**Лабораторная работа №1.**

Реализовать следующие алгоритмы:

1. Левенштейн рекурсивный;
2. Левенштейн рекурсивный с кэшированием;
3. Домерау-Левенштейн рекурсивный;
4. Домерау-Левенштейн с матрицей.

Выполнить замеры процессорного времени для каждого из алгоритмов, а также потребление оперативной памяти в пике. Эксперименты следует проводить множество раз и усреднять полученные значения, помимо программы не должно быть запущенно никаких процессов.

**Лабораторная работа №2.**

Реализовать следующие алгоритмы:

1. Стандартный алгоритм умножения матриц;
2. Алгоритм Винограда;
3. Алгоритм Винограда, выполнив как минимум 3 оптимизации.

Выполнить теоретический расчет величины трудоемкости и замеры процессорного времени для каждого из алгоритмов и отобразить полученные результаты на графиках. Эксперименты следует проводить множество раз и усреднять полученные значения, помимо программы не должно быть запущенно никаких процессов. Следует обеспечить удобный интерфейс (меню) и возможность автоматической генерации матриц.

**Лабораторная работа №3.**

Выбрать и реализовать 3 любых не рекурсивных алгоритма сортировки массива, а также сделать расчёт величины трудоемкости:

1. для массива из случайных элементов (среднее значение);
2. для лучшего случая соответствующего алгоритма;
3. для худшего случая соответствующего алгоритма.

Выполнить замеры процессорного времени для каждого из алгоритмов аналогичным образом и отобразить полученные результаты на графиках. Эксперименты следует проводить множество раз и усреднять полученные значения, помимо программы не должно быть запущенно никаких процессов. Следует обеспечить удобный интерфейс (меню) и возможность автоматической генерации массивов.

**Лабораторная работа №4.**

Следует выбрать любой алгоритм (кроме умножения матриц) размерностью 15-19 строк и реализовать его последовательную и многопоточную версии (для многопоточной версии можно использовать только нативные потоки).

Выполнить замеры времени для каждой из реализаций алгоритма и отобразить полученные результаты на графиках. Эксперименты следует проводить множество раз и усреднять полученные значения, помимо программы не должно быть запущенно никаких процессов.

При анализе многопоточной версии алгоритма нужно варьировать количество потоков.

Следует обеспечить удобный интерфейс (меню) и возможность автоматической генерации данных для вычислений.

**Лабораторная работа №5.**

Следует выбрать алгоритм, состоящий как минимум из 3 независимых частей, на основе которого нужно спроектировать конвейерные вычисления (на каждую из частей алгоритма реализовать отдельную конвейерную ленту). Для данного моделирования использовать ЯП с нативными потоками или Go (goroutines).

Также необходимо реализовать линейную версию алгоритма (без многопоточности). Для оценки производительности алгоритмов и порядка выполнения действий нужно расставить временные метки для каждого из этапов (начало, конец) и вывести в лог-файл.

**Лабораторная работа №6.**

Следует изучить проблему коммивояжера и реализовать два алгоритма, решающих данную задачу: алгоритм полного перебора и муравьиный алгоритм. Для последнего необходимо произвести параметризацию как минимум по 3-ем параметрам и сравнить точность ответа с алгоритмом полного перебора. Таким образом, будут найдены оптимальные значения параметров.

Выполнить замеры времени для каждого из алгоритмов и отобразить полученные результаты на графиках. Эксперименты следует проводить множество раз и усреднять полученные значения, помимо программы не должно быть запущенно никаких процессов.

Следует обеспечить удобный интерфейс (меню) и возможность автоматической генерации данных для вычислений.

**Лабораторная работа №7.**

Следует рассмотреть проблему поиска записи в словаре. Для этого необходимо изучить и реализовать следующие алгоритмы: алгоритм полного перебора, алгоритм бинарного поиска и алгоритм частотного анализа.

Размер словаря – минимум 2000 записей. Словарь должен содержать реальные данные (нельзя использовать генераторы для заполнения случайными данными).

Выполнить сравнение алгоритмов по количеству сравнений для поиска каждого из ключей словаря. Полученные результаты отобразить на графиках: упорядочить по номеру ключа и по количеству сравнений (6 графиков – по 2 на каждый из алгоритмов).

Следует обеспечить удобный интерфейс (меню) и возможность автоматической генерации данных для вычислений.