Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу «Дискретный анализ»

Студент: П.А. Земсков

Преподаватель: Н.К. Макаров

Группа: М8О-301Б

Дата: _____

Оценка: _____

Подпись: _____

Москва, 2025

Содержание

1 Лабораторная работа №5

Задача

Требуется разработать программу для поиска образцов в заранее известном тексте с использованием суффиксного массива.
Вариант алгоритма: Построение суффиксного массива с последующим поиском образцов методом бинарного поиска.

2 Описание

Суффиксным массивом (англ. suffix array, суфмасс) строки s называется перестановка индексов начал её суффиксов, которая задаёт порядок их лексикографической сортировки. Иными словами, чтобы его построить, нужно выполнить сортировку всех суффиксов заданной строки. Далее ниже приведен исходный код программы

3 Исходный код

Общая идея: Программа реализует поиск подстроки в тексте с использованием суффиксного массива. Сначала функция buildSuffixArray строит массив суффиксов строки, где каждый элемент представляет индекс начала суффикса. Алгоритм работает за O(n $\log^2$ n): сначала сортируются суффиксы по первым символам, затем итеративно происходит удвоение длины сравниваемых префиксов. После построения массива суффиксов поиск подстроки выполняется бинарным поиском по этому массиву. Функции lowerBound и upperBound находят диапазон суффиксов, начинающихся с заданного шаблона. Разность этих индексов даёт количество вхождений подстроки, а сами позиции определяются как sa[i] + 1. Таким образом, поиск одной подстроки имеет сложность O(m log n). Главная идея алгоритма заключается в том, чтобы один раз построить индекс для текста, после чего искать любые подстроки значительно быстрее, чем при использовании линейного поиска.

Листинг 1: Исходный код

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

std::vector<int> buildSuffixArray(const std::string &s)
{
    int n = s.length();
    std::vector<int> sa(n), rank(n);
    for (int i = 0; i < n; ++i)
    {
        sa[i] = i;
        rank[i] = static_cast<unsigned char>(s[i]);
    }
    for (int k = 1; k < n; k <<= 1)
    {
        std::stable_sort(sa.begin(), sa.end(), [&](int a, int b)
                         {
            if (rank[a] != rank[b]) return rank[a] < rank[b];
            int ra = (a + k < n) ? rank[a + k] : -1;
            int rb = (b + k < n) ? rank[b + k] : -1;
            return ra < rb; });
        std::vector<int> new_rank(n);
        new_rank[sa[0]] = 0;
        for (int i = 1; i < n; ++i)
        {
            int a = sa[i - 1], b = sa[i];
            int ra0 = rank[a], rb0 = rank[b];
            int ra1 = (a + k < n) ? rank[a + k] : -1;
            int rb1 = (b + k < n) ? rank[b + k] : -1;
            new_rank[b] = new_rank[a] + (ra0 != rb0 || ra1 != rb1);
        }
        rank.swap(new_rank);
        if (rank[sa[n - 1]] == n - 1)
            break;
    }
    return sa;
}

int lowerBound(const std::string &text, const std::string &pattern,
    const std::vector<int> &sa)
{
    int n = (int)text.size();
    int L = 0, R = n;
    while (L < R)
    {
        int M = (L + R) >> 1;
        int cmp = text.compare(sa[M], pattern.size(), pattern);
        if (cmp < 0)
            L = M + 1;
```

```
50          else
51              R = M;
52      }
53      return L;
54  }
55
56  int upperBound(const std::string &text, const std::string &pattern,
        const std::vector<int> &sa)
57  {
58      int n = (int)text.size();
59      int L = 0, R = n;
60      while (L < R)
61      {
62          int M = (L + R) >> 1;
63          int cmp = text.compare(sa[M], pattern.size(), pattern);
64          if (cmp <= 0)
65              L = M + 1;
66          else
67              R = M;
68      }
69      return L;
70  }
71
72  int main()
73  {
74      std::ios::sync_with_stdio(false);
75      std::cin.tie(nullptr);
76
77      std::string text;
78      if (!std::getline(std::cin, text))
79          return 0;
80
81      std::vector<int> sa = buildSuffixArray(text);
82
83      std::string pattern;
84      int qid = 0;
85      while (std::getline(std::cin, pattern))
86      {
87          ++qid;
88
89          int lb = lowerBound(text, pattern, sa);
90          int ub = upperBound(text, pattern, sa);
91
92          if (lb >= ub)
93              continue;
94
95          std::vector<int> pos;
96          pos.reserve(ub - lb);
97          for (int i = lb; i < ub; ++i)
98              pos.push_back(sa[i] + 1);
99          std::sort(pos.begin(), pos.end());
```

```
100
101         std::cout << qid << ":␣";
102         for (size_t i = 0; i < pos.size(); ++i)
103         {
104             if (i)
105                 std::cout << ",␣";
106             std::cout << pos[i];
107         }
108         std::cout << "\n";
109     }
110     return 0;
111 }
```

4 Консоль

Пример компиляции и демонстрация работы программы:

```
C:\Users\jocke\Documents\diskran\lab5> g++ --std=c++20 main.cpp -o main
C:\Users\jocke\Documents\diskran\lab5> ./main
abcdabc
abcd
bcd
bc
1: 1
2: 2
3: 2, 6
```

5 Тест производительности

При сравнении производительности с std::string::find результаты показывают, что для одного поиска стандартный метод быстрее, так как имеет меньшие константы и линейную сложность O(n·m).

```
C:\Users\jocke\Documents\diskran> g++ -std=c++20 lab5.cpp -o main
C:\Users\jocke\Documents\diskran> g++ -std=c++20 benchmark.cpp -o benchmark
C:\Users\jocke\Documents\diskran> .\benchmark
Search in suffix array time: 237 ms
std::find time: 4783 ms
```

Результат:

Однако при множественных запросах суффиксный массив выигрывает за счёт того, что индекс строится один раз, а каждый последующий поиск выполняется в логарифмическое время. Например, при тексте длиной 10 символов и 1000 запросов std::find требует около 5 секунд, тогда как поиск с использованием суффиксного массива занимает около 0.2 секунды.

6 Выводы

Суффиксный массив эффективен при большом количестве запросов к одному и тому же тексту, в то время как для одиночного поиска выгоднее применять std::find.

7 Список литературы

1. Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К. *Алгоритмы: построение и анализ. 3-е изд.* – М.: Издательский дом «Вильямс», 2010.

2. Knuth D. E. *The Art of Computer Programming. Vol. 3: Sorting and Searching. 2nd ed.* – Addison-Wesley, 1998.

3. *Суффиксный массив — Алгоритмика.* `https://ru.algorithmica.org/cs/string-structures/suffix-array`.