**Группа 11-810, 2019. Алгоритмы и структуры данных.**

**Семестровая # 1.**

1.    Найдите ваш алгоритм в сети. Адаптируйте его на C# с использованием входных данных в двух видов – массивов и LinkedList. +

2.   Приготовьте наборы случайно сгенерированных входных данных разного размера. Нужно около 50-100 наборов 100 до 10 000 элементов. Данные заготовить заранее, создавать их надо программно, при вызове алгоритма считывать их из файла перед выполнением самого алгоритма. +

3.   В алгоритм необходимо добавить два способа измерения времени: +

·         Измерение с помощью System.Now.Ticks (10,000,000 ticks per second). Разница между двумя вызовами соответствующих методов может показать вам время работы алгоритма, если первый вызов - до алгоритма, а второй - после. +

·         Измерение количество итераций. Проще всего - взять целочисленную переменную, считающую итерации самого вложенного цикла. Рекурсивный метод аналогично порождает последовательность итераций.

Измерять надо только время алгоритма (без ввода, без считываний). +

4. По полученным данным построить несколько графиков зависимости времени и итераций от размера входных данных (для массивов и коллекций). Графики можно строить в MS Excel. Для этого достаточно заготовить два столбца - табличное задание размера входа и количества итераций/шагов, а дальше можно загуглить. +

5. Дать оценку сложности данного алгоритма и доказать ее (наброском). Можно найти доказательство в интернете. После нарисовать примерный график оценки в теории и сравнить его с построенными графиками. Обосновать существенные различия, если они есть. +

6. Оформить отчет, содержащий: +

*1.       - название алгоритма, краткую историческую справку.*

*2.      - основной принцип работы. Основные особенности*

*3.      - оценка временной сложности с обоснованием (краткое доказательство)*

*4.      - таблицы полученных значений количества итераций и времени работы в зависимости от размера и характера данных (в конце презентации как приложение)*

*5.      - графики*

*6.   - выводы. Плюсы и минусы алгоритма, применимость его (т.е., когда его целесообразно применять).*

*7.      - список использованной литературы (источников)*

*8.     - Код алгоритма*

*9.      - Код генерации входных файлов для сортировки*

*10.  - Очень мелким шрифтом все входные данные*

7. Оформить презентацию, отражающую части  1-3, 5, 6 предыдущего пункта, с которой нужно будет выступить (возможно, онлайн).

**ВАРИАНТЫ**

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Algorithm** | **Assign** | **Review** |
| 1 | Merge sort | Бады Алтея |  |
| 2 | Tree sort | Валитов Константин |  |
| 3 | Timsort | Гимадиев Марсель |  |
| 4 | Bucket sort | Губаева Алсу |  |
| 5 | Stooge sort | Гусинников Дамир |  |
| 6 | Comb sort | Калашкин Никита |  |
| 7 | Heapsort | Квасников Игорь |  |
| 8 | Smoothsort | Константинов Владислав |  |
| 9 | Quicksort | Малов Илья |  |
| 10 | Introsort | Мигранов Эрнест |  |
| 11 | Patience sort | Мирошихина Екатерина |  |
| 12 | Shell sort | Мухамедьянов Райнур |  |
| 13 | Radix sort | Мырзин Валерий |  |
| 14 | Topological sort | Сафиуллин Альфред |  |
| 15 | Stackoverflow Sorting\* |  |  |
| 16 | Поиск подстрок. Алгоритм Кнута-Морриса-Пратта | Султанов Айнур |  |
| 17 | Поиск подстрок. Алгоритм Бойера-Мура | Сухова Алина |  |
| 18 | Кратчайший путь в графе. Алгоритм Флойда-Уоршелла | Хайбуллин Аскар |  |
| 19 | Кратчайший путь в графе. Алгоритм Беллмана-Форда | Хамматова Камила |  |
| 20 | Кратчайший путь в графе. Алгоритм Левита | Хищенко Анна |  |
| 21 | Построение минимального остовного дерева. Алгоритм Краскала | Чемкин Дмитрий |  |
| 22 | Построение минимального остовного дерева. Алгоритм Прима | Шамсутдинов Адель |  |
| 23 | Построение выпуклой оболочки. Алгоритм Грэхема | Шапиро Давид |  |
| 24 | Построение выпуклой оболочки. Алгоритм Джарвиса | Широков Роман |  |
| 25 | Построение выпуклой оболочки. Алгоритм Чана |  |  |

*16 апреля - DeadLine*