

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №2
по курсу «Алгоритмы и структуры данных»
Тема: Подстроки
Вариант 3

Выполнил:

Бунос М.В.

К3141

Проверила:

Артамонова В.Е.

Санкт-Петербург

2023 г.

Содержание отчета

Содержание отчета	2
Задачи	3
Задача №1. Наивный поиск подстроки в строке [2 s, 256 Mb, 1 балл]	3
Задача №2. Карта [2 s, 256 Mb, 1 балл]	6
Задача №3. Паттерн в тексте [2 s, 256 Mb, 1 балл]	9
Задача №4. Паттерн в тексте [2 s, 256 Mb, 1 балл]	12
Задача №5. Префикс-функция [2 s, 256 Mb, 1.5 балла]	15
Задача №6. Z-функция [2 s, 256 Mb, 1.5 балла]	18
Задача №7. Наибольшая общая подстрока [15 s, 512 Mb, 2 балла]	21
Задача №9. Декомпозиция строки [2 s, 256 Mb, 2 балла]	26
Задача №10-2. Басня о строке	31
Задача №10-3. Имена	34
Задача №10-4. Суффиксы	38
Задача №10-5. Поиск подстроки	42
Задача №10-6. Сдвиг текста	45
Задача №10-7. Сдвиг текста	49
Задача №10-8. Подстроки из одинаковых букв	52
Задача №10-9. Преобразование ДНК	56
Задача №10-10. Abracadabra	60
Вывод	64

Задачи

Задача №1. Наивный поиск подстроки в строке [2 s, 256 Mb, 1 балл]

Даны строки *p* и *t*. Требуется найти все вхождения строки *p* в строку *t* в качестве подстроки.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
    string p, t;
    fin >> p >> t;
```

```

vector <int> ans;

for (int i = 0; i <= t.size() - p.size(); i++) {
    bool found = true;
    for (int j = 0; j < p.size(); j++) {
        if (t[i+j] != p[j]) {
            found = false;
            break;
        }
    }
    if (found) {
        ans.push_back(i + 1);
    }
}

fout << ans.size() << '\n';

for(int i : ans)
    fout << i << ' ';
// ---- code ends here -----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Перебираем все возможные подстроки длины $|p|$ в t , если находим — добавляем в массив ответа

Результат работы кода на примерах из текста задачи:

input.txt	
1	aba
2	abaCaba
output.txt	
1	2
2	1 5

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.792969 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.031000 sec	0.878906 MB

Вывод по задаче:

Простая задача на поиск подстроки в строке.

Задача №2. Карта [2 s, 256 Mb, 1 балл]

В далеком 1744 году во время долгого плавания в руки капитана Александра Смоллетта попала древняя карта с указанием местонахождения сокровищ. Однако расшифровать ее содержание было не так уж и просто.

Команда Александра Смоллетта догадалась, что сокровища находятся на x шагов восточнее красного креста,

однако определить значение числа она не смогла. По возвращению на материк Александр Смоллетт решил обратиться

за помощью в расшифровке послания к знакомому мудрецу. Мудрец поведал, что данное послание таит за собой

некоторое число. Для вычисления этого числа необходимо было удалить все пробелы между словами, а потом посчитать

количество способов вычеркнуть все буквы кроме трех так, чтобы полученное слово из трех букв одинаково читалось

слева направо и справа налево.

Александр Смоллетт догадывался, что число, зашифрованное в послании, и есть число x . Однако, вычислить это число у него не получилось.

После смерти капитана карта была безнадежно утеряна до тех пор, пока не оказалась в ваших руках. Вы уже знаете

все секреты, осталось только вычислить число x .

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
```

```

    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
&pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
<< " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
" sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s;
    getline(fin, s);

    vector<int> cnt (26), curcnt(26);
    for(char c : s)
        if(c >= 'a' && c <= 'z')
            cnt[c - 'a'] ++;

    int64_t ans = 0;

    for(int i = 0; i < s.size(); ++ i) {
        if(s[i] >= 'a' && s[i] <= 'z') {
            cnt[s[i] - 'a']--;
            for (int j = 0; j < 26; ++j)
                ans += curcnt[j] * cnt[j];
            curcnt[s[i] - 'a']++;
        }
    }

    fout << ans;
    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Перебираем середину нашего палиндрома, а далее пробуем собрать из букв слева и справа. Количество – перемножение количества буквы слева на количество буквы справа.

Результат работы кода на примерах из текста задачи:

input.txt ×

1 you will never find the treasure

output.txt ×

1 146

input.txt ×

1 treasure

output.txt ×

1 8

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.800781 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB

Вывод по задаче:

Тоже довольно простая задачка.

Задача №3. Паттерн в тексте [2 s, 256 Mb, 1 балл]

В этой задаче ваша цель – реализовать алгоритм Рабина-Карпа для поиска заданного шаблона (паттерна) в заданном тексте.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

const int p = 61;
const int m = 1e9 + 9;

int getCode(char c) {
    if(c >= 'a' && c <= 'z')
        return c - 'a';
    return c - 'A' + 26;
}

vector<int> rabin_karp(const string& pattern, const string& text) {
```

```

int n = text.size(), cntM = pattern.size();
vector<long long> p_pow(n);
p_pow[0] = 1;
for (int i = 1; i < n; i++) {
    p_pow[i] = (p_pow[i-1] * p) % m;
}

vector<long long> h(n+1, 0);
for (int i = 0; i < n; i++) {
    h[i+1] = (h[i] + (getCode(text[i]) + 1) * p_pow[i]) % m;
}

long long h_pattern = 0;
for (int i = 0; i < cntM; i++) {
    h_pattern = (h_pattern + (getCode(pattern[i]) + 1) * p_pow[i]) % m;
}

vector<int> occurrences;
for (int i = 0; i <= n - cntM; i++) {
    long long cur_h = (h[i+cntM] + m - h[i]) % m;
    if (cur_h == h_pattern * p_pow[i] % m) {
        occurrences.push_back(i+1);
    }
}
return occurrences;
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here -----
    string pattern, text;
    fin >> pattern >> text;

    vector<int> occurrences = rabin_karp(pattern, text);

    int count = occurrences.size();
    fout << count << endl;
    for (int i = 0; i < count; i++) {
        fout << occurrences[i] << " ";
    }
    // ---- code ends here -----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

