 수열에 등장하는 수들의 출현 빈도 수를 구한다. O(N)

등장 수들을 출현 빈도 수 내림차순으로 정렬한다.

그 등장 수 각각에 대해서

그 등장 수가 공통 수인, 가능한 모든 부분수열 각각에 대해서

그 부분수열이 스타수열인지 검사하고

스타수열이라면 그 길이의 최대값을 구한다.

효율 높이기 전략

지금까지 찾은 스타수열 길이 최대값 보다, (출현 빈도 수 \* 2)가 작은 공통 수는 시도해 볼 필요가 없다.

공통 수의 (출현 빈도 수 \* 2) 보다 그 공통 수의 스타 수열 길이기 클 수 없기 때문이다.

## 풀이 #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  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | package programmers.c30.e70130;  import java.util.ArrayList;  import java.util.Collections;  public class Main2 {  static class Solution {  static class Entry {  int number, count;  public Entry(int number, int count) {  this.number = number;  this.count = count;  }  }  static int 최대길이 = 0;  static void 스타수열찾기(int[] A, int index, int 공통수, int 길이) {  int i = index;  while (i < A.length && A[i] != 공통수) // index 위치 뒤의 공통 수 찾기  ++i;  if (i >= A.length) { // 더 이상 공통 수가 없다면, 스타수열 한 개 완성  if (길이 > 최대길이) 최대길이 = 길이; // 최대 값을 구한다.  return;  }  if (i > index && A[i - 1] != A[i]) // {i - 1, i} 한쌍을 수열에 추가하고  스타수열찾기(A, i + 1, 공통수, 길이 + 2); // 다음 쌍 찾기 재귀호출  if (i + 1 < A.length && A[i] != A[i + 1]) // {i, i + 1} 한쌍을 수열에 추가하고  스타수열찾기(A, i + 2, 공통수, 길이 + 2); // 다음 쌍 찾기 재귀호출  스타수열찾기(A, i + 1, 공통수, 길이); // 현재 위치 수를 무시하고 다음 쌍 찾기 재귀호출  }  public int solution(int[] A) {  최대길이 = 0; // 구할 답  // 출현 빈도 수 구하기  int[] count = new int[A.length];  for (int i : A) count[i]++;  // 출현 빈도 수 내림차순 정렬  ArrayList<Entry> list = new ArrayList<>();  for (int i = 0; i < count.length; ++i)  if (count[i] > 0)  list.add(new Entry(i, count[i]));  Collections.sort(list, (a, b) -> b.count - a.count);  // 공통 수 각각에 대해서, 스타수열 찾기  for (Entry entry : list) {  // 지금까지 찾은 스타수열 최대값 보다 (출현 빈도 수 \* 2)가 작은 공통 수는 시도할 필요 없다.  if (entry.count \* 2 <= 최대길이) break;  스타수열찾기(A, 0, entry.number, 0);  }  return 최대길이;  }  }  public static void main(String[] args) {  Solution sol = new Solution();  System.out.println(sol.solution(new int[] {0})); // 0  System.out.println(sol.solution(new int[] {5,2,3,3,5,3})); // 4  System.out.println(sol.solution(new int[] {0,3,3,0,7,2,0,2,2,0})); // 8  System.out.println(sol.solution(new int[] {1, 1})); // 0  System.out.println(sol.solution(new int[] {1, 2, 1})); // 2  System.out.println(sol.solution(new int[] {1, 2, 1, 2})); // 4  System.out.println(sol.solution(new int[] {0, 1, 1, 2, 1, 2, 1})); // 6  }  } |

시간 초과 오류 발생한다.

성능을 개선하자.

그 등장 수 각각에 대해서

그 등장 수가 공통 수인, 가능한 모든 부분수열 각각에 대해서

그 부분수열이 스타수열인지 검사하고

스타수열이라면 그 길이의 최대값을 구한다.

시간 초과 오류를 피하려면

가능한 모든 부분수열을 다 시도하지 않아야 한다.

|  |
| --- |
| if (i > index && A[i - 1] != A[i]) // {i - 1, i} 한쌍을 수열에 추가하고  스타수열찾기(A, i + 1, 공통수, 길이 + 2); // 다음 쌍 찾기 재귀호출 (가)  if (i + 1 < A.length && A[i] != A[i + 1]) // {i, i + 1} 한쌍을 수열에 추가하고  스타수열찾기(A, i + 2, 공통수, 길이 + 2); // 다음 쌍 찾기 재귀호출 (나)  스타수열찾기(A, i + 1, 공통수, 길이); // 현재 위치 수를 무시하고 다음 쌍 찾기 재귀호출 (다) |

(가), (나), (다) 에서 구한 최대 수열 길이는

(가) >= (나) >= (다) 이다.

따라서 (가)를 호출했다면, (나) (다)를 생략하자

(가)를 호출하지 못했고, (나)를 호출했다면, (다)를 생략하자.

