LeetCode Training Day 23 State Machine

Finite state machine is used to process the token in the string, which allow you to transit from one state to another. The basic pattern is that any input may be corresponding to some state, and you need to transit a previous state to the next state.

## 1419. Minimum Number of Frogs Croaking

Medium

You are given the string croakOfFrogs, which represents a combination of the string "croak" from different frogs, that is, multiple frogs can croak at the same time, so multiple "croak" are mixed.

*Return the minimum number of*different*frogs to finish all the croaks in the given string.*

A valid "croak" means a frog is printing five letters 'c', 'r', 'o', 'a', and 'k' **sequentially**. The frogs have to print all five letters to finish a croak. If the given string is not a combination of a valid "croak" return -1.

**Example 1:**

**Input:** croakOfFrogs = "croakcroak"

**Output:** 1

**Explanation:** One frog yelling "croak**"** twice.

**Example 2:**

**Input:** croakOfFrogs = "crcoakroak"

**Output:** 2

**Explanation:** The minimum number of frogs is two.

The first frog could yell "**cr**c**oak**roak".

The second frog could yell later "cr**c**oak**roak**".

**Example 3:**

**Input:** croakOfFrogs = "croakcrook"

**Output:** -1

**Explanation:** The given string is an invalid combination of "croak**"** from different frogs.

**Constraints:**

* 1 <= croakOfFrogs.length <= 105
* croakOfFrogs is either 'c', 'r', 'o', 'a', or 'k'.

### Analysis:

The voice croak can be matched to state 0 to 4. The state 4 is a terminate state. When you see a voice, you can check what state it is, since the voice is sequential, it should reduce the count in lower state (if it is not zero) and increase in high state. At any time the sum of state from 0 to 4 is the total number of frogs. Remember when it reach the final state, after you count, reset it to zero for next repeat.

/// <summary>

/// Leet code #1419. Minimum Number of Frogs Croaking

///

/// Medium

///

/// Given the string croakOfFrogs, which represents a combination of

/// the string "croak" from different frogs, that is, multiple frogs

/// can croak at the same time, so multiple “croak” are mixed. Return

/// the minimum number of different frogs to finish all the croak in

/// the given string.

///

/// A valid "croak" means a frog is printing 5 letters 'c', 'r', 'o',

/// 'a', 'k' sequentially. The frogs have to print all five letters to

/// finish a croak. If the given string is not a combination of valid

/// "croak" return -1.

///

/// Example 1:

/// Input: croakOfFrogs = "croakcroak"

/// Output: 1

/// Explanation: One frog yelling "croak" twice.

///

/// Example 2:

/// Input: croakOfFrogs = "crcoakroak"

/// Output: 2

/// Explanation: The minimum number of frogs is two.

/// The first frog could yell "crcoakroak".

/// The second frog could yell later "crcoakroak".

///

/// Example 3:

/// Input: croakOfFrogs = "croakcrook"

/// Output: -1

/// Explanation: The given string is an invalid combination of

/// "croak" from different frogs.

///

/// Example 4:

/// Input: croakOfFrogs = "croakcroa"

/// Output: -1

///

/// Constraints:

/// 1. 1 <= croakOfFrogs.length <= 10^5

/// 2. All characters in the string are: 'c', 'r', 'o', 'a' or 'k'.

/// </summary>

int LeetCodeGreedy::minNumberOfFrogs(string croakOfFrogs)

{

vector<int> croak\_count(5);

unordered\_map<char, int> croak\_map;

string str = "croak";

for (size\_t i = 0; i < str.size(); i++)

{

croak\_map[str[i]] = i;

}

int frog\_count = 0;

int result = 0;

for (size\_t i = 0; i < croakOfFrogs.size(); i++)

{

int ord = croak\_map[croakOfFrogs[i]];

croak\_count[ord]++;

if (ord == 0)

{

frog\_count++;

}

else if (croak\_count[ord - 1] == 0)

{

return -1;

}

else

{

croak\_count[ord - 1]--;

}

result = max(result, frog\_count);

if (ord == 4)

{

croak\_count[ord]--;

frog\_count--;

}

}

for (size\_t i = 0; i < 4; i++)

{

if (croak\_count[i] != 0) return -1;

}

return result;

}