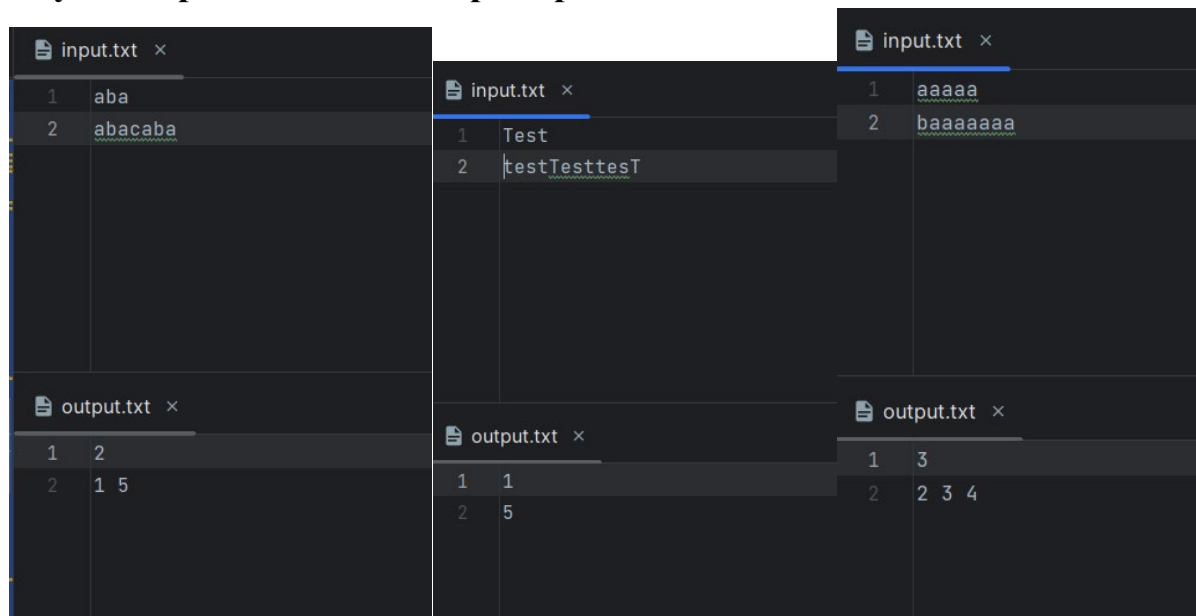
```

Текстовое объяснение решения:

Функция `rabin_karp` сначала предварительно вычисляет степени простого числа p по модулю большого простого числа m , а затем вычисляет хэш-значения всех подстрок текста, используя эти предварительно вычисленные

степени. Он также вычисляет хэш-значение шаблона. Наконец, он проверяет все подстроки текста длины m и для каждой подстроки вычисляет ее хэш-значение и проверяет, совпадает ли оно с хэш-значением шаблона. Если есть совпадение, он добавляет начальную позицию подстроки в список вхождений. Функция возвращает этот список вхождений.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.789062 MB
Пример из задачи	0.001000 sec	0.800781 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.789062 MB

Вывод по задаче:

Хорошая задача, много где используется в жизни

Задача №4. Паттерн в тексте [2 s, 256 Mb, 1 балл]

В этой задаче вы будете использовать хеширование для разработки алгоритма, способного предварительно обработать заданную строку s , чтобы ответить эффективно на любой запрос типа «равны ли эти две подстроки s ?» Это, в свою очередь, является основной частью во многих алгоритмах обработки строк.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

const int p = 31;
const int m = 1e9 + 9;

//-----

int main() {
    getFirstTime();
```

```

// ---- code starts here -----
string s;
fin >> s;

vector <int> pows (s.size() + 1);
pows[0] = 1;
for(int i = 1; i <= s.size(); ++ i) {
    pows[i] = (1ll * pows[i - 1] * p) % m;
}
vector <int> hashes (s.size() + 1, 0);
hashes[s.size() - 1] = (s[s.size() - 1] - 'a' + 1);

for(int i = s.size() - 2; i >= 0; -- i) {
    hashes[i] = (hashes[i + 1] + 1ll * (s[i] - 'a' + 1) * pows[s.size()
- i - 1]) % m;
}

int q;
fin >> q;

while(q --) {
    int a, b, l;
    fin >> a >> b >> l;
    if(a > b)
        swap(a, b);

    int firsthash = (hashes[a] - hashes[a + l]),
        secondhash = (1ll * (hashes[b] - hashes[b + l]) * pows[b - a])
% m;

    if(firsthash == secondhash) {
        fout << "Yes\n";
    } else {
        fout << "No\n";
    }
}
// ---- code ends here -----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Посчитаем хэш для суффиксов строки, а дальше будем сравнивать его, домножая на простое число в степени разницы индексов подстрок.

Результат работы кода на примерах из текста задачи:

```
input.txt x
1 trololo
2 4
3 0 0 7
4 2 4 3
5 3 5 1
6 1 3 2

output.txt x
1 Yes
2 Yes
3 Yes
4 No
5
```

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.781250 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.001000 sec	0.792969 MB

Вывод по задаче:

Классическая задача.

Задача №5. Префикс-функция [2 s, 256 Mb, 1.5 балла]

Постройте префикс-функцию для всех непустых префиксов заданной строки *s*.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

vector<int> prefix_function(const string &s) {
    int n = s.size();
    vector<int> pi(n);
    for (int i = 1; i < n; i++) {
        int j = pi[i - 1];
        while (j > 0 && s[i] != s[j]) {
            j = pi[j - 1];
        }
        if (s[i] == s[j]) {
            j++;
        }
    }
}
```

```

        pi[i] = j;
    }

    return pi;
}

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s;
    fin >> s;

    vector<int> pi = prefix_function(s);
    for (int i = 0; i < s.size(); i++) {
        fout << pi[i] << " ";
    }
    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Посчитаем префикс функцию, которая значит сколько символов суффикса совпадают с префиксом подстроки

Результат работы кода на примерах из текста задачи:

The image shows two side-by-side screenshots of a code editor window. Each window has two tabs: 'input.txt' and 'output.txt'.

Left Screenshot:

- input.txt:** Line 1 contains the string 'abacaba'.
- output.txt:** Line 1 contains the sequence of integers '0 0 1 0 1 2 3'.

Right Screenshot:

- input.txt:** Line 1 contains the string 'aaaAAA'.
- output.txt:** Line 1 contains the sequence of integers '0 1 2 0 0 0'.

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.001000 sec	0.781250 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB

Вывод по задаче:

Самый простой алгоритм с z-функцией на строки.

Задача №6. Z-функция [2 s, 256 Mb, 1.5 балла]

Постройте Z-функцию для заданной строки s. **Листинг кода:**

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s;
    fin >> s;

    int n = (int) s.length();
    vector<int> z(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i) {
        if (i <= r)
            z[i] = min(r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            z[i]++;
    }
}
```

```

        ++z[i];
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }

    for(int i = 1; i < z.size(); ++ i)
        fout << z[i] << ' ';
    // ---- code ends here -----

    printTimeUse();
    printMemoryUse();

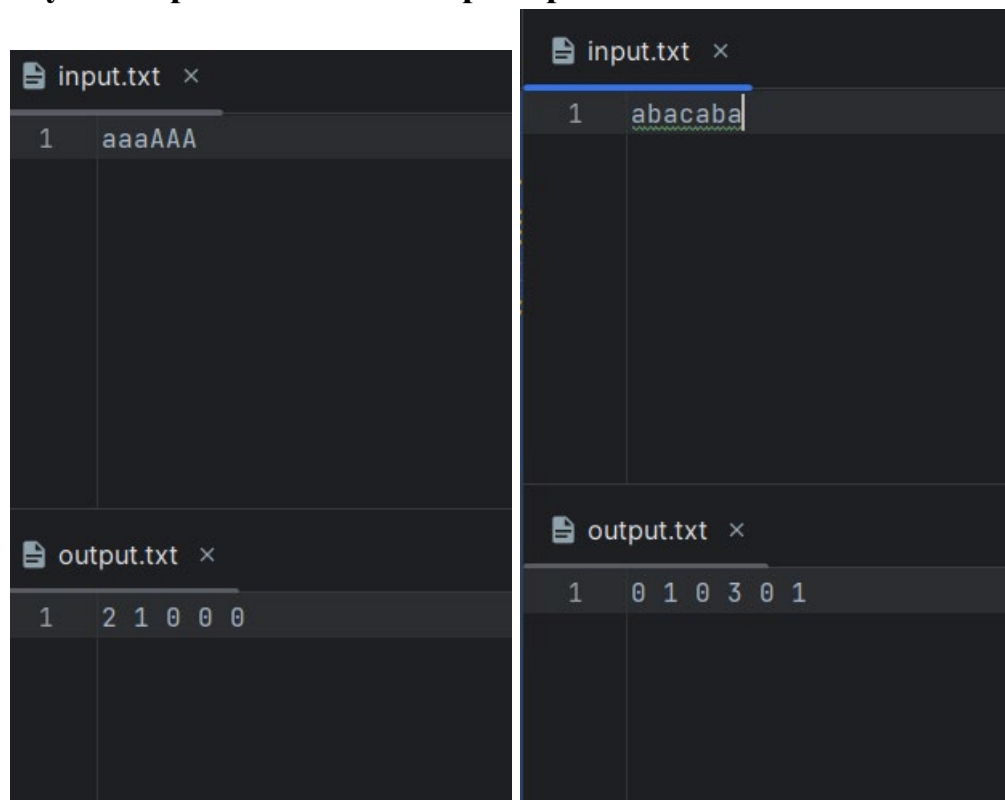
    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Посчитаем Z-функцию, которая значит сколько символов начинающих с I позиции совпадают с исходной строкой.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений	0.000000 sec	0.781250 MB

ВХОДНЫХ ДАННЫХ ИЗ ТЕКСТА ЗАДАЧИ		
Пример из задачи	0.000000 sec	0.777344 MB
Пример из задачи	0.001000 sec	0.785156 MB
Верхняя граница диапазона значений ВХОДНЫХ ДАННЫХ ИЗ ТЕКСТА ЗАДАЧИ	0.000000 sec	0.781250 MB

Вывод по задаче:

Самый простой алгоритм с префикс-функцией на строки.

Задача №7. Наибольшая общая подстрока [15 s, 512 Mb, 2 балла]

В задаче на наибольшую общую подстроку даются две строки s и t , и цель состоит в том, чтобы найти строку w максимальной длины, которая является подстрокой как s , так и t . Это естественная мера сходства между двумя строками. Задача имеет применения для сравнения и сжатия текстов, а также в биоинформатике. Эту проблему можно рассматривать как частный случай проблемы расстояния редактирования (Левенштейна), где разрешены только вставки и удаления. Следовательно, ее можно решить за время $O(|s||t|)$ с помощью динамического программирования. Есть также весьма нетривиальные структуры данных для решения этой задачи за линейное время $O(|s| + |t|)$. В этой задаче ваша цель — использовать хеширование для решения почти за линейное время.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <chrono>
#include <algorithm>
#include <type_traits>
#include <ext/pb_ds/assoc_container.hpp>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}
```

```

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----
#define int int64_t
int64_t binpow(int64_t a, int64_t b, int64_t mod) {
    int64_t res = 1;

    while(b) {
        if(b & 1)
            res = res * a % mod;
        a = a * a % mod;
        b /= 2;
    }

    return res;
}

vector<int> compute_hash(string s, int p, int mod, vector<int> &hash,
vector<int> &inv_mod) {
    int n = s.size();
    int p_pow = 1, hash_value = 0;

    for (int i = 0; i < n; i++) {
        int c = s[i] - 'a' + 1;
        hash_value = (hash_value + c * p_pow) % mod;
        hash[i] = hash_value;
        inv_mod[i] = binpow(p_pow, mod - 2, mod);
        p_pow = (p_pow * p) % mod;
    }

    return hash;
}

int get_hash(vector<int>& hash, vector<int>& inv_mod, int l, int r, int
mod) {
    if (l == 0) {
        return hash[r];
    }

    int window = (hash[r] - hash[l - 1] + mod) % mod;
    return (window * inv_mod[l]) % mod;
}

const int RANDOM =
chrono::high_resolution_clock::now().time_since_epoch().count();
struct chash {
    int operator()(pair<int, int> x) const { return x.first * 31 +
x.second; }
};

pair<int, pair<int, int>> longest_common_substr(string X, string Y, int
n, int m) {
    int p1 = 31, p2 = 37;

