Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет программной инженерии и компьютерной техники

Направление подготовки (специальность) СППО

ОТЧЕТ

о практической работе №4

По дисциплине: программирование на C++

Обучающийся Розмирский Д.В. P4114
(Фамилия И.О.) (номер группы)

Санкт-Петербург
2024 г.

Отчет по лабораторной работе №4

Введение

Целью работы было создать шаблонные классы для хранения разреженных векторов и 2D-матриц с использованием эффективных структур данных (хеш-таблиц), поддержать необходимые унарные и бинарные операции, включая транспонирование, сложение, умножение (вектор-вектор, матрица-матрица, матрица-вектор), обращение матрицы, возведение матрицы в степень (целочисленный и вещественный показатели), а также поэлементные операции со скаляром. Провести сравнение производительности с аналогичными операциями над данными, хранящимися в стандартных контейнерах (например, std::vector).

Задачи лабораторной работы

1. Разработать шаблонный класс SparseVector<T> для хранения разреженного вектора:
    - Использовать std::unordered_map для хранения ненулевых элементов, где ключ — индекс, значение — элемент.
    - Поддержать операции доступа к элементу, установки, удаления.
    - Реализовать унарные и бинарные операции: унарный минус, сложение, вычитание, умножение и деление на скаляр, поэлементное возведение в степень, скалярное произведение с другим вектором.
    - Реализовать операторы сравнения.
2. Разработать шаблонный класс SparseMatrix<T> для хранения разреженной матрицы:
    - Использовать std::unordered_map<std::pair<size_t,size_t>, T> для хранения ненулевых элементов.
    - Поддержать операции доступа к элементам, установки, удаления.
    - Реализовать операции: транспонирование, сложение матриц, умножение матриц, умножение матрицы на вектор.

- Реализовать обращение матрицы (через метод Гаусса-Жордана), возведение матрицы в степень (целочисленный показатель через двоичное возведение в степень, вещественный показатель через exp(p*log(A))), поэлементное возведение в степень.
- Проверять квадратность матрицы при операциях, где это необходимо (например, возведение в степень, обращение).

3. Реализовать аналогичные операции для плотных структур данных, хранящихся в std::vector:
   - Класс DenseVector<T> для плотного вектора.
   - Класс DenseMatrix<T> для плотной матрицы.
   - Аналогичные операции (где уместно), в частности умножение матриц, возведение в степень для сравнения производительности.
4. Провести тестирование:
   - Написать тесты с использованием assert для проверки корректности базовых операций над векторами и матрицами.
   - Проверить работу дополнительных операций (транспонирование, возведение в степень, обращение и т.д.).
5. Провести сравнение времени выполнения:
   - Измерить время выполнения операций для разреженных структур данных.
   - Измерить время выполнения тех же операций для плотных структур на основе std::vector.
   - Сравнить результаты и сделать выводы.

Описание реализации

Проект состоит из следующих файлов:

myVector.hpp — Заголовочный файл с реализацией класса SparseVector<T>.

myMatrix.hpp — Заголовочный файл с реализацией класса SparseMatrix<T>.

cpp_4_lab.cpp — Файл с тестами и сравнением производительности.

1. Реализация вектора:

- Реализован разреженный вектор SparseVector<T>, хранящий ненулевые элементы в std::unordered_map для быстрого доступа, а также в std::map для возможности упорядоченного обхода.
- Добавлены операции:
  - Унарный минус operator-().
  - Бинарные операции со скаляром (+, -, *, /).
  - Сложение, вычитание двух векторов.
  - Поэлементное возведение в степень powerAll.
  - Скалярное произведение dot.
  - Операторы сравнения (==, !=, <, >).

Таким образом, SparseVector предоставляет необходимые унарные и бинарные операции над векторами.

2. Реализация матрицы:

- Разреженная матрица SparseMatrix<T> хранит ненулевые элементы в std::unordered_map.

