Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **ПРОГРАММНАЯ ИНЖЕНЕРИЯ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04** Программная инженерия

О Т Ч Е Т

По лабораторной работе № 3

Название: Обработка разреженных матриц

Дисциплина: Типы и структуры данных

Студент ИУ7-32Б (Группа) С.С. Беляк (И.О. Фамилия)

Преподаватель Барышникова М.Ю.

Москва, 2023

Описание условия задачи

Разреженная (содержащая много нулей) матрица хранится в форме 3-х объектов: - вектор A содержит значения ненулевых элементов; - вектор JA содержит номера столбцов для элементов вектора A; - вектор IA, в элементе Nk которого находится номер компонент в A и JA, с которых начинается описание строки Nk матрицы A.

1. Смоделировать операцию умножения матрицы и вектора-столбца, хранящегося в форме вектора A и вектора, содержащего номера строк ненулевых элементов, с получением результата в форме хранения вектора-столбца.
2. Произвести операцию умножения, применяя стандартный алгоритм работы с матрицами.
3. Сравнить время выполнения операций и объем памяти при использовании этих 2-х алгоритмов при различном проценте заполнения матриц.

Описание ТЗ

1. Описание исходных данных

Программа принимает следующие входные данные:

1. Пользователь выбирает опцию из меню, представленного в консоли.

Возможные варианты выбора:

- 1: умножить разреженную матрицу на вектор.
- 2: Умножить полную матрицу на вектор.
- 3: вывести матрицу.
- 4: сравнить алгоритмы матриц.
- 0: Выход из программы.

2. **Матрица и вектор: В зависимости от выбора пользователя, программа может потребовать следующие данные:**

- Матрица A:
    1. Ввод матрицы вручную:
       Пользователь должен ввести количество строк и столбцов, а затем значения элементов матрицы и их координаты (значение, строка, столбец). Матрица записывается как в разряженном виде в форме 3-х объектов, где вектор A содержит значения ненулевых элементов; - вектор JA содержит номера столбцов для элементов вектора A; - вектор IA, в элементе Nk которого находится номер компонент в A и JA, с которых начинается описание строки Nk матрицы A, так и в стандартном.
    2. Чтение матрицы из файла:
       Программа считывает матрицу из файла.
- Вектор:
  Пользователь вводит элементы вектора.
  Элементы вводятся после выбора действия и могут быть введены вручную.

Имеются следующие ограничения:

1. **Ограничение по формату ввода данных**: Программа может требовать, чтобы данные вводились в формате целых чисел.
2. **Ограничение по размеру матрицы**: Программа может поддерживать максимальный размер матрицы значением, которое равно 1000. Если размер матрицы превышает это значение, программа выдаст сообщение о том, что матрица слишком большая.
3. **Ограничение на ввод вектора:** Вектор, вводимый пользователем, также имеет ограничение на количество элементов. Программа требует положительное количество элементов в векторе, и она будет возвращать ошибку, если введенное количество элементов не соответствует этим требованиям.
4. **Ограничение на работу с файлами:** при попытке чтения матрицы из файла, программа предполагает, что файл имеет определенный формат данных, включая количество строк и столбцов.

2. Описание результатов программы

Ввод матрицы и вектора:
Матрица и вектор будут загружены в программу.
Умножение разреженной матрицы на вектор:
Если пользователь выбирает умножение разреженной матрицы на вектор, программа выполнит это умножение.
Умножение полной матрицы на вектор:
Если пользователь выбирает умножение полной матрицы на вектор, программа выполнит соответствующие операции.
Вывод матрицы:
Результирующая матрица умножения будет выведена на экран.
Сравнение алгоритмов матриц:
Если выбрана опция сравнения алгоритмов матриц, программа выполнит сравнение различных алгоритмов.

3. Описание задачи, реализуемой в программе

Цель работы - реализовать алгоритмы обработки разреженных матриц, сравнить эффективность использования этих алгоритмов (по времени выполнения и по требуемой памяти) со стандартными алгоритмами обработки матриц при различном процентном заполнении матриц ненулевыми значениями и при различных размерах матриц.

4. Способ обращения к программе

Способ обращения к программе пользователем происходит через исполняемый файл app.exe.