```

```

int m1 = 1e9 + 9, m2 = 1e9 + 7;
vector<int> inv_for_p1, inv_for_p2, hash1, hash2, hash3, hash4;
inv_for_p1.assign(max(n, m), 0);
inv_for_p2.assign(max(n, m), 0);
hash1.assign(n, 0);
hash2.assign(n, 0);
hash3.assign(m, 0);
hash4.assign(m, 0);
compute_hash(X, p1, m1, hash1, inv_for_p1);
compute_hash(X, p2, m2, hash2, inv_for_p2);
compute_hash(Y, p1, m1, hash3, inv_for_p1);
compute_hash(Y, p2, m2, hash4, inv_for_p2);

auto exists = [&](int k) -> pair<int, int> {
    if (k == 0) {
        return {0, 0};
    }
    __gnu_pbds::gp_hash_table<pair<int, int>, int, chash> st;
    for (int i = 0; i < n - k + 1; i++) {
        int h1 = get_hash(hash1, inv_for_p1, i, i + k - 1, m1);
        int h2 = get_hash(hash2, inv_for_p2, i, i + k - 1, m2);
        st[{h1, h2}] = i;
    }
    for (int i = 0; i < m - k + 1; i++) {
        int h1 = get_hash(hash3, inv_for_p1, i, i + k - 1, m1);
        int h2 = get_hash(hash4, inv_for_p2, i, i + k - 1, m2);
        if (st.find({h1, h2}) != st.end()) {
            return {st[{h1, h2}], i};
        }
    }
    return {-1, -1};
};

int answer = 0;
int low = 0, high = min(n, m);

pair<int, int> curAns;

while (low <= high) {
    int mid = (low + high) / 2;
    auto ans = exists(mid);
    if (ans.first != -1) {
        answer = mid;
        curAns = ans;
        low = mid + 1;
    } else {
        high = mid - 1;
    }
}

return {answer, curAns};
}

signed main() {
    getFirstTime();

    // ---- code starts here ----
    string f, s;
    while(fin >> f >> s) {
        auto ans = longest_common_substr(f, s, f.size(), s.size());
        fout << ans.second.first + 1 << ' ' << ans.second.second + 1 << ' '

```

```

<< ans.first << '\n';
    }
    // ---- code ends here -----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Дыхаем длину и проверяем по хэшам совпадают ли подстроки.

Результат работы кода на примерах из текста задачи:

```

input.txt x
1 cool toolbox
2 aaa bb
3 aabaa babbaab

output.txt x
1 2 2 3
2 1 1 0
3 3 4 3
4

```

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.796875 MB

Верхняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.800781 MB
---	--------------	-------------

Вывод по задаче:

Довольно сложная реализация, но я смог.

Задача №9. Декомпозиция строки [2 s, 256 Mb, 2 балла]

Строка ABCABCDEDEDEF содержит подстроку ABC , повторяющуюся два раза подряд, и подстроку DE , повторяющуюся три раза подряд. Таким образом, ее можно записать как $ABC*2+DE*3+F$, что занимает меньше места, чем исходная запись той же строки.

Ваша задача – построить наиболее экономное представление данной строки s в виде, продемонстрированном выше, а именно, подобрать такие $s_1, a_1, \dots, s_k, a_k$, где s_i - строки, а a_i - числа, чтобы $s = s_1 \cdot a_1 + \dots + s_k \cdot a_k$. Под операцией умножения строки на целое положительное число подразумевается конкатенация одной или нескольких копий строки, число которых равно числовому множителю, то есть, $ABC*2=ABCABC$. При этом требуется минимизировать общую длину итогового описания, в котором компоненты разделяются знаком $+$, а умножение строки на число записывается как умножаемая строка и множитель, разделенные знаком $*$. Если же множитель равен единице, его, вместе со знаком $*$, допускается не указывать.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;
```

```

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
<< " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
" sec\n";
}

//-----
vector<int> prefixFunction(string s, int offset) {
    int n = s.length();
    vector<int> p(n - offset, 0);

    for (int i = 1, j = 0; offset + i < n; ++i) {
        while (j > 0 && s[offset + i] != s[offset + j])
            j = p[j - 1];
        if (s[offset + i] == s[offset + j]) {
            ++j;
            p[i] = j;
        }
    }

    return p;
}

pair<int, int> findRepeatingPrefix(string s, int offset) {
    vector<int> p = prefixFunction(s, offset);
    int n = s.length() - offset;
    int max = p[0], maxIdx = 0;

    for (int i = 1; i < n; ++i) {
        if (p[i] > max && (i + 1) % (i + 1 - p[i]) == 0) {
            max = p[i];
            maxIdx = i;
        }
    }

    if (max > 0) {
        int l = maxIdx + 1 - max;
        return make_pair(l, (maxIdx + 1) / l);
    } else {
        return make_pair(1, 1);
    }
}

int cntDigitsInNumber(int num) {
    return (1 + log10(num));
}

string decomposeString(const string &s) {
    const int n = s.length();

    vector<int> minimum_join(n + 1), minimum_prefix(n + 1, 10000);
    vector<pair<int, int>> p(n);

```

```

    for (int i = n - 1; i >= 0; --i) {
        minimum_join[i] = 1 + min(minimum_join[i + 1], 1 + minimum_prefix[i
+ 1]);

        auto [l, k] = findRepeatingPrefix(s, i);
        p[i] = {l, k};

        const int minSuffixLen = min(minimum_join[i + l * k],
minimum_prefix[i + l * k]);
        minimum_prefix[i] = 1 + 1 + cntDigitsInNumber(k) + (minSuffixLen >
0 ? 1 : 0) + minSuffixLen;
    }

    vector<string> answer;

    int i = 0;
    bool needPlus = false;
    while (i < n) {
        if (minimum_join[i] <= minimum_prefix[i]) {
            if (needPlus) {
                answer.push_back("+");
            }
            answer.push_back(s.substr(i, 1));
            i += 1;
            needPlus = false;
        } else {
            const auto [l, k] = p[i];
            if (i > 0) {
                answer.push_back("+");
            }
            answer.push_back(s.substr(i, l) + "*" + to_string(k));
            i += l * k;
            needPlus = true;
        }
    }

    return accumulate(answer.begin(), answer.end(), string(""));
}

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s;
    fin >> s;

    fout << decomposeString(s);

    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Мы используем функции префикса для поиска самого длинного повторяющегося префикса s , начинающегося с каждой позиции i , а затем с помощью динамического программирования для вычисления минимальной длины представления суффикса s , начинающегося с каждой позиции i . Это делается путем рассмотрения всех возможных способов представления суффикса в виде конкатенации одной или нескольких копий повторяющихся префиксов и выбора представления, минимизирующего общую длину.

Результат работы кода на примерах из текста задачи:

