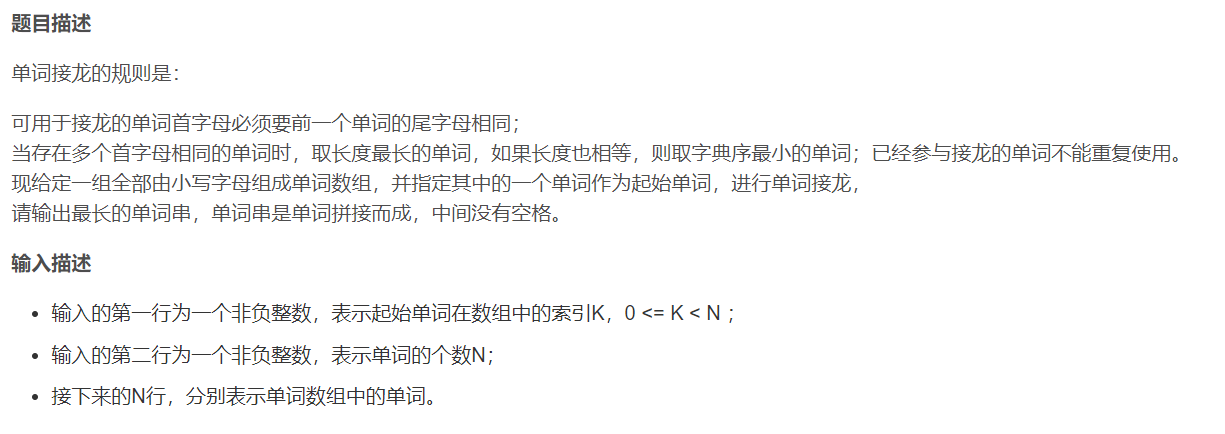
# **E卷-单词接龙[100分]（ Java | Python3 | C++ | C语言 | JsNode | Go）**











import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int k = sc.nextInt();

int n = sc.nextInt();

ArrayList<String> words = new ArrayList<>();

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

words.add(sc.next());

}

ArrayList<String> res = new ArrayList<>();

String st = words.remove(k);

res.add(st);

LinkedList<String>[] dq = new LinkedList[505];

for (int i = 0; i < 505; ++i) {

dq[i] = new LinkedList<String>();

}

for (String s : words) {

dq[s.charAt(0)].add(s);

}

for (int i = 0; i < 505; ++i) {

dq[i].sort((String a, String b) -> {

if (a.length() != b.length()) {

return b.length() - a.length();

}

return a.compareTo(b);

});

}

while (true) {

int sz = res.size();

String now = res.get(sz - 1);

int c = now.charAt(now.length() - 1);

if (dq[c].isEmpty()) {

break;

}

String s = dq[c].removeFirst();

res.add(s);

}

String ans = String.join("", res);

System.out.println(ans);

}

}



import sys

from collections import deque

def main():

input = sys.stdin.read

data = input().split()

k = int(data[0])

n = int(data[1])

words = data[2:(2+n)]

res = []

st = words.pop(k)

res.append(st)

dq = [[] for \_ in range(505)]

for s in words:

dq[ord(s[0])].append(s)

for strings in dq:

strings.sort(key=lambda a: (-len(a), a))

while True:

sz = len(res)

now = res[sz - 1]

c = ord(now[-1])

if not dq[c]:

break

s = dq[c].pop(0)

res.append(s)

ans = ''.join(res)

print(ans)

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

main()



#include <iostream>

#include <vector>

#include <string>

#include <deque>

#include <algorithm>

#include <sstream>

using namespace std;

int main() {

// 读取输入

int k, n;

cin >> k >> n;

cin.ignore();

vector<string> words(n);

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

cin >> words[i];

}

vector<string> res;

string st = words[k];

words.erase(words.begin() + k);

res.push\_back(st);

// 使用数组模拟多个双端队列

vector<deque<string>> dq(505);

for (const string& s : words) {

dq[s[0]].push\_back(s);

}

// 对双端队列进行排序

for (int i = 0; i < 505; ++i) {

sort(dq[i].begin(), dq[i].end(), [](const string & a, const string & b) {

if (a.length() != b.length()) {

return a.length() > b.length();

}

return a < b;

});

}

while (true) {

int sz = res.size();

string now = res.back();

char c = now.back();

if (dq[c].empty()) {

break;

}

string s = dq[c].front();

dq[c].pop\_front();

res.push\_back(s);

}

// 拼接最终结果

stringstream ans;

for (const string& s : res) {

ans << s;

}

cout << ans.str() << endl;

return 0;

}

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_WORDS 500

#define MAX\_WORD\_LENGTH 500

// 结构体定义来存储单词的长度及其内容

typedef struct {

char\*\* deque;

int front;

int back;

int capacity;

