

Лабораторная работа 1+2

1. (5 баллов) Про статическую оптимальность BST(binary search tree).

Реализовать алгоритм, который находит статическое оптимальное BST, то есть среди всех BST, найти BST с минимальным $cost$, где

$$cost(BST) = \sum_{i=1}^n p_i d_i + \sum_{i=0}^n q_i (d_i - 1),$$

d_i - глубина i -го ключа в BST (считаем, что корень имеет глубину 1),

p_i - вероятность успешного поиска ключа, q_i - вероятность неуспешного поиска ключа.

Input: $n, \{p_i\}_{i=1}^n, \{q_i\}_{i=0}^n$.

Output: $\min_{BST} cost(BST), BST = \operatorname{argmin}_{BST} cost(BST)$, где BST выводим in-order обходом.

Асимптотика: привести решение за $O(n^3)$ и $O(n^2)$.

Подсказка: https://peltorator.org/posts/knuth_yao_optimization/

2. (5 баллов) Scapegoat tree.

Реализовать Scapegoat tree, в котором присутствует следующая функциональность:

- `insert(key)`
- `delete(key)`
- `search(key)`
- вспомогательные методы для построения и перебалансировки дерева

Input: $n, \{key_i\}_{i=1}^n$.

Output: $insert(key_i), delete(key_i), search(key_i)$, построить графики зависимости времени

перебалансировки Scapegoat tree изменяя параметр “максимальное число вершин” в данном дереве (на занятиях обозначалось $m = 2n$, где n - число вершин в дереве) на разных заранее сгенерированных последовательностях операций вставки/удаления ключей дерева (продумать худшие и лучшие по асимптотике последовательности операций вставки/удаления ключей дерева), сделать анализ полученных зависимостей.

Подсказка: https://neerc.ifmo.ru/wiki/index.php?title=Scapegoat_Tree

3. (5 баллов) External quicksort.

Реализовать быструю сортировку, чтобы она работала во внешней памяти за оптимальное время $O(n/B \cdot \log_{M/B} n/B)$.

Подсказка: разобраться с

https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_%D0%B2%D0%BE_%D0%B2%D0%BD%D0%B5%D1%88%D0%BD%D0%B5%D0%B9_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D0%B8_%D0%91%D0%B0%D0%B7%D0%BE%D0%B2%D1%8B%D0%B5_%D0%BA%D0%BE%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D0%B8 и <https://www.geeksforgeeks.org/external-sorting/>

4. (5 баллов) Cashe-oblivious matrix multiplication.

Реализовать кеш-независимое перемножение матриц.

Подсказка: <https://en.algorithmica.org/hpc/external-memory/oblivious/>

5. (5 баллов) Connection to PostgreSQL.

Написать класс соединения к БД PostgreSQL и выполнить простейший SQL запрос.

Проанализировать план запроса. Подкрутить конфигурационные параметры БД на основе доп. видео.

6. (5 баллов) SS-tree / R-tree.


Реализовать по выбору либо R-tree, либо SS-tree, в котором присутствует следующая функциональность:

- insert (rectangle / sphere)
- delete (rectangle / sphere)
- search (rectangle / sphere)
- вспомогательные методы для построения и перебалансировки дерева

7. (5 баллов) MinHashLSH.

Реализовать поиск схожих объектов с помощью MinHashing + LSH(local sensitive hashing) на основе статьи из подсказки.

Подсказка: <https://nicods96.github.io/hi/fast-document-similarity-in-python/>



8. (5 баллов) FAISS.

С помощью FAISS из библиотеки langchain найти похожее предложение по заданному запросу.

Подсказка: см. занятие