```

input.txt x
1 ABCABCDDEDEF

output.txt x
1 ABC*2+DE*3+F
  
```

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.796875 MB

Верхняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
---	--------------	-------------

Вывод по задаче:

Хорошая задача на префикс-функцию, но можно решить с помощью z-функций.

Задача №10-2. Басня о строке

Однажды Три Программиста придумали занятную игру для тренировки памяти и умственных способностей. Первый Программист сочинил строку S из N символов и сообщил её Второму и Третьему Программистам. Второй Программист произвёл над этой строкой X ($0 \leq X < N$) последовательных циклических сдвигов (под циклическим сдвигом строки понимается перенос её последнего символа в начало). В результате этих манипуляций получилась строка T , которую он сообщил Третьему Программисту. Задачей Третьего Программиста было определить число X , либо сообщить Второму Программисту, что он ошибся, поскольку строка T не могла быть получена из строки S с помощью циклических сдвигов.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}
```

```

}

//-----

int main() {
    getFirstTime();

    // ---- code starts here -----
    int cnt;
    fin >> cnt;

    string s, s1, t;
    fin >> t >> s1;

    s = t + ' ' + s1 + s1;
    int l = 0, r = 0;
    int n = s.size();

    vector<int> z (n);
    for(int i = 0; i < n; i++)
        z[i] = 0;

    int id = 0;

    for(int i = 1; i < n; i++) {
        if(i <= r)
            z[i] = min(z[i - 1], r - i + 1);
        while(i + z[i] < n && s[z[i]] == s[i + z[i]])
            z[i]++;
        if(i + z[i] - 1 > r) {
            l = i;
            r = i + z[i] - 1;
        }
        if(z[i] > z[id])
            id = i;
    }

    if(z[id] != t.length())
        fout << -1;
    else fout << (id - cnt - 1);

    // ---- code ends here -----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Используем Z-функцию для решения данной задачи, с помощью нее найдем какой у нас циклический сдвиг (сформируем новую строку, состоящую из первой и двух вторых строк)

Результат работы кода на примерах из текста задачи:

input.txt ×

1

11

2

abracadabra

3

racadabraab

output.txt ×

1

9

Проверка задачи на (openedu, астр и тд при наличии в задаче):

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10176606	05:32:49 21 фев 2023	g1rby	1423_Басня о строке	G++ 9.2 x64	Accepted		0.078	5 076 КБ

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.808594 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.078000 sec	3.781250 MB

Вывод по задаче:

Неплохая задача.

Задача №10-3. Имена

На далекой планете Тау Кита есть непонятные нам обычаи. Например, таукитяне очень необычно для землян выбирают имена своим детям. Родители так выбирают имя ребенку, чтобы оно могло быть получено как удалением некоторого набора букв из имени отца, так и удалением некоторого набора букв из имени матери. Например, если отца зовут «abacaba», а мать — «bbssaa», то их ребенок может носить имена «a», «bba», «bcaa», но не может носить имена «aaa», «ab» или «bbc». Возможно, что имя ребенка совпадает с именем отца и/или матери, если оно может быть получено из имени другого родителя удалением нескольких (возможно, ни одной) букв.

Пусть отец по имени X и мать по имени Y выбирают имя своему новорожденному ребенку. Так как в таукитянских школах учеников часто вызывают к доске в лексикографическом порядке имен учеников, то есть в порядке следования имен в словаре, то они хотят выбрать своему ребенку такое имя, чтобы оно лексикографически следовало как можно позже.

Формально, строка S лексикографически больше строки T , если выполняется одно из двух условий:

- строка T получается из S удалением одной или более букв с конца строки S ;
- первые $(i - 1)$ символов строк T и S не различаются, а буква в i -й позиции строки T следует в алфавите раньше буквы в i -й позиции строки S .

Требуется написать программу, которая по именам отца и матери находит лексикографически наибольшее имя для их ребенка.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>
```

```

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
&pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
<< " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
" sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s, f;
    fin >> s >> f;

    vector<int> cnt1(26), cnt2(26);
    for (char i: s) {
        cnt1[i - 'a']++;
    }

    for (char i: f) {
        cnt2[i - 'a']++;
    }

    string ans;
    pair<int, int> curPos = {0, 0};
    for (int i = 25; i >= 0; --i) {
        while (cnt1[i] && cnt2[i] && curPos.first < s.size() &&
curPos.second < f.size()) {
            while (s[curPos.first] != ('a' + i)) {
                cnt1[s[curPos.first] - 'a'] --;
                curPos.first++;
            }

            while (f[curPos.second] != ('a' + i)) {
                cnt2[f[curPos.second] - 'a'] --;

```

```

        curPos.second++;
    }

    ans += ('a' + i);
    cnt1[i]--;
    cnt2[i]--;
    curPos.first++;
    curPos.second++;
}

}

fout << ans;

// ---- code ends here -----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

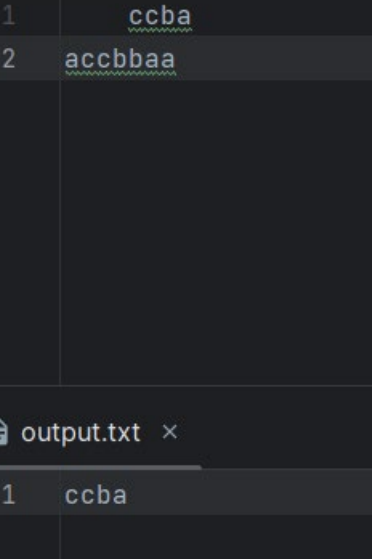
Текстовое объяснение решения:

Посчитаем количество каждой буквы в строках, дальше переберем буквы с больших к меньших, если они есть в наших строках, то будем идти до первой встречной, по пути убирая количества. Таким образом, сформируем максимально лексикографическую строку.

Результат работы кода на примерах из текста задачи:

