ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 6

«Решение задач с использованием жадных алгоритмов

на сторонней платформе»

Выполнил работу

Воробьев Андрей

Академическая группа №J3110

Принято

Ментор, Вершинин Владислав

Санкт-Петербург

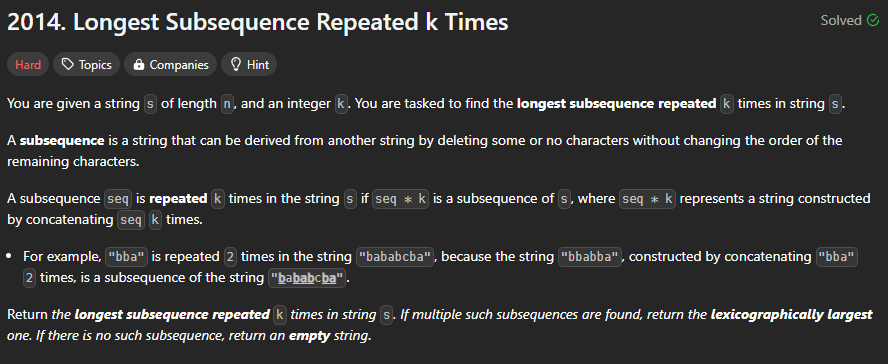
2024

1. Введение

Цель лабораторной работы – научиться уместно и осознанно применять на практике жадные алгоритмы.

Задачи:

* найти платформу для решения задач на жадные алгоритмы
* решить [задачу](https://leetcode.com/problems/find-original-array-from-doubled-array/description/?envType=problem-list-v2&envId=greedy), представленную на изображении 1:



Изображение 1 – Задача на платформе leetcode.com

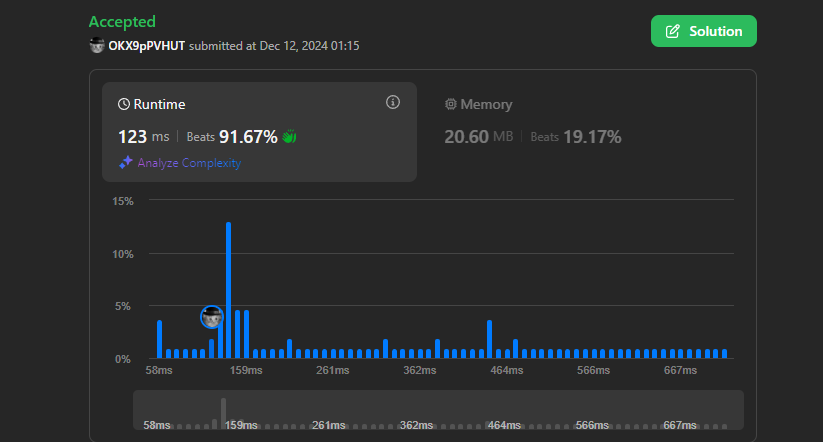
* пройти тесты на платформе, оценить корректность и производительность алгоритма
* осознать преимущества использования жадного алгоритма в решении задачи

1. Реализация

* реализация считывания строки и представления её как массива числа каждой из букв
* отбрасывание ненужных букв (которых будет недостаточно, чтобы повторить k раз в строке)
* динамическое увеличение размеров строки и сохранение в результат только наибольшего по лексикографии и по длине результата
* реализация проверки подстроки и отбрасывания не прошедших проверку подстрок из дальнейшего алгоритма

1. Экспериментальная часть.

Вся экспериментальная часть с тестами и проверками была реализована на платформе, результаты работы алгоритма представлены на изображении 2.



Изображение 2 – Результаты тестирования алгоритма

По результатам видно, что алгоритм полностью верен и лучше большинства других алгоритмов, написанных пользователями.

1. Заключение.

В ходе выполнения работы мною был реализован алгоритм, сочетающий в себе элементы динамического программирования и жадного алгоритма. Задача действительно, имеет единственное эффективное решение с использование жадного алгоритма, так как это позволяет убрать очень многие лишние вычисления. Это связано с тем, что задача сводится к последовательным проверкам возможных решение, и самое главное, это сократить этот перебор, что и позволяет сделать жадный алгоритм.

Алгоритм был реализован на языке C++.

1. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода алгоритма

class Solution {  
public:  
 bool check(const string &s, int k, const string word) {  
 *// проверка, что в строке s есть k подпоследовательностей word* int word\_index = 0;  
 for (char c: s) {  
 if (c == word[word\_index % word.size()])  
 word\_index++;  
 if (word\_index == k \* word.size())  
 return true;  
 }  
 return false;  
   
 *// O(n) - время* }  
  
 string longestSubsequenceRepeatedK(string s, int k) {  
 *// n = s.size()* vector<int> letter(26);  
 *// массив количеств букв в строке s* for (char c: s)  
 letter[c - 'a']++;  
 *// O(n)* for (int &amount: letter)  
 amount /= k;  
 *// считаем сколько букв можно использовать, чтобы они вместе могли встретиться k раз* set<pair<string, vector<int> > > words;  
 *// множество пар: {слово, массив неиспользованных букв}* words.insert({"", letter});  
  
 string result = "";  
  
 while (!words.empty()) {  
 set<pair<string, vector<int> > > new\_words;  
 *// новые более длинные слова* for (auto word = words.rbegin(); word != words.rend(); ++word) {  
 *// в порядке лексикографического убывания* for (int i = 25; i > -1; --i) {  
 if (word->second[i] > 0) {  
 if (check(s, k, word->first + (char) ('a' + i))) {  
 *// если новое слово встретилось k раз в строке s* vector<int> new\_letter = word->second;  
 --new\_letter[i];  
 new\_words.insert({word->first + (char) ('a' + i), new\_letter});  
   
 if (word->first.size() + 1 > result.size())  
 result = word->first + (char) ('a' + i);  
 }  
 }  
 }  
 }  
 *// O(n) - время* words = new\_words;  
 }  
 *// перебор длин может дойти до n/k  
 // O(n \* n/k) - время* return result;  
 }  
   
 *// O(n \* n/k) - время*};