Общие условия

Программа должна быть реализована на одном из языков: C/C++, Java, C#, Python, Haskell, Scala. Программа должна быть платформонезависимой (для C# есть Mono), не иметь зависимостей от нестандартных библиотек и выполнена в виде консольного приложения.

Программа принимает входные данные в виде файла и записывает результат работы в файл. Формат файла студент задает самостоятельно. Имена входных и выходных файлов задаются через аргументы командной строки.

Если в условии требуется реализовать какую-либо структуру данных, то в программе должна присутствовать реализация операций вставки, поиска, удаления и других, специфичных для указанной структуры данных.

К программе должен прилагаться краткий отчет в формате PDF, содержащий:

- Теоретическую часть;
- Инструкцию по использованию;
- Краткое описание логики программы;
- Описание формата входных и выходных данных;
- Оценку сложности основных алгоритмов программы.

Выполненное домашнее задание, включающее:

- исходный код программы;
- несколько примеров, на которых можно протестировать программу;
- отчет

необходимо выслать на e-mail преподавателя не позднее, чем за неделю до конца семестра.

Варианты

1. Реализовать планировщик задач. Задачи находятся в нескольких постоянно обновляющихся очередях с разными приоритетами. Необходимо сформировать единую очередь задач.
2. Реализовать пул потоков. В качестве задач можно использовать заглушки из простейших арифметических операций.
3. Реализовать AVL-дерево.
4. Реализовать 2-3-дерево.
5. Реализовать AA-дерево.
6. Реализовать дерево со штрафами (scapegoat tree).
7. Реализовать алгоритм поиска пути A*.
8. Реализовать имитацию иерархической файловой системы на основе дерева.
9. Разработать алгоритм определения геолокации объекта: попадание точки в один из пересекающихся интервалов. Необходимо вернуть интервал с наибольшим весом. Вариант решения - через интервальное дерево.
10. Реализовать алгоритм сжатия по Хаффману. Опишите используемые структуры данных и обоснуйте их выбор.
11. Реализовать алгоритм сжатия на основе алгоритма расширяющегося префикса.
12. Реализовать кэш на основе дерева.
13. Реализовать алгорим Дейкстры с использованием кучи.
14. Реализовать ассоциативный массив на основе дерева.
15. Реализовать косое, красно-черное дерево и кучу. Сгенерировать набор тестов на вставку/удаление/поиск для различных случаев. Протестировать, предоставить результаты производительности.
16. Реализовать декартово дерево (treap).
17. Реализовать рандомизированное бинарное дерева поиска (Randomized binary search tree).
18. Реализовать список с пропусками.
19. Реализовать B-дерево.
20. Реализовать алгоритм для решения задачи формирования портфеля (на основе метода ветвей и границ).
21. Реализовать квадродерево.
22. Реализовать развёрнутый связный список.
23. Реализовать фибоначчиеву кучу.
24. Реализовать биномиальную кучу.