```
input.txt  ×
1  abcabca
2  abcda

output.txt  ×
1  ca
```



The screenshot shows a text editor with two files open. The first file, 'input.txt', contains two lines of text: '1 ccba' and '2 accbbaa'. The second file, 'output.txt', contains one line of text: '1 ccba'.

File	Line	Content
input.txt	1	ccba
	2	accbbaa
output.txt	1	ccba

Проверка задачи на (openedu, астр и тд при наличии в задаче):

33	curPos.second++;	23	Accepted	0,015	404 Кб
34	}	24	Accepted	0,015	408 Кб
35		25	Accepted	0,015	412 Кб
36	ans += ('a' + i);	26	Accepted	0,015	412 Кб
37	cnt1[i]--;	27	Accepted	0,015	412 Кб
38	cnt2[i]--;	28	Accepted	0,015	408 Кб
39	curPos.first ++;	29	Accepted	0,015	412 Кб
40	curPos.second ++;	30	Accepted	0,015	408 Кб
41	}	31	Accepted	0,015	912 Кб
42	}	32	Accepted	0,015	528 Кб
43		33	Accepted	0,015	672 Кб
44	cout << ans;	34	Accepted	0,015	1036 Кб
45		35	Accepted	0,015	660 Кб
46	return 0;	36	Accepted	0,015	676 Кб
47		37	Accepted	0,015	732 Кб
48	}	38	Accepted	0,015	908 Кб
Размер кода: 597		39	Accepted	0,03	756 Кб
Посылки решений:		40	Accepted	0,015	680 Кб
		41	Accepted	0,015	688 Кб
		42	Accepted	0,015	624 Кб
		43	Accepted	0,015	576 Кб
		44	Accepted	0,015	764 Кб
		45	Accepted	0,015	516 Кб
		46	Accepted	0,03	748 Кб
		47	Accepted	0,015	644 Кб
		48	Accepted	0,015	664 Кб
		49	Accepted	0,015	636 Кб
		50	Accepted	0,015	656 Кб

ID	Дата	Язык	Результат	Тест	Время	Память
18860482	21.02.2023 3:58:45	C++	Accepted		0,03	1044 Кб
18860479	21.02.2023 3:57:28	C++	Wrong answer	2	0,015	416 Кб
18860474	21.02.2023 3:51:54	C++	Time limit exceeded	12	1,468	912 Кб
18860473	21.02.2023 3:50:58	C++	Time limit exceeded	12	1,468	788 Кб

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.792969 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.015000 sec	0.792969 MB

Вывод по задаче:

Легкая задача на жадный алгоритм на строках.

Задача №10-4. Суффиксы

По данной строке S выясните, сколько ее суффиксов S_i имеют такое же циклическое расширение, как и сама строка S , то есть количество таких i , что $S^* = S_i^*$.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s;
    fin >> s;
    int n, l = 0, r = 0;
```

```

n = s.length();

vector<int> z (n);
for(int i = 0; i < n; i++)
    z[i] = 0;

int ans = 1;

for(int i = 1; i < n; i++) {
    if(i <= r)
        z[i] = min(z[i - 1], r - i + 1);
    while(i + z[i] < n && s[z[i]] == s[i + z[i]])
        z[i]++;
    if(i + z[i] - 1 > r) {
        l = i;
        r = i + z[i] - 1;
    }
    if(i + z[i] == n) {
        if(n % i == 0) {
            ans = n / i;
        }
        break;
    }
}

fout << ans;

// ---- code ends here -----

printTimeUse();
printMemoryUse();

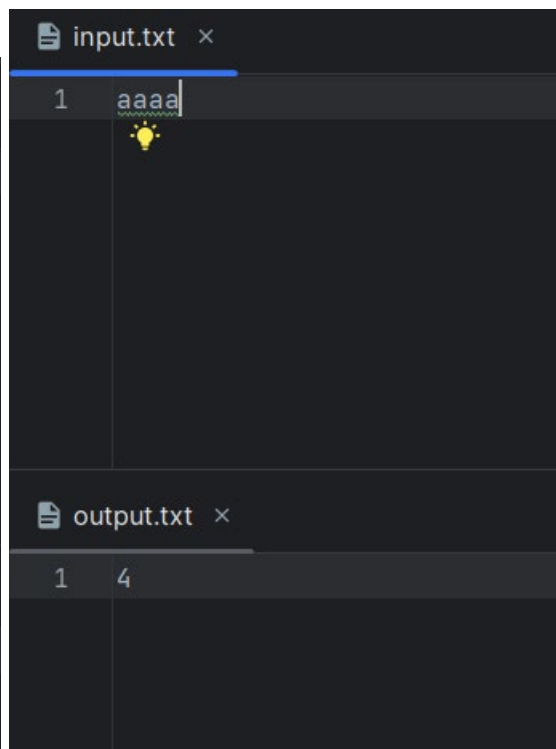
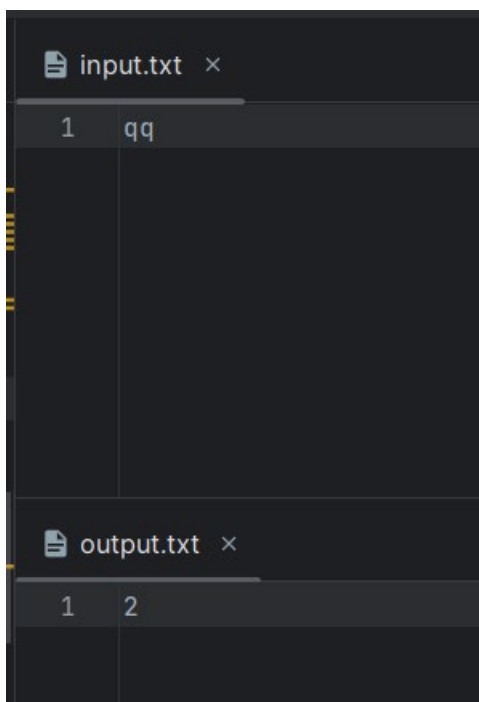
fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Посчитаем z-функцию, а потом найдем минимальный период строки. Ответом будет являться $n / \text{длину периода}$ или 1, если он отсутствует.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

20		15	Accepted	0,015	1036 Кб
21	for(int i = 1; i < n; i++) {	16	Accepted	0,015	1040 Кб
22	if(i <= r)	17	Accepted	0,015	492 Кб
23	z[i] = min(z[i - 1], r - i + 1);	18	Accepted	0,015	1016 Кб
24	while(i + z[i] < n && s[z[i]] == s[i + z[i]])	19	Accepted	0,015	1028 Кб
25	z[i] ++;	20	Accepted	0,015	1032 Кб
26	if(i + z[i] - 1 > r) {	21	Accepted	0,015	936 Кб
27	l = i;	22	Accepted	0,015	924 Кб
28	r = i + z[i] - 1;	23	Accepted	0,015	932 Кб
29	}	24	Accepted	0,03	900 Кб
30	if(i + z[i] == n) {	25	Accepted	0,015	1020 Кб
31	if(n % i == 0) {	26	Accepted	0,015	1000 Кб
32	ans = n / i;	27	Accepted	0,015	996 Кб
33	} break;	28	Accepted	0,015	992 Кб
34	}	29	Accepted	0,015	960 Кб
35	}	30	Accepted	0,015	1004 Кб
36	}	31	Accepted	0,015	920 Кб
37		32	Accepted	0,015	1028 Кб
38	cout << ans;	33	Accepted	0,015	1012 Кб
39	return 0;	34	Accepted	0,015	1020 Кб
40		35	Accepted	0,015	996 Кб
41	}	36	Accepted	0,015	1000 Кб
		37	Accepted	0,015	996 Кб
		38	Accepted	0,015	956 Кб
		39	Accepted	0,015	1004 Кб
		40	Accepted	0,015	924 Кб
		41	Accepted	0,015	1036 Кб
		42	Accepted	0,015	1016 Кб

Размер кода: 400

Посылки решений:

ID	Дата	Язык	Результат	Тест	Время	Память
18860497	21.02.2023 4:15:22	C++	Accepted		0,03	1044 Кб

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.808594 MB

Верхняя граница диапазона значений входных данных из текста задачи	0.015000 sec	0.892969 MB
---	--------------	-------------

Вывод по задаче:

Прикольная задача на z-функцию.

Задача №10-5. Поиск подстроки

Найти все вхождения строки T в строке S.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s, f;
    fin >> s >> f;

    string newstr = f + '#' + s;
    int n = newstr.size();
    vector<int> z (n);
    int l, r;
```

```

l = r = 0;
for(int i = 1; i < n; ++ i) {
    if(i <= r)
        z[i] = min(r - i + 1, z[i - 1]);
    while(i + z[i] < n && newstr[z[i]] == newstr[i + z[i]])
        z[i] ++;
    if(i + z[i] - 1 > r) {
        l = i;
        r = i + z[i] - 1;
    }
}

for(int i = 0; i < n; ++ i) {
    if(z[i] == f.size())
        fout << i - f.size() - 1 << ' ';
}

// ---- code ends here ----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Посчитаем z-функцию для совмещенной строки s и f, а затем если z-функция = f, значит она входит в s, выведем индекс.

Результат работы кода на примерах из текста задачи:

input.txt	
1	ababbababa
2	aba
output.txt	
1	0 5 7

Проверка задачи на (openedu, астр и тд при наличии в задаче):

[Вернуться к задаче] [Редактировать решение]