5. Описание возможных аварийных ситуаций и ошибок пользователя

1. **ERR_IO Ошибка ввода/вывода:** Код ошибки, который используется для обозначения ситуаций, связанных с некорректным вводом или выводом данных. Это может включать в себя некорректный формат ввода данных, прерывание пользователем процесса ввода, ошибки чтения или записи файлов и т. д.
2. **ERR_RANGE Превышение ограничений по размеру**: Код ошибки, который сообщает пользователю о том, что введенные данные превышают ограничения по размеру.
3. **ERR_MEM Ошибка с памятью:** Код ошибки, который указывает на проблемы с выделением или освобождением памяти.
4. **ERR_FILE Ошибка при открытии файла:** Код ошибки, связанный с невозможностью открыть или прочитать файлы. Это может быть вызвано отсутствием файла, отсутствием прав доступа или другими проблемами с файловой системой.
5. **ERR_INVALID_OPERATION Попытка выполнения невозможной операции:** Код ошибки, который сообщает пользователю, что попытка выполнения операции невозможна из-за некорректных данных или параметров операции.
6. **ERR_EMPTY_DATA Ввод нулевых данных:** Код ошибки, который указывает на то, что введены нулевые или пустые данные, и операция не может быть выполнена.
7. **ERR_TIMEOUT Превышение ограничений по времени**: Код ошибки, который указывает на то, что выполнение операции занимает слишком много времени, и оно было прервано из-за превышения временных ограничений.
8. **ERR_GENERAL Общая ошибка:** Код ошибки, который используется для обозначения других ошибок, связанных с внутренней логикой программы или неожиданными сбоями.

6.Описание внутренних структур данных

```
int *A = NULL, *JA = NULL, *IA = NULL;
int *B = NULL, *IB = NULL, *JB = NULL;
int *TValue = NULL, *TRowIndex = NULL, *TCol = NULL, *c = NULL;
```

Массивы матрицы A, JA, и IA:
int *A: Этот массив хранит значения ненулевых элементов матрицы. Его размер равен произведению количества строк и столбцов матрицы.
int *JA: Этот массив хранит столбцовые индексы ненулевых элементов матрицы. Размер этого массива также равен произведению количества строк и столбцов матрицы.
int *IA: Этот массив содержит индексы, указывающие на начало каждой строки в массиве A и JA. Размер этого массива равен количеству строк матрицы.

Массивы вектора и его индексов:
int *vec: Этот массив содержит элементы вектора-столбца. Его размер равен количеству элементов в векторе.
int *IB: Этот массив содержит индексы, указывающие на начало каждой строки в массиве vec. Размер этого массива равен количеству строк вектора.

Дополнительные массивы и указатели:
int *TValue, int *TRowIndex, int *TCol: Эти массивы используются для временного хранения значений, индексов строк и столбцов при выполнении умножения разреженной матрицы на вектор.
int *c: Этот массив хранит результат умножения матрицы на вектор-столбец и используется для хранения результата в форме вектора-столбца.

1. Алгоритм программы

Общий алгоритм программы:

1. Инициализация системы и выделение памяти.
2. Проверка выделения памяти.
3. Основной цикл программы начинается. Пользователю предоставляется меню с различными опциями, и программа ожидает ввода выбора пользователя.
4. В зависимости от выбора пользователя, программа выполняет следующие действия:
   - Ввод матрицы и вектора-столбца вручную или из файла.
   - Умножение матрицы A на вектор-столбец и получение результата в форме вектора-столбца.
   - Вывод результатов на экран.
   - Сравнение алгоритмов умножения матрицы и вектора.
   - Выход из программы.
5. После выполнения каждой операции программа возвращает пользователя в главное меню, где он может выбрать следующее действие.
6. При завершении программы освобождается выделенная память и программа завершает свою работу.

Функция multiply_sparse_matrix_to_vector:

1. Функция принимает указатели Val, Row, Col, vec, а также целые значения n и nv, и указатель на указатель c.
2. Выделяется память для массива c с nv элементами.
3. Внешний цикл проходит через строки матрицы (от 0 до n-1).
4. Внутренний цикл проходит через индексы Row[i] до Row[i+1] - 1 включительно.

5. Для каждого элемента Val[j], умножается на соответствующий элемент вектора vec[Col[j]] и суммируется с результатами для строки i.
6. Итоговая сумма для строки i сохраняется в массиве c[i]

Функция multiply_matrix_to_vector:

1. Функция принимает указатели A, IA, JA, v, а также целые значения n, m, nv, и указатель на указатель c.
2. Выделяется память для массива c с nv элементами.
3. Внешний цикл проходит через индексы i от 0 до n * m - 1.
4. Для каждого элемента A[i], умножается на соответствующий элемент вектора v[JA[i]] и суммируется с результатами для строки IA[i].
5. Итоговая сумма для строки IA[i] сохраняется в массиве c[IA[i]].

