Проекты

Алгоритмы-2021/07

Базовая информация

- Проект можно делать, а можно не делать. Например, доделывать в течение месяца домашние работы.

- Проект, в общем-то, нужен для знаний, если нравится тема, или если хочется добавить что-то интересное в портфолио (на свой GitHub)

Организационное

- На проекты запланирован месяц живого времени и 4 занятия, это занятие - первое.
- Промежуточные занятия - это встречи по поводу проектов. Если есть необходимость: задать вопросы по проекту и удобнее решить их в таком формате
- Также запланирована промежуточная встреча с презентацией результатов. Но это по желанию - можно и защитить в личном порядке (т.к. курс не про защиту проектов, а про алгоритмы)

Темы проектов

Михаил Горшков

1. Мини-поисковик по базе текстов
2. Сервис саджестов. Работает по аналогии с поисковой строкой в Яндексе или Гугле
3. Кастомная хэш-таблица
4. Менеджер памяти и garbage collector
5. Индексы для СУБД. Можно реализовать один из индексов: B-Tree, LSM

Мини-поисковик по базе текстов

- Софт (индексатор) индексирует файлы и создает индекс для последующего использования поисковой частью. При изменении текста (текстов) необходимо переиндексировать корпус.

- Поисковик принимает на вход слово (фразу) и выдает тексты, в которых слово (фраза) встречаются. Опционально сделать поддержку словоформ. Вводишь “собака“, поисковик выводит все тексты со словом “собака“, “собаки“, “Собакевич” и т.д.

- Техническую реализацию можно забрейнстормить прямо на вебинаре.

- Можно ранжирование

Сервис саджестов

- Работает по аналогии с поисковой строкой в Яндексе или Гугле. Задается список текстов. Они индексируются. Пишется софт, который по началу слова-фразы-предложения выдает все возможные продолжения, содержащиеся в этих текстах

- Можно усложнять, добавляя ранжирование

Кастомная хеш-таблица

- более-менее экзотический алгоритм хэширования, например, реализовать Cuckoo Hashing и сравнить его работу, скажем, со стандартной реализацией хэш-таблиц в вашем ЯП.

- Также можно взять любой другой не очень распространенный алгоритм (robin hood, … и т.д)

Менеджер памяти

- Менеджер памяти и garbage collector

- Менеджер памяти применяется для узкоспециализированный целей, скажем, в C++

- Можно реализовать менеджер памяти, построенный на базе одного из алгоритмов, рассмотренных на вебинаре по менеджменту памяти.

Индексы для СУБД

- Можно реализовать один из индексов: B-Tree, LSM
- Подробнее про LSM: “LSM-деревья (Log-structured merge-tree). Идея в том, что у LSM дерева есть несколько уровней хранения увеличивающихся размеров, первый (нулевой, он же memtable) из которых лежит в памяти, а остальные лежат на диске. Каждый уровень представляет собой дерево или список деревьев, и у каждого уровня есть растущий (обычно в 10 раз на уровень) лимит размера. Вставка производится сначала в нулевой уровень, при переполнении он сбрасывается в первый, при переполнении первого — попадает во второй… И так далее. Вставки получаются быстрые, так как не нужно мучаться с перезаписью отдельных мелких блоков данных. Однако зато мы, во-первых, имеем неслабый write amplification, а во-вторых — при поиске значений должны просматривать все деревья (что не круто). С первым ничего не поделаешь, а со вторым борются поддержанием в памяти bloom-фильтров, посмотрев в которые, можно сказать, в каких деревьях данных точно НЕТ, а в каких они МОГУТ БЫТЬ.”

База хешей
и расстояние Хэмминга

- Дано: база бинарных хэшей одного и того же размера, искомый хэш и расстояние.

- Нужно найти все хэши, отстоящие от заданного не более, чем на расстояние

База хешей
и расстояние Хэмминга

- Однако, есть алгоритм “мульти-индекс“, решающий эту задачу быстрее.

- Презентация для ознакомления: https://norouzi.github.io/research/posters/mih_poster.pdf

- Основная статья для ознакомления: https://www.cs.toronto.edu/~norouzi/research/papers/multi_index_hashing.pdf

- и его улучшенная версия: http://pages.di.unipi.it/rossano/wp-content/uploads/sites/7/2016/05/sigir16b.pdf

База хешей
и расстояние Хэмминга

- Идея алгоритма: каждый элемент датасета (то есть в нашем случае это хеши) разбивается на части (например, на байты), по которым отдельно строятся индексы, запрос разбивается на такие же части, затем производится поиск похожих по отдельным частям (пространство перебора гораздо меньше – самих данных меньше и расстояние ищется меньшее: профит) и результаты объединяются, затем удаляются лишние результаты (false positives появляются из-за особенностей алгоритма).

- Выпускной проект, связанный с вероятностными алгоритмами и структурами данных, в принципе, отличается от соответствующей домашки только обязательностью собственной реализации алгоритма. Подробное описание есть в вебинаре.

Евгений Волосатов

1. Архиватор: Huffman, LZW
2. Шифратор/дешифратор RSA
3. Шахматы – 3-фигурные таблицы решений
4. Пентамимо или Судоку на Dancing Links
5. Решение игры Сокобан
6. Олимпиадные задачи на leetcode codewars
7. Визуализация алгоритмов сортировки

Сокобан

- NP-сложная
- Большой фактор ветвления, глубина задачи
- Сложности составления солвера для задачи https://en.wikipedia.org/wiki/Sokoban#Scientific_research_on_Sokoban

Дальнейший план

- Подумать, какие проекты вам нравятся и окончательно определиться с темой.

- До промежуточной встречи нужно уже начать самостоятельно работать над проектом, подобрать необходимый материал и т.п.