```

1 #include <iostream>
2 #include <fstream>
3 #include <vector>
4
5 using namespace std;
6
7 int main() {
8     ios_base::sync_with_stdio(false);
9     cin.tie(0);
10    string s, f;
11    cin >> s >> f;
12
13    string newstr = f + '#' + s;
14    int n = newstr.size();
15    vector<int> z (n);
16    int l, r;
17    l = r = 0;
18    for(int i = 1; i < n; ++ i) {
19        if(i <= r)
20            z[i] = min(r - i + 1, z[i - 1]);
21        while(i + z[i] < n && newstr[z[i]] == newstr[i + z[i]])
22            z[i]++;
23        if(i + z[i] - 1 > r) {
24            l = i;
25            r = i + z[i] - 1;
26        }
27    }
28
29    for(int i = 0; i < n; ++ i) {
30        if(z[i] == f.size())
31            cout << i - f.size() - 1 << ' ';
32    }
33
34    return 0;
35 }
```

Размер кода: 418

Тест	Результат	Время	Память
1	Accepted	0,03	404 Кб
2	Accepted	0,015	408 Кб
3	Accepted	0,015	412 Кб
4	Accepted	0,015	408 Кб
5	Accepted	0,015	404 Кб
6	Accepted	0,03	404 Кб
7	Accepted	0,015	588 Кб
8	Accepted	0,015	688 Кб
9	Accepted	0,015	588 Кб
10	Accepted	0,015	1320 Кб
11	Accepted	0,03	1000 Кб
12	Accepted	0,015	1000 Кб
13	Accepted	0,015	400 Кб

Посылки решений:

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.796875 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.015000 sec	0.796875 MB

Вывод по задаче:

Классическая задача на z-функцию.

Задача №10-6. Сдвиг текста

Мальчик Кирилл написал однажды на листе бумаги строчку, состоящую из больших и маленьких английских букв, а после этого ушел играть в футбол. Когда он вернулся, то обнаружил, что его друг Дима написал под его строкой еще одну строчку такой же длины. Дима утверждает, что свою строчку он получил циклическим сдвигом строки Кирилла направо на несколько шагов (циклический сдвиг строки abcde на 2 позиции направо даст строку deabc). Однако Дима известен тем, что может случайно ошибиться в большом количестве вычислений, поэтому Кирилл в растерянности - верить ли Диме? Помогите ему!

По данным строкам выведите минимально возможный размер сдвига вправо или -1, если Дима ошибся.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
```

```

    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
    int cnt;
    string s, s1, t;
    fin >> t >> s1;
    cnt = t.size();

    s = t + ' ' + s1 + s1;
    int l = 0, r = 0;
    int n = s.size();

    vector<int> z (n);
    for(int i = 0; i < n; i++)
        z[i] = 0;

    int id = 0;

    for(int i = 1; i < n; i++) {
        if(i <= r)
            z[i] = min(z[i - 1], r - i + 1);
        while(i + z[i] < n && s[z[i]] == s[i + z[i]])
            z[i]++;
        if(i + z[i] - 1 > r) {
            l = i;
            r = i + z[i] - 1;
        }
        if(z[i] > z[id])
            id = i;
    }

    if(z[id] != t.length())
        fout << -1;
    else fout << (id - cnt - 1);

    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Используем Z-функцию для решения данной задачи, с помощью нее найдем какой у нас циклический сдвиг (сформируем новую строку, состоящую из первой и двух вторых строк)

Результат работы кода на примерах из текста задачи:

input.txt ×

1

abcde

2

deabc

output.txt ×

1

2

Проверка задачи на (openedu, астр и тд при наличии в задаче):

Вернуться к задаче

Редактировать решение

```
1 #include <iostream>
2 #include <fstream>
3 #include <vector>
4
5 using namespace std;
6
7 int main() {
8     int cnt;
9     string s, s1, t;
10    cin >> t >> s1;
11    cnt = t.size();
12
13    s = t + ' ' + s1 + s1;
14    int l = 0, r = 0;
15    int n = s.size();
16
17    vector<int> z (n);
18    for(int i = 0; i < n; i++)
19        z[i] = 0;
20
21    int id = 0;
22
23    for(int i = 1; i < n; i++) {
24        if(i <= r)
25            z[i] = min(z[i - 1], r - i + 1);
26        while(i + z[i] < n && s[z[i]] == s[i + z[i]])
27            z[i]++;
28        if(i + z[i] - 1 > r) {
29            l = i;
30            r = i + z[i] - 1;
31        }
32        if(z[i] > z[id])
33            id = i;
34    }
35
36    if(z[id] != t.length())
37        cout << -1;
38    else cout << (id - cnt - 1);
39    return 0;
40 }
```

Тест	Результат	Время	Память
1	Accepted	0,015	404 Кб
2	Accepted	0,015	408 Кб
3	Accepted	0,015	412 Кб
4	Accepted	0,015	412 Кб
5	Accepted	0,03	416 Кб
6	Accepted	0,015	412 Кб
7	Accepted	0,015	412 Кб
8	Accepted	0,015	440 Кб
9	Accepted	0,015	520 Кб
10	Accepted	0,015	628 Кб
11	Accepted	0,015	640 Кб
12	Accepted	0,015	636 Кб
13	Accepted	0,015	640 Кб
14	Accepted	0,015	644 Кб
15	Accepted	0,015	640 Кб
16	Accepted	0,015	636 Кб
17	Accepted	0,015	640 Кб
18	Accepted	0,015	412 Кб

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 МВ
Пример из задачи	0.000000 sec	0.796875 МВ

Верхняя граница диапазона значений входных данных из текста задачи	0.015000 sec	0.415875 MB
---	--------------	-------------

Вывод по задаче:

Аналогично классическая задача на z-функцию.

Задача №10-7. Сдвиг текста

Строка S была записана много раз подряд, после чего из получившейся строки взяли подстроку и дали Вам. Ваша задача определить минимально возможную длину исходной строки S.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s;
    fin >> s;
    int n, l = 0, r = 0;
```

```

n = s.length();

vector<int> z (n);
for(int i = 0; i < n; i++)
    z[i] = 0;

int ans = 1;

for(int i = 1; i < n; i++) {
    if(i <= r)
        z[i] = min(z[i - 1], r - i + 1);
    while(i + z[i] < n && s[z[i]] == s[i + z[i]])
        z[i]++;
    if(i + z[i] - 1 > r) {
        l = i;
        r = i + z[i] - 1;
    }
    if(i + z[i] == n) {
        cout << i;
        return 0;
    }
}

fout << n;

// ---- code ends here ----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Найдем период с помощью z-функции, посчитав если $i+z[i] == n$, значит это период и выведем его.

Результат работы кода на примерах из текста задачи:

input.txt	
1	abababa
output.txt	
1	2

Проверка задачи на (openedu, астр и тд при наличии в задаче):

[курсы](#)
[\[олимпиады\]](#)

Учащийся: Матвей Бунос
[Выход](#)

ИСХОДНИК РЕШЕНИЯ №18860511 ЗАДАЧИ №204

[\[Вернуться к задаче\]](#)
[\[Редактировать решение\]](#)