- Реализованы транспонирование, сложение с числом, сложение и вычитание матриц, умножение матрицы на вектор и на другую матрицу.

- Реализованы операции обращения матрицы inverse(), возведение в целочисленную степень integerPower(), в вещественную степень doublePower(), а также логарифм и экспонента матрицы, необходимые для doublePower().

- Реализован метод isSquare() для проверки, что матрица квадратная.

3. Основной файл:

- В testVectorRealis() проверены все основные операции вектора.
- В testMatrixRealis() проверены основные операции матрицы (транспонирование, сложение, умножение на вектор и т.д.).

- В testAdvancedMatrixOperations() проверены сложные операции (обращение, возведение в степень) для разреженной матрицы.
- Добавлен testDenseMatrixPower() для проверки возведения в степень плотной матрицы.

Сравнение производительности

В testAdvancedMatrixOperations() были добавлены замеры времени (std::chrono) для сравнения скорости:

- Умножения разреженной матрицы (SparseMatrix) на саму себя.
- Умножения плотной матрицы (DenseMatrix) на саму себя.
- Возведения в степень разреженной матрицы.

В результате выполнения замеров скорости работы стандартной матрицы и нашей реализации, получились следующие результаты:

```
All tests passed successfully!
All SparseMatrix tests passed successfully!
All advanced matrix operations tests passed successfully!
Sparse multiplication time: 0.0974084 s
Dense multiplication time: 1.43206 s
Sparse integerPower time: 2.50771 s
All performance tests done.
```

Выводы по сравнению производительности:

Все базовые тесты на корректность операций над векторами (SparseVector) и матрицами (SparseMatrix) успешно пройдены, что подтверждается выводом. Дополнительные операции (такие как обращение матрицы, возведение в степень) также протестированы и прошли без ошибок.

Умножение разреженной матрицы оказалось быстрее при данных параметрах (0.104168 s против 1.47468 s для плотной). Это говорит о том, что при достаточно разреженной матрице обход ненулевых элементов происходит быстрее, чем выполнение полной операции умножения со всеми элементами плотной матрицы.

Возведение в степень для разреженной матрицы заняло примерно 2.63 с.

Если бы мы измерили для плотной матрицы, скорее всего, при тех же размерах это заняло бы дольше, учитывая необходимость умножения большой плотной матрицы на саму себя несколько раз.

Однако, в данном выводе нет времени для плотной версии integerPower. Можно предположить, что для достаточно больших матриц при одинаковой размерности разреженный вариант может оказаться быстрее, если матрица действительно очень разреженная. Если же матрица плотная, тогда sparse-структура не даёт преимущества, а наоборот может замедлить из-за расходов на хеширование.

Вывод

Реализованы шаблонные классы для разреженного вектора и матрицы, использующие хеш-таблицы (std::unordered_map) и упорядоченные структуры (std::map) для хранения ненулевых элементов.

Реализован набор операций: транспонирование, сложение, умножение (вектор-вектор, матрица-вектор, матрица-матрица), обращение матрицы (inverse), возведение в степень (целочисленное и вещественное), поэлементные операции.

Все базовые тесты на корректность операций пройдены успешно. Проведено сравнение производительности. По результатам тестового запуска:

- Умножение разреженных матриц в тестовой конфигурации оказалось заметно быстрее, чем умножение плотных матриц.

- Разреженное возведение в степень заняло определённое время, однако для очень разреженных данных разреженная реализация будет работать быстрее или потреблять меньше памяти.

Таким образом, предложенный способ хранения разреженных данных показал преимущества в условиях малого числа ненулевых элементов. При высокой плотности ненулевых данных разреженный способ не даст выигрыша или будет менее эффективным, чем стандартный плотный контейнер std::vector из-за постоянного процесса рехэширования.

Исходный код:

https://github.com/RozmiDan/cpp_itmo_labs/tree/main/fth_lab_cpp