**Дополнительно задание на тему «Простейшие структуры данных»**

Ответьте *РАЗВЕРНУТО* на следующие вопросы. Документ с ответами загрузите в свой репозиторий, после чего создайте коммит с названием «C2 additional complete». Напишите мне в дискорде о выполнении, чтобы получить баллы.

1. Какие алгоритмы сортировки вам известны? В чем преимущество и недостатки каждого из них?

1. Пузырьковая сортировка - попарное сравнение значений, самое маленькое поднимается наверх, как “пузырек”, самое большое остается на “дне”.

* Простая в реализации;
* Быстро работает для почти отсортированных списков;
* Медленная, так как для n элементов нужно совершить n^2 операций.

1. Сортировка поиском - находится минимум или максимум массива (обычно минимум помещается в начало, максимум в конец), далее он сравнивается с не отсортированным элементами массива по очереди и меняется местами до тех пор, пока массив не будет отсортирован.

* Простая в реализации;
* Быстро сортирует небольшие списки;
* Медленно сортирует большие списки.

1. Сортировка вставками - массив перебирается слева направо, элемент размещается между ближайшими элементами с минимальным и максимальным значением.

* Простая в реализации;
* Стабильная сортировка небольших списков;
* Скорость O(n^2) в худшем случае.

1. Heap sort/Пирамидная сортировка - похожа на сортировку поиском, находим максимальный элемент и помещаем его в конец, повторяем это действие для неотсортированной части массива. Также, массив можно схематично представить в виде дерева, где у узла не больше 2 подузлов, при этом основной узел будет больше своих подузлов. Левые подузлы всегда меньше правых.

* Быстро сортирует большие списки;
* Не требует места для временных значений;
* Медленнее, чем сортировка слиянием.

1. Merge sort/Сортировка слиянием - обычно массив делится на части (до разделения попарно), потом все части массива сортируются отдельно, постепенно соединяясь в группы и сортируясь заново, пока массив не будет полностью отсортирован.

* Быстро сортирует большой объём данных;
* Требует место под временные значения;
* Медленнее, чем быстрая сортировка.

1. Quick sort/Быстрая сортировка - выбирается основной элемент, все значения меньшие значения основного элемента распределяются до него, большие - после него. К 2 этим группам применяется тот же алгоритм - выбирается элемент, создаются 2 группы по обе стороны от него и т.д.

* Быстро сортирует большой объём данных;
* Работает некорректно при большом количестве одинаковых значений;
* Сложная в реализации;
* Необходимо дополнительное стэковое пространство;
* Нестабильная сортировка.

2. Как устроена структура данных «Двоичное дерево поиска»? Какой интерфейс она предоставляет? Какие разновидности двоичных деревьев вам известны? В чем преимущества и недостатки каждой разновидности.

1. Двоичное дерево имеет структуру, в которой у каждого узла не более двух подузлов “детей”. Значение <= значению узла становится его левым подузлом или левым подузлом следующего узла, значение >= значению узла становится его правым подузлом или правым подузлом следующего узла.
2. Базовый интерфейс двоичного дерева поиска состоит из трёх операций: поиск узла, добавление узла в дерево, удаление узла.
3. 1) AVL дерево

* Ускоренный поиск из-за жёсткой балансировки;
* Более сложная реализация;
* После выполнения каждой операции вставки и удаления придется выполнять операцию проверки дерева на сбалансированность, и при обнаружении разбалансировки выполнять операции поворота узлов дерева.

2) Красно-чёрное дерево

* Более быстрые операции вставки и удаления, чем в AVL-деревьях;
* На хранение информации о высоте и балансовых коэффициентах расходуется лишь один бит информации;
* Большое время выполнения операции поиска.

3) B-дерево

* Хорошо подходят для структур данных;
* Хранится на диске, поэтому может занимать очень много места;
* Операции вставки и удаления очень медленные.