```

1  #include <iostream>
2  #include <fstream>
3  #include <vector>
4
5  using namespace std;
6
7  int main() {
8      string s;
9      cin >> s;
10     int n, l = 0, r = 0;
11     n = s.length();
12
13     vector<int> z (n);
14     for(int i = 0; i < n; i++)
15         z[i] = 0;
16
17     int ans = 1;
18
19     for(int i = 1; i < n; i++) {
20         if(i <= r)
21             z[i] = min(z[i - 1], r - i + 1);
22         while(i + z[i] < n && s[z[i]] == s[i + z[i]])
23             z[i]++;
24         if(i + z[i] - 1 > r) {
25             l = i;
26             r = i + z[i] - 1;
27         }
28         if(i + z[i] == n) {
29             cout << i;
30             return 0;
31         }
32     }
33
34     cout << n;
35     return 0;
36 }

```

Размер кода: 344

Тест	Результат	Время	Память
1	Accepted	0,015	404 Кб
2	Accepted	0,015	408 Кб
3	Accepted	0,015	400 Кб
4	Accepted	0,015	408 Кб
5	Accepted	0,015	400 Кб
6	Accepted	0,015	404 Кб
7	Accepted	0,015	412 Кб
8	Accepted	0,015	556 Кб
9	Accepted	0,015	556 Кб
10	Accepted	0,015	668 Кб
11	Accepted	0,03	700 Кб
12	Accepted	0,015	700 Кб
13	Accepted	0,015	696 Кб

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.796875 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.015000 sec	0.796875 MB

Вывод по задаче:

Аналогично классическая задача на z-функцию.

Задача №10-8. Подстроки из одинаковых букв

В заданной строке, состоящей из малых английских букв, необходимо найти пару самых длинных подстрок, состоящих из одних и тех же букв (возможно, в разном порядке). Например, в строке twotwow это будут подстроки wotwo и otwow.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s;
    fin >> s;
```

```

int n = s.size();
vector <vector <int> > cnt (n + 1, vector <int> (26, 0));
for(int i = 1; i <= n; ++ i) {
    cnt[i] = cnt[i - 1];
    cnt[i][s[i - 1] - 'a']++;
}

for(int length = n - 1; length >= 1; -- length) {
    for(int i = 0; i < n - length + 1; ++ i) {
        for(int j = 0; j < n - length + 1; ++ j) {
            if(i == j)
                continue;

            bool ok = true;
            for(int z = 0; z < 26; ++ z)
                if(cnt[j + length][z] - cnt[j][z] != cnt[i + length][z]
- cnt[i][z]) {
                    ok = false;
                    break;
                }

            if(ok) {
                cout << length;
                return 0;
            }
        }
    }

    fout << 0;

    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

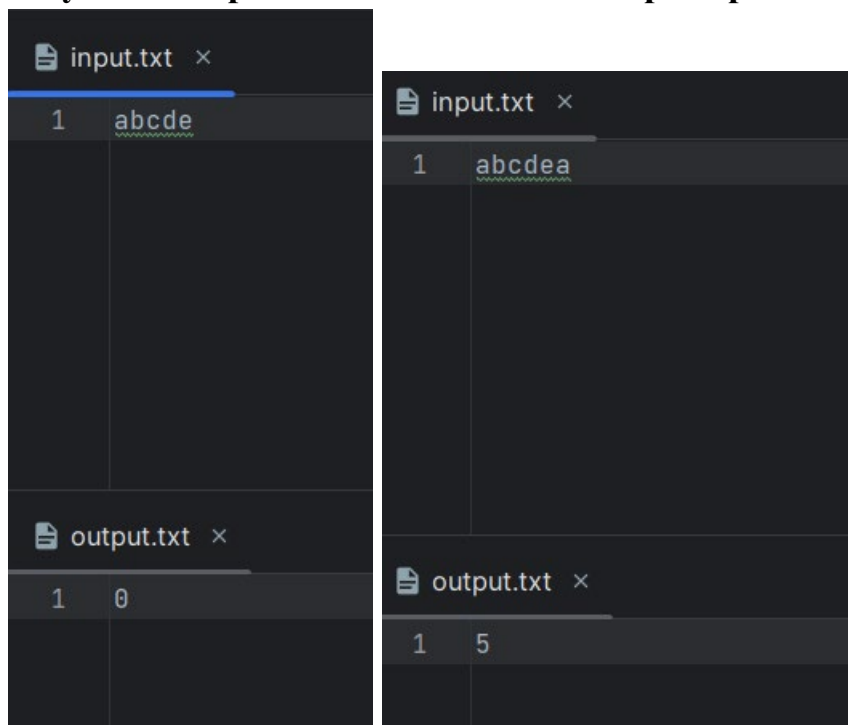
    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Предпочитаем количество символов на отрезках $[0..i]$, далее переберем длину, переберем начало первой подстроки и начало второй подстроки, сравним количество символов на них и, если количество символов совпадает, выведем длину.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

ИСХОДНИК РЕШЕНИЯ №18860516 ЗАДАЧИ №361			
Тест	Результат	Время	Память
1	Accepted	0,015	412 Кб
2	Accepted	0,015	412 Кб
3	Accepted	0,015	416 Кб
4	Accepted	0,015	420 Кб
5	Accepted	0,015	420 Кб
6	Accepted	0,015	412 Кб
7	Accepted	0,015	416 Кб
8	Accepted	0,015	412 Кб
9	Accepted	0,03	408 Кб
10	Accepted	0,015	416 Кб
11	Accepted	0,015	412 Кб
12	Accepted	0,015	412 Кб
13	Accepted	0,015	412 Кб
14	Accepted	0,015	416 Кб
15	Accepted	0,03	416 Кб
16	Accepted	0,015	424 Кб
17	Accepted	0,015	420 Кб
18	Accepted	0,015	420 Кб
19	Accepted	0,015	424 Кб
20	Accepted	0,015	424 Кб

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.796875 MB

Пример из задачи	0.000000 sec	0.796875 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.015000 sec	0.781250 MB

Вывод по задаче:

Обычный жадный алгоритм.

Задача №10-9. Преобразование ДНК

Биологи лаборатории Advanced Cellular Mechanics Lab. (ACM Lab.) занимаются исследованиями в области геномов и ДНК. Недавно в этой лаборатории была разработана технология, позволяющая достаточно дешево производить с цепочкой ДНК некоторые преобразования.

Представим себе цепочку ДНК как строку длины N из символов из множества $\{A, G, C, T\}$. Элементарное преобразование, которое умеют проводить биологи лаборатории, представляет собой разворот подстроки с L -ого по R -ый символ (целые числа L и R выбираются так, что $1 \leq L \leq R \leq N$). Таким образом, из строки $a_1a_2 \dots a_La_{L+1} \dots a_{R-1}a_R \dots a_N$ получается строка $a_1a_2 \dots a_Ra_{R-1} \dots a_{L+1}a_L \dots a_N$.

Теперь биологи разрабатывают аппаратно-программный комплекс для выполнения преобразований ДНК. Одной из его функций будет преобразование исходной цепочки ДНК в требуемую.

Ваша задача – написать программу, которая по исходной и требуемой цепочкам ДНК будет находить необходимую для этого цепочку элементарных преобразований.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
```



```

SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

cerr << fixed << setprecision(6);
cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
<< " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
" sec\n";
}

//-----

void reverseString(string &a, int l, int r) {
    string tmp = a;
    for (int i = 0; i < r - l + 1; i++) {
        a[l + i] = tmp[r - i];
    }
}

int main() {
    getFirstTime();

    // ---- code starts here ----
    string s, f;
    fin >> s >> f;

    int n = s.size();
    vector<pair<int, int> > ans;
    for (int i = 0; i < n - 1; ++i) {
        for (int j = i + 1; j < n; ++j) {
            if (s[j] == f[i]) {
                ans.emplace_back(i + 1, j + 1);
                reverseString(s, i, j);
                break;
            }
        }
    }

    fout << ans.size() << '\n';
    for (auto i: ans)
        fout << i.first << ' ' << i.second << '\n';

    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

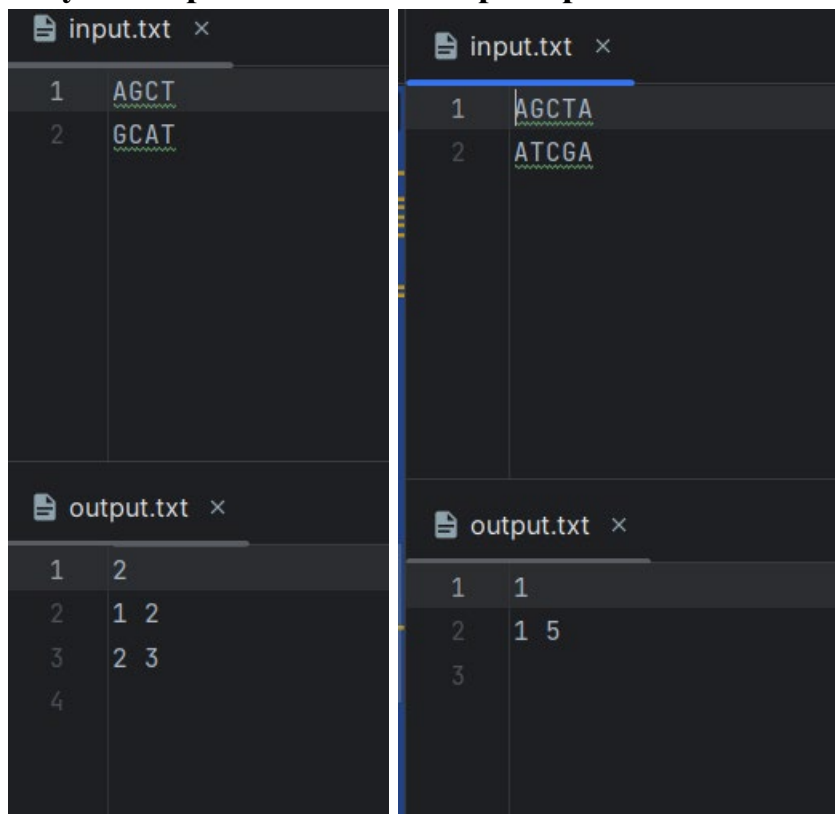
    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Переберем индексы, если у нас символы совпадают, но находятся в разных местах, зареверсим строку.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

1

#include <iostream>

2

#include <fstream>

3

#include <vector>

4

5

using namespace std;

6

7

void reverseString(string &a, int l, int r) {

8

string tmp = a;

9

for (int i = 0; i < r - l + 1; i++) {

10

a[l + i] = tmp[r - i];

11

}

12

}

13

14

int main() {

15

string s, f;

16

cin >> s >> f;

17

18

int n = s.size();

19

vector<pair<int, int> > ans;

20

for (int i = 0; i < n - 1; ++i) {

21

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

22

if (s[j] == f[i]) {

23

ans.emplace_back(i + 1, j + 1);

24

reverseString(s, i, j);

25

break;

26

}

27

}

28

}

29

30

cout << ans.size() << '\n';

31

for (auto i: ans)

32

cout << i.first << ' ' << i.second << '\n';

33

return 0;

34

}

Размер кода: 425

Посылки решений:

ID	Дата	Язык	Результат	Тест	Время	Память
18860524	21.02.2023 5:07:02	C++	Accepted		0.03	644 Кб

1

Accepted

0,015

408 Кб

2

Accepted

0,015

436 Кб

3

Accepted

0,015

524 Кб

4

Accepted

0,015

428 Кб

5

Accepted

0,03

508 Кб

6

Accepted

0,015

424 Кб

7

Accepted

0,015

632 Кб

8

Accepted

0,03

632 Кб

9

Accepted

0,03

640 Кб

10

Accepted

0,015

568 Кб

11

Accepted

0,015

508 Кб

12

Accepted

0,03

636 Кб

13

Accepted

0,03

512 Кб

14

Accepted

0,015

424 Кб

15

Accepted

0,015

448 Кб

16

Accepted

0,015

424 Кб

17

Accepted

0,015

512 Кб

18

Accepted

0,015

508 Кб

19

Accepted

0,015

412 Кб

20

Accepted

0,015

408 Кб

21

Accepted

0,015

632 Кб

22

Accepted

0,015

520 Кб

23

Accepted

0,015

512 Кб

24

Accepted

0,015

516 Кб

25

Accepted

0,015

508 Кб

26

Accepted

0,015

524 Кб

27

Accepted

0,015

520 Кб

28

Accepted

0,015

452 Кб

29

Accepted

0,015

628 Кб

30

Accepted

0,015

644 Кб

31

Accepted

0,015

440 Кб

32

Accepted

0,015

640 Кб

	Время выполнения	Затраты памяти
Нижняя граница	0.000000 sec	0.781250 MB

диапазона значений входных данных из текста задачи		
Пример из задачи	0.000000 sec	0.796875 MB
Пример из задачи	0.000000 sec	0.796875 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.015000 sec	0.796875 MB

Вывод по задаче:

Тоже легкая задача на жадные алгоритмы.

Задача №10-10. Abracadabra

Строка s называется супрефиксом для строки t , если t начинается с s и заканчивается на s . Например, «abra» является супрефиксом для строки «abracadabra». В частности, сама строка t является своим супрефиксом. Супрефиксы играют важную роль в различных алгоритмах на строках.

В этой задаче требуется решить обратную задачу о поиске супрефикса, которая заключается в следующем. Задан словарь, содержащий n слов t_1, t_2, \dots, t_n и набор из m строк-образцов s_1, s_2, \dots, s_m . Необходимо для каждой строки-образца из заданного набора найти количество слов в словаре, для которых эта строка-образец является супрефиксом.

Требуется написать программу, которая по заданному числу n , n словам словаря t_1, t_2, \dots, t_n , заданному числу m и m строкам-образцам s_1, s_2, \dots, s_m вычислит для каждой строки-образца количество слов из словаря, для которых эта строка-образец является супрефиксом.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <map>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}
```

```

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

void reverseString(string &a, int l, int r) {
    string tmp = a;
    for (int i = 0; i < r - l + 1; i++) {
        a[l + i] = tmp[r - i];
    }
}

int main() {
    getFirstTime();

    // ---- code starts here ----
    int n;
    fin >> n;

    vector<string> words(n);
    for (auto &i: words)
        fin >> i;

    int m;
    fin >> m;
    vector<string> patterns(m);
    for (auto &i: patterns)
        fin >> i;

    map<string, int> howmany;

    for (auto &i: words) {
        for (int length = 1; length <= i.size(); ++length) {
            string prefix = i.substr(0, length),
                suffix = i.substr(i.size() - length, length);

            if (prefix != suffix)
                continue;

            howmany[prefix]++;
        }
    }

    for (auto &i: patterns)
        fout << howmany[i] << '\n';

    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
}

