### Московский авиационный институт

**Факультет прикладной математики и физики**

**Лабораторная работа №7**

**по курсу:**

**«Информационный поиск»**

**по теме:**

**«Прыжки по координатному индексу»**

**2 семестр**

Студент: Ахмед С. Х.

Преподаватель: Калинин А. Л.

Группа: 8О-106М

**Москва, 2019 г**

Постановка задачи

В полученный в предыдущих лабораторных работах индекс нужно добавить специальную информацию, позволяющую выполнить «прыжки по индексу», чтобы ускорить пересечение высокочастотных терминов.

Ход решения

В Python довольно сложно реализовывать алгоритмы строго завязанные на указатели. В силу этого было решено использовать некоторую альтернативу в виде рабоытс индексами. Сама суть алгоритма проста: вместо проходов по всему списку(а для высоко-частотных терминов это очень большие списки) мы разделим наш список на блоки, и будет прыгать через те блоки, в которых врядл и мы встретим второй термин. Для реализации прыжка в питоне удобно брать число, которое обозначает расстояние на которое можно прыгнуть. Оно символизирует количество элементов, через которое в идеале можно перепрыгнуть, чтобы получить пересечение. В питоне это относительно просто, так как в моей реализации( словаря ) мне просто нужно итерироваться по массиву ключей, и между ними нет значительных промежутков.

В качестве базового размера прыжка я решил взял корень квадратный от размера списка. На этом размере очень заметно существенное увеличение пересечения двух высокочастотных терминов. Однако разница между стандартной реализацией по производительности заметна, стандартная реализация быстрее на высокочастотных запросах. Что касается размера прыжка, то здесь очевидна тенденция увеличения времени выполнения, в тоже самое время, уменьшается риск пропуска значений, при увеличении прыжка время уменьшается, но увеличивается риск пропуска значений.

Побитовая схема для такого способа представления:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  | … |  |  | … |  | | |
| M байтов – (Сжатое) количество документов (Len(doc\_ids)) | | | | K байт – doc\_id(сжатое) | | | | P байтов (сжатое) – количество координат (N) | | | | N байт – координаты (сжаты) | | | L байт – смещение по координатному блоку (длина прыжка = jump), повторяется каждые jump позиций | | | |  | | |  | |  | |  | |  | | |  | | N байт – координаты (сжаты) | |  | |  | |
|  | | | | Повторение блока Len(doc\_ids) раз | | | | | | | | | | |  | | |  | | |  | |  | |  | |  | |  |  | |  | |  | | Повторение блока Len(doc\_ids) раз | |  | |  |

Примеры запросов

|  |  |  |
| --- | --- | --- |
| Запрос | Время стандартной библиотеки | Время пересечения с прыжками |
| Надо && или | 1.02 ms | 9.22 ms |
| До && про | 2.99 ms | 11.4 ms |
| После && когда | 2.99 ms | 11.6 ms |
| правительством || (феврале || (года (вступил && силу))) | 80 ms | 120 ms |

Почему так: так как я изначально фиксировал длину прыжка(длина прыжка есть ничто иное как корень квадратный от числа элементов в соответствующем поле), то хранить саму длину мне не имеет смысла, а имеет смысл хранить бит, символизирующий нужно ли делать прыжок для данного термина, индекса документа и позиции. Что касается проверки корректности, после добавления информации о прыжках, результат не должен измениться (сравнивается с тем же булевым поиском), так это всего лишь оптимизация над пересечением, которое не должно менять свой результат