Функция input_matrix:

1. Функция принимает указатели n, m, A, JA, IA, а также указатели на len_i и ct.
2. Функция запрашивает у пользователя количество строк и столбцов для матрицы.
3. Выделяется память для массивов A, JA, IA и временных массивов tmp_val и tmp_j.
4. Далее начинается цикл, в котором происходит ввод чисел и их координат.
5. Ввод происходит в формате num i j, где num - значение элемента, i и j - его координаты.
6. Если координаты i или j выходят за пределы размеров матрицы, выводится сообщение об ошибке.
7. Значение num сохраняется в массив A, а его координаты в JA. Также сохраняется во временных массивах для вычисления IA.
8. После завершения ввода данных, вычисляется IA и сохраняется len_i и ct.
9. Временные массивы освобождаются.

Функция full_matrix:

1. Функция принимает указатели a, ia, ja, а также указатели на n и m.
2. Функция запрашивает у пользователя количество строк и столбцов для матрицы.
3. Выделяется память для массивов a, ia, ja.
4. Заполняется ia и ja для формы матрицы по координатам.
5. Далее начинается цикл для ввода значений элементов по координатам. Если ввод некорректен, происходит обработка ошибки.
6. После завершения ввода данных и вычисления координат, освобождается память.

Функция input_vector:

1. Функция принимает указатель vec и указатель n.
2. Запрашивает у пользователя количество элементов вектора и выделяет память для вектора.
3. После этого запрашивает ввод элементов вектора и сохраняет их в выделенной памяти.
4. В случае ошибки ввода, функция возвращает соответствующий код ошибки.

Функция read_full_matrix_from_file:

1. Функция принимает указатель на файл f, указатели a, ia, ja, а также указатели n и m.
2. Функция сначала считывает количество строк и столбцов матрицы из файла.
3. Затем выделяется память для массивов a, ia, ja.
4. Заполняются значения a, ia, ja из файла.
5. Если в процессе чтения произошла ошибка, функция возвращает соответствующий код ошибки.

Функция read_matrix_from_file:

1. Функция аналогична предыдущей, за исключением того, что она также вычисляет ia и сохраняет len_i и ct.

Функция print_matrix_to_vector:

1. Функция принимает указатель A и целое число n.
2. Выводит элементы вектора A в столбцовом формате.

Функция print_matrix:

Функция принимает указатель A, целые числа n и m.
Выводит элементы матрицы A в матричном формате.

Функция compare_matrix_operations():

1. Запрашивает у пользователя количество строк и столбцов матрицы.
2. Заполняет разреженную и стандартную матрицы случайными значениями с разными процентами заполнения.
3. Заполняет вектор случайными значениями.
4. Засекает время выполнения умножения разреженной матрицы на вектор и стандартной матрицы на вектор.
5. Выводит информацию о проценте заполнения, объеме памяти, времени выполнения для обоих операций

Вывод по алгоритму работающей программы:

Программа может включать в себя возможность ввода и обработки разреженных матриц и векторов, выполнения матричных операций, чтения данных из файла, а также сравнения времени выполнения различных операций на матрицах с разным процентом заполнения. Программа предоставляет удобный интерфейс для работы с данными матриц и позволяет оценить эффективность алгоритмов.

Тесты

Таблица положительных тестов.

Номер теста	Описание	Ожидаемый результат
1	Ввод матрицы вручную и умножение разреженной матрицы на вектор. Ввод разреженной матрицы с 3 строками и 3 столбцами: 3 3 1 0 0 2 0 2 3 2 1 Ввод вектора: 3 2 1 3	Ожидаемый результат: Успешное умножение матрицы на вектор и вывод результата: [8, 0, 3].
2	Чтение матрицы из файла и умножение разреженной матрицы на вектор. Ввод из файла разреженной матрицы с 3 строками и 3 столбцами: 3 3 1 0 2 0 0 0 0 3 0 Ввод вектора: 3 2 1 3	Ожидаемый результат: Успешное умножение матрицы на вектор и вывод результата: [8, 0, 3].

3	Ввод матрицы вручную и умножение полной матрицы на вектор. Ввод полной матрицы с 3 строками и 3 столбцами: 3 3 1 0 0 2 0 1 3 0 2 4 1 0 5 1 1 6 1 2 7 2 0 8 2 1 Ввод вектора: 3 2 1 3	Ожидаемый результат: Успешное умножение матрицы на вектор и вывод результата: [11 32 53]
4	Чтение матрицы из файла и умножение полной матрицы на вектор. Ввод из файла полной матрицы с 3 строками и 3 столбцами: 3 3 1 2 3 4 5 6 7 8 9 Ввод вектора: 3 2 1 3	Ожидаемый результат: Успешное умножение матрицы на вектор и вывод результата: [11 32 53]
5	Вывод матрицы на экран после умножения	Успешный вывод результата умножения матрицы на вектор.
6	Сравнение разных алгоритмов умножения матриц	Вывод результатов сравнения алгоритмов.

