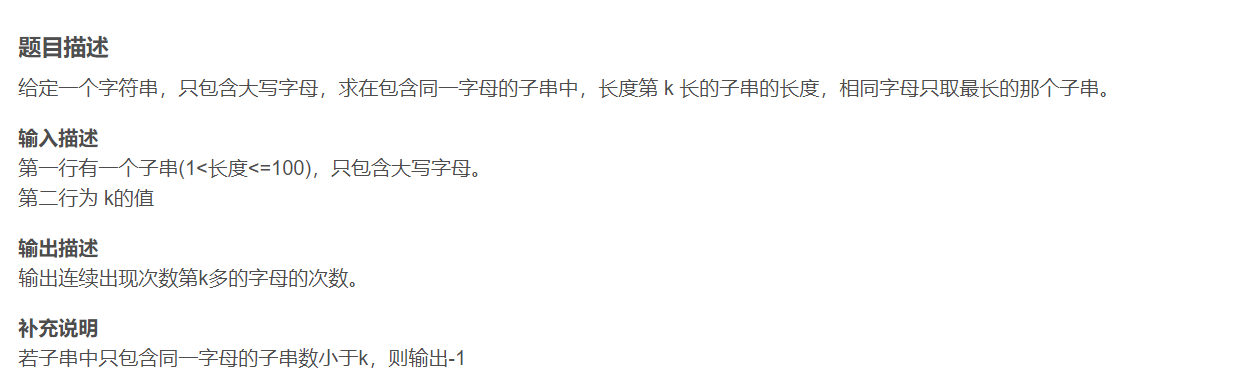
# **E卷-连续字母的长度[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go ）**





用例1

输入

AAAAHHHBBCDHHHH

3

输出

2

说明：同一字母连续出现的最多的是A和H，四次；第二多是H，三次，但H已经存在4个连续的，故不考虑；下个最长子串是BB，所以最终答案应该输出2。

用例2

输入

AABAAA

2

输出

1

说明：同一字母连续出现的最多的是A，三次；第二多还是A，两次，但A已经存在最大连续次数三次，故不考虑；下个最长子串是B，所以输出1

用例3

输入

ABC

4

输出

-1

说明：只含有3个包含同一字母的子串，小于k，输出-1

用例4

输入

ABC

2

输出

1

说明：三个子串长度均为1，所以此时k=1，k=2，k=3，这三种情况均输出1，特此说明，避免歧义。



import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s = sc.nextLine();

int n = sc.nextInt();

int []mp = new int[26];

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

int j = i;

int cnt = 1;

while (j + 1 < s.length() && s.charAt(j + 1) == s.charAt(j)) {

j++;

cnt++;

}

int id = s.charAt(j) - 'A';

mp[id] = Math.max(mp[id], cnt);

i = j;

}

List<Integer> res = new ArrayList<>();

for (int i = 0; i < 26; i++) {

if (mp[i] > 0) res.add(mp[i]);

}

res.sort((x, y)->y.compareTo(x));

if (n > res.size() || n <= 0)

System.out.println(-1);

else

System.out.println(res.get(n - 1));

}

}



def main():

import sys

input = sys.stdin.read

data = input().split("\n")

s = data[0]

n = int(data[1])

# 初始化一个大小为26的数组，表示26个英文字母

mp = [0] \* 26

# 遍历字符串s并记录每个字符的最长连续出现次数

i = 0

while i < len(s):

j = i

cnt = 1

while j + 1 < len(s) and s[j + 1] == s[j]:

j += 1

cnt += 1

id = ord(s[j]) - ord("A")

mp[id] = max(mp[id], cnt)

i = j + 1

# 收集所有大于0的连续次数

res = [x for x in mp if x > 0]

# 按降序排序这些次数

res.sort(reverse=True)

# 输出第n大的连续次数

if n > len(res) or n <= 0:

print(-1)

else:

print(res[n - 1])

if \_\_name\_\_ == "\_\_main\_\_":

main()



#include <iostream>

#include <vector>

#include <algorithm>

int main() {

std::string s;

int n;

// 读取输入

std::getline(std::cin, s); // 读取整个输入行作为字符串

std::cin >> n; // 读取整数

// 初始化一个长度为26的整数数组，用于记录每个字母的最长连续出现次数

std::vector<int> mp(26, 0);

// 遍历字符串s并记录每个字符的最长连续出现次数

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

int j = i;

int cnt = 1;

while (j + 1 < s.length() && s[j + 1] == s[j]) {

++j;