(가), (나)를 호춣하지 못한 경우에만, (다)를 호출하자.

|  |
| --- |
| if (i > index && A[i - 1] != A[i]) // {i - 1, i} 한쌍을 수열에 추가하고  스타수열찾기(A, i + 1, 공통수, 길이 + 2); // 다음 쌍 찾기 재귀호출 (가)  else if (i + 1 < A.length && A[i] != A[i + 1]) // {i, i + 1} 한쌍을 수열에 추가하고  스타수열찾기(A, i + 2, 공통수, 길이 + 2); // 다음 쌍 찾기 재귀호출 (나)  else 스타수열찾기(A, i + 1, 공통수, 길이); // 현재 위치 수를 무시하고 다음 쌍 찾기 재귀호출 (다) |

## 풀이 #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  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65 | package programmers.c30.e70130;  import java.util.ArrayList;  import java.util.Collections;  public class Main3 {  static class Solution {  static class Entry {  int number, count;  public Entry(int number, int count) {  this.number = number;  this.count = count;  }  }  static int 최대길이 = 0;  static void 스타수열찾기(int[] A, int index, int 공통수, int 길이) {  int i = index;  while (i < A.length && A[i] != 공통수)  ++i;  if (i >= A.length) {  if (길이 > 최대길이) 최대길이 = 길이;  return;  }  if (i > index && A[i - 1] != A[i])  스타수열찾기(A, i + 1, 공통수, 길이 + 2);  else if (i + 1 < A.length && A[i] != A[i + 1])  스타수열찾기(A, i + 2, 공통수, 길이 + 2);  else 스타수열찾기(A, i + 1, 공통수, 길이);  }  public int solution(int[] A) {  최대길이 = 0;  int[] count = new int[A.length + 1];  for (int i : A) count[i]++;  ArrayList<Entry> list = new ArrayList<>();  for (int i = 0; i < count.length; ++i)  if (count[i] > 0)  list.add(new Entry(i, count[i]));  Collections.sort(list, (a, b) -> b.count - a.count);  for (Entry entry : list) {  if (entry.count \* 2 <= 최대길이) break;  스타수열찾기(A, 0, entry.number, 0);  }  return 최대길이;  }  }  public static void main(String[] args) {  Solution sol = new Solution();  System.out.println(sol.solution(new int[] {0})); // 0  System.out.println(sol.solution(new int[] {5,2,3,3,5,3})); // 4  System.out.println(sol.solution(new int[] {0,3,3,0,7,2,0,2,2,0})); // 8  System.out.println(sol.solution(new int[] {1, 1})); // 0  System.out.println(sol.solution(new int[] {1, 2, 1})); // 2  System.out.println(sol.solution(new int[] {1, 2, 1, 2})); // 4  System.out.println(sol.solution(new int[] {0, 1, 1, 2, 1, 2, 1})); // 6  System.out.println(sol.solution(new int[100000])); // 0  }  } |

채점 도중 한 케이스에서 런타임 오류가 발생해서, (줄63)의 테스트에서

stack overflow error가 발생한다.

너무 많은 재귀호출 때문에 발생한 에러이다.

재귀호출을 반복문으로 수정하자.

|  |
| --- |
| static void 스타수열찾기(int[] A, int index, int 공통수, int 길이) {  int i = index;  while (i < A.length && A[i] != 공통수)  ++i;  if (i >= A.length) {  if (길이 > 최대길이) 최대길이 = 길이;  return;  }  if (i > index && A[i - 1] != A[i])  스타수열찾기(A, i + 1, 공통수, 길이 + 2);  else if (i + 1 < A.length && A[i] != A[i + 1])  스타수열찾기(A, i + 2, 공통수, 길이 + 2);  else 스타수열찾기(A, i + 1, 공통수, 길이);  } |

|  |
| --- |
| static void 스타수열찾기(int[] A, int index, int 공통수, int 길이) {  while (true) {  int i = index;  while (i < A.length && A[i] != 공통수)  ++i;  if (i >= A.length) {  if (길이 > 최대길이) 최대길이 = 길이;  return;  }  if (i > index && A[i - 1] != A[i]) {  index = i + 1; 길이 = 길이 + 2;  } else if (i + 1 < A.length && A[i] != A[i + 1]) {  index = i + 2; 길이 = 길이 + 2;  } else  index = i + 1;  }  } |

이렇게 재귀호출을 반복문으로 수정하는 기법이

꼬리 재귀 제거 최적화(tail recursion elimination optimization) 이다.

이 기법을 공부하자. (구글링)

## 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  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69 | package programmers.c30.e70130;  import java.util.ArrayList;  import java.util.Collections;  public class Main4 {  static class Solution {  static class Entry {  int number, count;  public Entry(int number, int count) {  this.number = number;  this.count = count;  }  }  static int 최대길이 = 0;  static void 스타수열찾기(int[] A, int index, int 공통수, int 길이) {  while (true) {  int i = index;  while (i < A.length && A[i] != 공통수)  ++i;  if (i >= A.length) {  if (길이 > 최대길이) 최대길이 = 길이;  return;  }  if (i > index && A[i - 1] != A[i]) {  index = i + 1; 길이 = 길이 + 2;  } else if (i + 1 < A.length && A[i] != A[i + 1]) {  index = i + 2; 길이 = 길이 + 2;  } else  index = i + 1;  }  }  public int solution(int[] A) {  최대길이 = 0;  int[] count = new int[A.length + 1];  for (int i : A) count[i]++;  ArrayList<Entry> list = new ArrayList<>();  for (int i = 0; i < count.length; ++i)  if (count[i] > 0)  list.add(new Entry(i, count[i]));  Collections.sort(list, (a, b) -> b.count - a.count);  for (Entry entry : list) {  if (entry.count \* 2 <= 최대길이) break;  스타수열찾기(A, 0, entry.number, 0);  }  return 최대길이;  }  }  public static void main(String[] args) {  Solution sol = new Solution();  System.out.println(sol.solution(new int[] {0})); // 0  System.out.println(sol.solution(new int[] {5,2,3,3,5,3})); // 4  System.out.println(sol.solution(new int[] {0,3,3,0,7,2,0,2,2,0})); // 8  System.out.println(sol.solution(new int[] {1, 1})); // 0  System.out.println(sol.solution(new int[] {1, 2, 1})); // 2  System.out.println(sol.solution(new int[] {1, 2, 1, 2})); // 4  System.out.println(sol.solution(new int[] {0, 1, 1, 2, 1, 2, 1})); // 6  System.out.println(sol.solution(new int[100000])); // 0  }  } |