Таблица негативных тестов.

Номер теста	**Описание**	**Ожидаемый результат**
1	Ввод нецелочисленных данных для матрицы	Ошибка: Введены недопустимые символы.
2	Ввод нецелочисленных данных для вектора	Ошибка: Введены недопустимые символы.
3	Ввод некорректных данных для размерности матрицы	Ошибка: Некорректные размеры матрицы.
4	Ввод некорректных данных для размерности вектора	Ошибка: Некорректный размер вектора.
5	Попытка умножения разреженной матрицы на вектор с неправильными размерами	Ошибка: Размеры матрицы и вектора не совпадают.
6	Ввод матрицы с нулевой размерностью	Ошибка: Размеры матрицы должны быть положительными.
7	Попытка умножения полной матрицы на вектор с неправильными размерами	Ошибка: Размеры матрицы и вектора не совпадают.
8	Ввод нецелочисленных данных для выбора в меню	Ошибка: Введены недопустимые символы.
9	Ввод некорректных значений для выбора в меню	Ошибка: Некорректный выбор в меню.
10	Ввод несуществующего файла для чтения матрицы	Ошибка: Файл не найден.

Оценка эффективности

Эффективность использования разреженной матрицы перед полной матрицей при умножении матрицы на вектор зависит от структуры и размера матрицы, а также от конкретной задачи. В общем случае, разреженная матрица может иметь преимущества по сравнению с полной матрицей:

1. Экономия памяти: Разреженная матрица хранит только ненулевые элементы, что позволяет существенно снизить объем занимаемой памяти по сравнению с полной матрицей. Это особенно важно, если матрица большая и большая часть элементов равна нулю.
2. Ускорение умножения: при умножении разреженной матрицы на вектор можно исключить операции умножения на нулевые элементы, что уменьшает количество операций. Это может привести к ускорению вычислений.
3. Эффективность хранения: Структура формата предоставляет эффективный способ хранения и доступа к разреженной матрице, особенно в случаях, когда матрица имеет множество нулевых элементов в строчках.

Однако есть некоторые ограничения:

Избыточность структуры: в некоторых случаях, если матрица не очень разреженная, структура разреженной матрицы может стать менее эффективной с точки зрения памяти, чем полная матрица. Это происходит из-за дополнительных данных, которые необходимо хранить для индексации ненулевых элементов.

Итак, эффективность использования разреженной матрицы перед полной матрицей зависит от конкретной задачи и данных. Она может привести к экономии памяти и более быстрому умножению, но требует адаптации к конкретным условиям задачи.

Временная эффективность и затраты памяти

Сравнение времени

Найдем эффективность обработки разряженной матрицы в %, используя формулу Эффективность = (100-(100* Время выполнения разряженной матрицы / Время выполнения стандартной матрицы))

10% заполненности

Количество элементов	**Разреженная матрица, нс**	**Стандартная матрица, нс**
50*50	1646	13559
100*100	5884	81178
1000*1000	379139	3742096

Эффективность = (87.8 + 92.7 + 89.8) / 3 = 90%

30% заполненности

Количество элементов	**Разреженная матрица**	**Стандартная матрица**
50*50	2073	7596
100*100	13987	58446
1000*1000	1292582	14137491

Эффективность = (72.7 + 76 + 91) / 3 = 79.8%

50% заполненности

Количество элементов	**Разреженная матрица**	**Стандартная матрица**
50*50	3389	7642
100*100	22399	58902
1000*1000	1927508	7048426

Эффективность = (55.6 + 61.9 +72.7) / 3 = 63.4 %

80% заполненности

Количество элементов	Разреженная матрица	Стандартная матрица
50*50	5628	7667
100*100	20838	34657
1000*1000	2553208	8070788

Эффективность = (26.5 +39.8 + 68.3) / 3 = 44,8%

100% заполненности

Количество элементов	Разреженная матрица	Стандартная матрица
50*50	6429	8978
100*100	25507	41254
1000*1000	13765410	28148260

Эффективность = (28.4 + 38.2 + 51) / 3 = 39%

Сравнение памяти

10% заполненности

Количество элементов	Разреженная матрица, байт	Стандартная матрица, байт
50*50	1208	10000
100*100	4428	40000
1000*1000	437392	4000000