} Deque;

// 初始化双端队列

void initDeque(Deque\* dq, int capacity) {

dq->deque = (char\*\*)malloc(capacity \* sizeof(char\*));

dq->front = 0;

dq->back = 0;

dq->capacity = capacity;

}

// 向双端队列添加元素

void pushBack(Deque\* dq, char\* str) {

dq->deque[dq->back++] = str;

}

// 从双端队列中移除元素

char\* popFront(Deque\* dq) {

return dq->deque[dq->front++];

}

// 检查双端队列是否为空

int isEmpty(Deque\* dq) {

return dq->front == dq->back;

}

// 排序函数，用于根据长度和字典序排序

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

const char\* str1 = \*(const char\*\*)a;

const char\* str2 = \*(const char\*\*)b;

int len1 = strlen(str1);

int len2 = strlen(str2);

if (len1 != len2) {

return len2 - len1;

}

return strcmp(str1, str2);

}

int main() {

int k, n;

scanf("%d %d", &k, &n);

char\* words[MAX\_WORDS];

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

words[i] = (char\*)malloc(MAX\_WORD\_LENGTH \* sizeof(char));

scanf("%s", words[i]);

}

char\* res[MAX\_WORDS];

int resCount = 0;

char\* st = words[k];

// 删除第 k 个元素，并将其存入结果数组

for (int i = k; i < n - 1; i++) {

words[i] = words[i + 1];

}

n--; // 减少数组大小

res[resCount++] = st;

// 初始化双端队列数组

Deque deques[256];

for (int i = 0; i < 256; i++) {

initDeque(&deques[i], MAX\_WORDS);

}

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

char c = words[i][0];

pushBack(&deques[c], words[i]);

}

// 对每个双端队列进行排序

for (int i = 0; i < 256; i++) {

if (deques[i].back > 0) {

qsort(deques[i].deque + deques[i].front, deques[i].back - deques[i].front,

sizeof(char\*), compareStrings);

}

}

// 按规则拼接结果

while (1) {

char\* now = res[resCount - 1];

char c = now[strlen(now) - 1];

if (isEmpty(&deques[c])) {

break;

}

char\* s = popFront(&deques[c]);

res[resCount++] = s;

}

// 输出结果

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

printf("%s", res[i]);

}

printf("\n");

// 释放内存

for (int i = 0; i < 256; i++) {

free(deques[i].deque);

}

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

free(words[i]);

}

return 0;

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

const input = [];

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

input.push(line.trim());

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

// 读取输入

let [firstLine, ...words] = input;

let [k, n] = firstLine.split(" ").map(Number);

n = words[0];

words = words.slice(1);

const res = [];

const st = words[k];

words.splice(k, 1);

res.push(st);

// 使用对象模拟多个双端队列

const deques = {};

for (const word of words) {

const firstChar = word[0];

if (!deques[firstChar]) {

deques[firstChar] = [];

}

deques[firstChar].push(word);

}

Object.keys(deques).forEach((key) => {

deques[key].sort((a, b) => {

if (a.length !== b.length) {

return b.length - a.length;

}

return a < b ? -1 : a > b ? 1 : 0;

});

});

while (true) {

const now = res[res.length - 1];

const lastChar = now[now.length - 1];

if (!deques[lastChar] || deques[lastChar].length === 0) {

break;

}

const nextWord = deques[lastChar].shift();

res.push(nextWord);

}

// 拼接最终结果

const result = res.join("");

console.log(result);

});



package main

import (

"bufio"

"fmt"

"os"

"sort"

"strings"

)

func main() {

// 标准输入

reader := bufio.NewReader(os.Stdin)

var k, n int

fmt.Scanf("%d", &k)

fmt.Scanf("%d", &n)

words := make([]string, n)

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

words[i], \_ = reader.ReadString('\n')

words[i] = strings.TrimSpace(words[i])

}

st := words[k]

words = append(words[:k], words[k+1:]...) // 删除第k个元素

res := []string{st}

// 使用map模拟多个双端队列

deques := make(map[byte][]string)

for \_, word := range words {

firstChar := word[0]

deques[firstChar] = append(deques[firstChar], word)

}

// 对每个双端队列进行排序

for key := range deques {

sort.Slice(deques[key], func(i, j int) bool {

if len(deques[key][i]) != len(deques[key][j]) {

return len(deques[key][i]) > len(deques[key][j])

}

return deques[key][i] < deques[key][j]

})

}

// 按规则拼接结果

for {

now := res[len(res)-1]

lastChar := now[len(now)-1]

if len(deques[lastChar]) == 0 {

break

}

s := deques[lastChar][0]

deques[lastChar] = deques[lastChar][1:]

res = append(res, s)

}

// 输出结果

finalResult := strings.Join(res, "")

fmt.Println(finalResult)

}