++cnt;

}

int id = s[j] - 'A';

mp[id] = std::max(mp[id], cnt);

i = j;

}

// 收集所有大于0的连续次数

std::vector<int> res;

for (int i = 0; i < 26; ++i) {

if (mp[i] > 0) res.push\_back(mp[i]);

}

// 按降序排序这些次数

std::sort(res.begin(), res.end(), std::greater<int>());

// 输出第n大的连续次数

if (n > res.size() || n <= 0)

std::cout << -1 << std::endl;

else

std::cout << res[n - 1] << std::endl;

return 0;

}



#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int compare(const void\* a, const void\* b) {

return (\*(int\*)b - \* (int\*)a);

}

int main() {

char s[1001];

int n;

// 读取输入

fgets(s, sizeof(s), stdin); // 读取整个输入行作为字符串

scanf("%d", &n); // 读取整数

// 去掉字符串末尾的换行符

s[strcspn(s, "\n")] = 0;

// 初始化一个大小为26的数组，记录每个字母的最长连续出现次数

int mp[26] = {0};

// 遍历字符串s并记录每个字符的最长连续出现次数

for (int i = 0; i < strlen(s); ++i) {

int j = i;

int cnt = 1;

while (j + 1 < strlen(s) && s[j + 1] == s[j]) {

++j;

++cnt;

}

int id = s[j] - 'A';

mp[id] = (cnt > mp[id]) ? cnt : mp[id];

i = j;

}

// 统计所有大于0的连续次数

int res[26];

int res\_size = 0;

for (int i = 0; i < 26; ++i) {

if (mp[i] > 0) {

res[res\_size++] = mp[i];

}

}

// 按降序排序这些次数

qsort(res, res\_size, sizeof(int), compare);

// 输出第n大的连续次数

if (n > res\_size || n <= 0) {

printf("-1\n");

} else {

printf("%d\n", res[n - 1]);

}

return 0;

}



const readline = require("readline");

// 创建接口来读取控制台输入

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

// 读取输入行

let input = [];

rl.on("line", (line) => {

input.push(line);

if (input.length === 2) {

rl.close();

}

}).on("close", () => {

const s = input[0];

const n = parseInt(input[1], 10);

// 初始化一个大小为26的数组，记录每个字母的最长连续出现次数

let mp = new Array(26).fill(0);

// 遍历字符串s并记录每个字符的最长连续出现次数

for (let i = 0; i < s.length; ++i) {

let j = i;

let cnt = 1;

while (j + 1 < s.length && s[j + 1] === s[j]) {

++j;

++cnt;

}

let id = s[j].charCodeAt(0) - "A".charCodeAt(0);

mp[id] = Math.max(mp[id], cnt);

i = j;

}

// 收集所有大于0的连续次数

let res = [];

for (let i = 0; i < 26; ++i) {

if (mp[i] > 0) res.push(mp[i]);

}

// 按降序排序这些次数

res.sort((a, b) => b - a);

// 输出第n大的连续次数

if (n > res.length || n <= 0) {

console.log(-1);

} else {

console.log(res[n - 1]);

}

});



package main

import (

"bufio"

"fmt"

"os"

"sort"

"strconv"

"strings"

)

func main() {

// 从标准输入中读取数据

reader := bufio.NewReader(os.Stdin)

s, \_ := reader.ReadString('\n')

nStr, \_ := reader.ReadString('\n')

// 去除新行字符

s = strings.TrimSpace(s)

nStr = strings.TrimSpace(nStr)

n, err := strconv.Atoi(nStr)

if err != nil {

fmt.Println("-1")

return

}

// 初始化一个大小为26的数组，记录每个字母的最长连续出现次数

mp := make([]int, 26)

// 遍历字符串s并记录每个字符的最长连续出现次数

for i := 0; i < len(s); i++ {

j := i

cnt := 1

for j+1 < len(s) && s[j+1] == s[j] {

j++

cnt++

}

id := s[j] - 'A'

if cnt > mp[id] {

mp[id] = cnt

}

i = j

}

// 收集所有大于0的连续次数

var res []int

for i := 0; i < 26; i++ {

if mp[i] > 0 {

res = append(res, mp[i])

}

}

// 按降序排序这些次数

sort.Sort(sort.Reverse(sort.IntSlice(res)))

// 输出第n大的连续次数

if n > len(res) || n <= 0 {

fmt.Println("-1")

} else {

fmt.Println(res[n-1])

}

}