30% заполненности

Количество элементов	Разреженная матрица	Стандартная матрица
50*50	3016	10000
100*100	12096	40000
1000*1000	1228792	4000000

50% заполненности

Количество элементов	Разреженная матрица	Стандартная матрица
50*50	5112	10000
100*100	19928	40000
1000*1000	2019360	4000000

80% заполненности

Количество элементов	Разреженная матрица	Стандартная матрица
50*50	8152	10000
100*100	32128	40000
1000*1000	3208960	4000000

100% заполненности

Количество элементов	Разреженная матрица	Стандартная матрица
50*50	10000	10000
100*100	40000	40000
1000*1000	4000000	4000000

Вывод

В данной лабораторной работе было проведено сравнение времени выполнения операций умножения и объема памяти, используемого разреженными и стандартными матрицами. Мы исследовали различные уровни разреженности матриц и оценили их влияние на эффективность разреженных матриц.

Время выполнения операций над разреженными и стандартными матрицами зависит от размеров матрицы и уровня ее разреженности. Разреженные матрицы занимают гораздо меньше памяти по сравнению со стандартными матрицами, особенно при высоких уровнях разреженности.

Однако, эффективность разреженных матриц может уменьшаться с увеличением размера матрицы или уровня разреженности. Мой алгоритм обработки разреженных матриц оказался эффективнее обработки стандартных матриц, так, при 10 % заполненности разреженной матрицы мы получили алгоритм, который эффективнее по времени на 90 %, а при обработке 100 % заполненности разреженной матрицы, алгоритм эффективнее на 39%. Использование разреженной матрицы для выигрыша по времени с эффективностью более 50% оправдано при работе с матрицами заполненностью около 75%. Затраты по памяти меньше, чем при использовании алгоритма умножения стандартных матриц при любой разреженности.

Ответы на контрольные вопросы

1. Что такое разреженная матрица, какие схемы хранения таких матриц Вы знаете?

Разреженная матрица - это матрица, в которой большинство элементов равно нулю. Такие матрицы встречаются, например, в задачах обработки изображений, графов и систем линейных уравнений.

Существуют различные методы хранения элементов матрицы в памяти. Например, линейный связный список, т.е. последовательность ячеек, связанных в определенном порядке. Существует диагональная схема хранения симметричных матриц, а также - связные схемы разреженного хранения.

Связная схема хранения матриц, предложенная Кнутом, предлагает хранить в массиве (например, в AN) в произвольном порядке сами элементы, индексы строк и столбцов соответствующих элементов (например, в массивах I и J), номер (из массива AN) следующего ненулевого элемента, расположенного в матрице по строке (NR) и по столбцу (NC), а также номера элементов, с которых начинается строка (указатели для входа в строку – JR) и номера элементов, с которых начинается столбец (указатели для входа в столбец – JC.

Наиболее широко используемая схема хранения разреженных матриц - это схема, предложенная Чангом и Густавсоном, называемая: "разреженный строчный формат". Эта схема предъявляет минимальные требования к памяти и очень удобна при выполнении операций сложения, умножения матриц, перестановок строк и столбцов, транспонирования, решения систем линейных уравнений, при хранении коэффициентов в разреженных матрицах и т.п.

2. Каким образом и сколько памяти выделяется под хранение разреженной и обычной матрицы?

Для хранения разреженной матрицы выделяется значительно меньше памяти по сравнению с обычной матрицей, т.к. хранятся только ненулевые элементы. Количество памяти, которое выделяется для хранения разреженной матрицы, зависит от выбранной схемы хранения и структуры самой матрицы. Обычная матрица, в свою очередь, выделяет память под каждый элемент матрицы вне зависимости от его значения n*m*sizeof(int).

3. Каков принцип обработки разреженной матрицы?

Обработка разреженной матрицы отличается от обработки обычной матрицы. Основной принцип обработки разреженной матрицы - исключить операции с нулевыми элементами, чтобы снизить вычислительные затраты и использование памяти.

4. В каком случае для матриц эффективнее применять стандартные алгоритмы обработки матриц? От чего это зависит?

Стандартные алгоритмы обработки матриц эффективнее применять, когда матрица имеет почти все ненулевые элементы и размеры матрицы не очень велики. Это зависит от плотности разреженной матрицы - чем больше ненулевых элементов, тем менее эффективным будет использование схемы хранения разреженной матрицы.

Если матрица плотная или имеет малое количество ненулевых элементов, использование стандартных алгоритмов обработки матриц может быть более эффективным.