

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

Отчёт по лабораторной работе № 7
«Задачи на жадные алгоритмы»

Выполнил работу
Муртазалиев Матвей
Академическая группа
J3110
Принято
Вершинин Владислав

Санкт-Петербург
2024

Структура отчёта:

1. Введение

Цель работы — научиться понимать и правильно применять жадные алгоритмы в работе. Задача — решить задачу на жадный алгоритм

2. Теоретическая подготовка

Необходимо знать концепцию работы жадных алгоритмов, полный перебор, brute force

3. Реализация

Идея решения заключается в переборе позиции на префиксе которой фиксированы цифры и их произведение для которого подбираются множители справа от этой позиции перебором. Для вычисления необходимых множителей используется НОД так как его можно разложить на общие множители

Сама позиция перебирается справа налево от конца (или первого нуля, так как его можно перебрать а нули справа от него изменить на подобранные множители)

Если после перебора не получилось найти решение единственный вариант это разложить t от большего к меньшему, справа налево. Также нужно учитывать что мы добавляем цифры в число и смотрим их произведение, а значит не можем использовать простые множители выше 1 разряда, то есть больше 9.

Если после разложения t мы не превзошли изначальное число, то так как ищем минимальное, будем добавлять единицы слева, пока не получим большее

4. Экспериментальная часть

Подсчёт по памяти (только для циклов и сложных структур) – $n \cdot 8$ байт для типа данных long long (хранение вектора need)

Подсчёт асимптотики (только для циклов и сложных структур) – $O(n^2 \cdot 10^2)$. Перебор возможных вариантов позиций и цифр

5. Заключение

В ходе выполнения работы я решил задачу на жадный алгоритм. Именно жадник здесь стоило использовать так как мы отталкивались от возможности перебрать позицию для фиксированного префикса и подобрать для него необходимые множители, чтобы при переумножении их всех получалось число кратное t

6. Приложения

ПРИЛОЖЕНИЕ А

Greedy < > ↺

Accepted 13 minutes ago C++ 96 ms 51.4 MB

Wrong Answer 15 minutes ago C++ N/A N/A

Wrong Answer 18 minutes ago C++ N/A N/A

Wrong Answer 20 minutes ago C++ N/A N/A

Wrong Answer 2 hours ago C++ N/A N/A

Wrong Answer 2 hours ago C++ N/A N/A

Wrong Answer 2 hours ago C++ N/A N/A

Time Limit Exceeded 2 hours ago C++ N/A N/A

Code

```
11 return false;
12 }
13 string smallestNumber(string num, long long t) {
14     int n = num.size(), stop = 0;
15     vector<long long> need(n + 1, t); // Сколько нужно добрать в суффиксе от i включительно
16     for (int i = 0; i < n; i++) {
17         if (num[i] == '0') { // Можно начать с первого нуля а остальные заменим подбором
18             stop = i + 1;
19             break;
20         }
21         need[i + 1] = need[i] / gcd(need[i], num[i] - '0');
22         stop++;
23     }
24     // cout << stop << "\n";
25     if (need[n] == 1) // Уже ответ
26         return num;
27
28     // Перебираем все позиции справа налево пытаемся найти минимальное
29     for (int i = stop - 1; i >= 0; i--) {
30         // cout << i << "\n";
31         // Увеличиваем число в текущей позиции до 9 включительно и подбираем часть справа
32         while (num[i] < '9') {
33             num[i]++;
34             long long cur = need[i];
35             cur /= gcd(cur, num[i] - '0'); // Сколько теперь добрать
36             //cout << cur << "\n";
37             for (int j = n - 1; j > i; j--) {
38                 for (int k = 9; k > 0; k--) {
39                     if (cur % k == 0) { // Добавляем если подходит
40                         num[j] = k + '0';
41                         cur /= k;
42                     }
43                 }
44             }
45             num[i]++;
46         }
47     }
48     return num;
49 }
```

Testcase > Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4 Case 5 Case 6

Input

num =

Greedy < > ↺

Description Editorial Solutions Submissions

3348. Smallest Divisible Digit Product II Solved

Hard Topics Companies Hint

You are given a string `num` which represents a **positive** integer, and an integer `t`.

A number is called **zero-free** if none of its digits are 0.

Return a string representing the **smallest zero-free** number greater than or equal to `num` such that the **product of its digits** is divisible by `t`. If no such number exists, return `"-1"`.

Example 1:

Input: `num = "1234", t = 256`

Output: `"1488"`

Explanation:

The smallest zero-free number that is greater than 1234 and has the product of its digits divisible by 256 is 1488, with the product of its digits equal to 256.

Example 2:

Input: `num = "12355", t = 50`

Output: `"12355"`

Explanation:

12355 is already zero-free and has the product of its digits divisible by 50, with the product of its digits equal to 150.

Example 3:

Input: `num = "11111", t = 26`

Output: `"-1"`

Explanation:

Code

```
11 return false;
12 }
13 string smallestNumber(string num, long long t) {
14     int n = num.size(), stop = 0;
15     vector<long long> need(n + 1, t); // Сколько нужно добрать в суффиксе от i включительно
16     for (int i = 0; i < n; i++) {
17         if (num[i] == '0') { // Можно начать с первого нуля а остальные заменим подбором
18             stop = i + 1;
19             break;
20         }
21         need[i + 1] = need[i] / gcd(need[i], num[i] - '0');
22         stop++;
23     }
24     // cout << stop << "\n";
25     if (need[n] == 1) // Уже ответ
26         return num;
27
28     // Перебираем все позиции справа налево пытаемся найти минимальное
29     for (int i = stop - 1; i >= 0; i--) {
30         // cout << i << "\n";
31         // Увеличиваем число в текущей позиции до 9 включительно и подбираем часть справа
32         while (num[i] < '9') {
33             num[i]++;
34             long long cur = need[i];
35             cur /= gcd(cur, num[i] - '0'); // Сколько теперь добрать
36             //cout << cur << "\n";
37             for (int j = n - 1; j > i; j--) {
38                 for (int k = 9; k > 0; k--) {
39                     if (cur % k == 0) { // Добавляем если подходит
40                         num[j] = k + '0';
41                         cur /= k;
42                     }
43                 }
44             }
45             num[i]++;
46         }
47     }
48     return num;
49 }
```

Testcase > Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4 Case 5 Case 6

Input