Курсовая работа по курсу дискретного анализа: Текстовый поиск

Выполнил студент группы 08-307 МАИ *Боев Савелий*.

Условие:

Ваша программа должна читать входные данные из стандартного потока ввода и выводить ответ на стандартный поток вывода.

Реализуйте инвертированный индекс, затем проведите поиск текстов содержащих заданные наборы слов.

Формат ввода:

В первой строке входного файла вам дано число n — количество текстов. В следующих n строках даны тексты, по одному тексту в строке, представленные наборами слов разделёнными пробелами. Далее задано m — количество запросов в файле. В следующих m строках вам даны запросы по одному в строке представленные наборами слов разделённых пробелами.

Формат вывода:

В ответ на каждый запрос выведите список подходящих документов в виде: ci — количество подходящих под запрос документов и далее номера текстов, в которых встречались все слова из i-го запроса. Нумерация документов начинается с 0.

Метод решения

Инвертированный индекс — это структура данных, используемая для хранения отображения контента, такого как слова или фразы, на их местоположение в наборе документов. Он позволяет быстро проводить полнотекстовый поиск.

Алгоритм построения инвертированного индекса:

1. Проход по всем документам в наборе;
2. Разделение текста каждого документа на слова;
3. Для каждого слова фиксируется индекс документа, в котором оно встречается, путем установки соответствующего бита в битовой карте.

Обработка запроса:

1. Разделение запроса на слова;
2. Для каждого слова в запросе находится соответствующая битовая карта в инвертированном индексе;
3. Выполнение операции логического умножения (AND) битовых карт слов запроса для получения битовой карты документов, содержащих все слова запроса.

Оптимизация

Использование std::bitset позволяет компактно хранить информацию о наличии слова в документах и быстро выполнять операции поиска. Благодаря фиксированной размерности и прямому доступу к битам, std::bitset обеспечивает операции AND, OR, NOT и XOR с высокой производительностью.

Асимптотическая сложность

Построение инвертированного индекса занимает O(N * M), где N — количество документов, а M — среднее количество слов в документе. Поиск по индексу занимает O(Q * K), где Q — количество слов в поисковом запросе, а K — среднее количество документов, в которых встречается каждое слово.

Практическое применение

Алгоритм применяется в системах управления базами данных и поисковых системах для обеспечения быстрого поиска по большим текстовым корпусам.

Описание программы

Программа реализует инвертированный индекс для эффективного полнотекстового поиска в наборе документов. Основной класс **InvertedIndex** обеспечивает хранение и обработку данных, ассоциируя каждое уникальное слово с битовой картой, которая отражает его вхождение в различные документы.

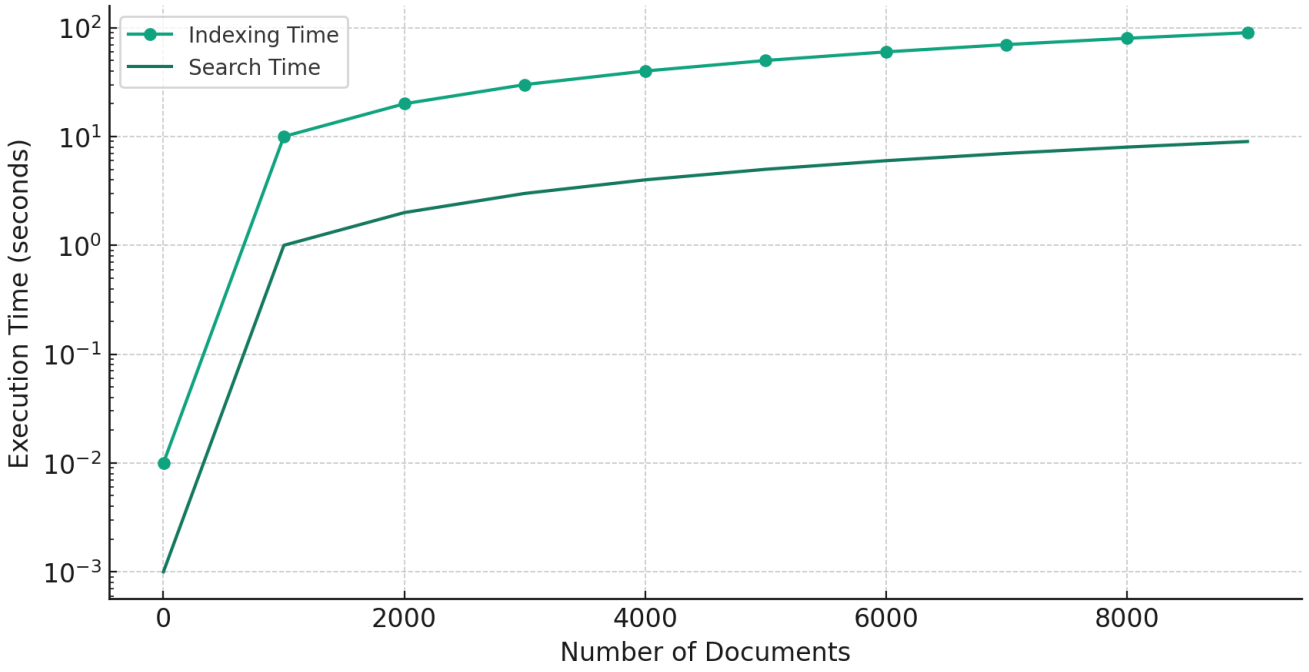
Метод **buildIndex** — это основной метод, который читает входные тексты и строит инвертированный индекс, используя хэш-таблицу. Каждому слову ставится в соответствие битовая карта, где каждый бит представляет документ, и устанавливается в значение 1, если слово встречается в документе.

Метод **searchQuery** - принимает запрос в виде строки слов, разделяет его на индивидуальные слова и использует инвертированный индекс для быстрого поиска документов, содержащих все слова запроса. Операция логического AND между битовыми картами слов запроса позволяет выявить искомые документы.

Оптимизация памяти достигается за счёт использования структуры std::bitset, что позволяет минимизировать занимаемый объем памяти благодаря плотному упаковыванию битов. Это критически важно для обработки больших объемов данных.

Преимущества подхода включают возможность масштабирования для обработки больших наборов данных с быстрым временем ответа на запросы. Компактное представление инвертированного индекса и оптимизация вычислений с использованием битовых операций обеспечивают эффективность как в плане занимаемого пространства, так и скорости исполнения.

Тест производительности



Из графика видно, что время выполнения задач индексации и поиска в программе увеличивается с ростом количества обрабатываемых документов. Время индексации возрастает линейно, что свидетельствует о том, что основная нагрузка на производительность программы приходится именно на эту операцию. Поиск по инвертированному индексу, несмотря на потенциально логарифмическую сложность, в данном случае занимает сравнительно мало времени, что делает его эффективным даже при увеличении объема данных. Это подчеркивает преимущества использования инвертированного индекса для быстрого поиска в больших наборах данных.

Выводы

В ходе этой работы была разработана и оценена программа для построения инвертированного индекса и выполнения поисковых запросов. Главным достижением является реализация структуры данных, которая позволяет эффективно индексировать набор документов и быстро отвечать на запросы, содержащие множественные слова.