```

```

    return 0;
}

```

Текстовое объяснение решения:

Переберем все возможные префиксы для всех строк, запишем их в мапку, а потом выведем количество для каждого паттерна.

Результат работы кода на примерах из текста задачи:

input.txt ×

```

1      4
2  abacaba
3  abracadabra
4  aa
5  abra
6  3
7  a
8  abra
9  abac

```

output.txt ×

```

1  4
2  2
3  0
4

```

Проверка задачи на (openedu, астр и тд при наличии в задаче):

ИСХОДНИК РЕШЕНИЯ №18860543 ЗАДАЧИ №857

перейти к задаче

Редктировать решение

```

1 | #include <iostream>
2 | #include <fstream>
3 | #include <vector>
4 | #include <ext/pb_ds/assoc_container.hpp>
5 |
6 | using namespace std;
7 |
8 | int main() {
9 |     int n;
10 |    cin >> n;
11 |
12 |    vector<string> words (n);
13 |    for(auto &i : words)
14 |        cin >> i;
15 |
16 |    int m;
17 |    cin >> m;
18 |    vector<string> patterns (m);
19 |    for(auto &i : patterns)
20 |        cin >> i;
21 |
22 |    map<string, int> howmany;
23 |
24 |    for(auto &i : words) {
25 |        for(int length = 1; length <= i.size(); ++ length) {
26 |            string prefix = i.substr(0, length),
27 |                suffix = i.substr(i.size() - length, length);
28 |
29 |            if (prefix != suffix)
30 |                continue;
31 |
32 |            howmany[prefix]++;
33 |        }
34 |    }
35 |
36 |    for(auto &i : patterns)
37 |        cout << howmany[i] << '\n';
38 |    return 0;
39 | }

```

Тест	Результат	Время	Память
1	Accepted	0,015	408 Кб
2	Accepted	0,015	440 Кб
3	Accepted	0,015	444 Кб
4	Accepted	0,03	440 Кб
5	Accepted	0,015	444 Кб
6	Accepted	0,015	448 Кб
7	Accepted	0,218	6,5 Мб
8	Accepted	1	5,8 Мб
9	Accepted	0,342	5,8 Мб
10	Accepted	1,186	5092 Кб
11	Accepted	0,656	4004 Кб
12	Accepted	0,936	7,9 Мб
13	Accepted	0,406	6,4 Мб
14	Accepted	0,436	6,5 Мб
15	Accepted	0,562	7,2 Мб
16	Accepted	0,468	7,3 Мб
17	Accepted	0,624	7,9 Мб
18	Accepted	0,406	5,5 Мб
19	Accepted	0,436	6,2 Мб
20	Accepted	0,436	6,2 Мб
21	Accepted	1,686	43 Мб
22	Accepted	1,968	44 Мб
23	Accepted	2,53	62 Мб

	Время выполнения	Затраты памяти
--	------------------	----------------

Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB
Пример из задачи	0.000000 sec	0.796875 MB
Верхняя граница диапазона значений входных данных из текста задачи	2.530000 sec	62 MB

Вывод по задаче:

Легкая задача на реализацию.

Вывод

Очень интересная лабораторная работа по строкам.