Лабораторная работа № 5

Методические указания

Обработка разреженных матриц

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.

Краткие теоретические сведения

В виде матриц достаточно широко представляется информация во многих областях человеческой жизнедеятельности. Матричные задачи часто используются при решении разреженных линейных алгебраических уравнений; разреженных обычных и обобщенных спектральных задач, при этом матрицы могут быть достаточно большие (больше 10^{10-20} элементов), а число ненулевых элементов при матрице порядка **n** может выражаться как $\mathbf{n}^{1+g}$, где $\mathbf{g < 1}$.

В подобных матричных задачах значения **g** лежат в интервале **0.2** … **0.5**, т.е. матрица разрежена. Но разреженность матрицы следует учитывать только в том случае, когда из этого можно извлечь выгоду за счет игнорирования нулевых элементов. На самом деле разреженную матрицу можно обрабатывать так же как плотную, и наоборот, плотную матрицу можно обрабатывать так же как разреженную. В обоих случаях получаются правильные числовые результаты, но вычислительные затраты в том или другом случае могут существенно различаться.

Алгоритмы обработки разреженных матриц предусматривают действия только с ненулевыми элементами и, таким образом, количество операций будет пропорционально количеству ненулевых элементов.

Существуют различные методы хранения элементов матрицы в памяти.

Например, линейный связный список, т.е. последовательность ячеек, связанных в определенном порядке. Каждая ячейка списка содержит элемент списка и указатель на положение следующей ячейки. Можно хранить матрицу, используя кольцевой связный список, двунаправленные стеки и очереди. 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). Данная схема хранения избыточна, но позволяет легко осуществлять любые операции с элементами матрицы.

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.

В этом случае значения ненулевых элементов хранятся в массиве AN, соответствующие им столбцовые индексы - в массиве JA. Кроме того, используется массив указателей, например IA, отмечающих позиции AN и JA, с которых начинаются описание очередной строки. Дополнительная компонента в IA содержит указатель первой свободной позиции в JA и AN.

Разреженный вектор - это разреженная матрица-строка или матрица-столбец, поэтому рассмотрим *скалярное* умножение разреженных векторов (как частный случай работы с матрицей) с использованием так называемого расширенного массива указателей IP. Например, есть два разреженных вектора **a** и **b** размером N. Значения векторов приведены в таблице.

индекс J	1	2	3	4	5	6	7	8	9	10	11	12
a		3	1.5		-0.4		4.4	-8				
b			1.2	-2.2	0.4						7	

Представление этих векторов в разреженном строчном формате будет следующим:

Индексы элементов вектора **a :** JA = 2; 7; 3; 8; 5.

Значения элементов вектора **a :** AN = 3; 4,4; 1,5 –8; -0,4.

Индексы элементов вектора **b :** JB = 4; 3; 11; 5.

Значения элементов вектора **b :** BN = -2,2; 1,2; 7; 0,4.

Данная схема хранения является упакованной, так как хранятся только ненулевые элементы, и неупорядоченной, так как номера индексов могут идти и не по возрастанию (что и показано в данном примере).

Допустим, необходимо вычислить скалярное произведение векторов **a** и **b:**

$$\mathbf{h} = \sum_{i=1}^{N} a_i \, b_i$$

При вычислении стандартным способом нужно N^2 просмотров массива. Для сокращения алгебраических операций удобно во время работы хранить расширенный (по размерности массивов **a** и **b**) массив указателей IP (его начальное состояние - нулевое). Этот массив заполняется путем одного просмотра массива JA

Алгоритм разреженного вычисления следующий:

1. Просматривается массив JA (первое значение - 2) и в соответствующий элемент массива IP (для данного примера - во второй элемент) вписывается индекс массива JA (т.е. - 1). Таким образом хранятся указатели позиций ненулевых элементов в AN. Получается массив указателей IP в виде

	1	2	3	4	5	6	7	8	9	10	11	12...
IP	0	1	3	0	5	0	2	4	0	0	0	0,

Где вторая строка – номер индекса в JA.

Отсюда видно, что второй, третий, пятый, седьмой и восьмой элементы исходного массива не являются нулевыми, а хранящийся во втором элементе IP индекс 1 указывает на то, что элемент **a2** хранится в первом элементе массива AN(1), а элемент **a7** – во втором элементе массива AN (2),

2. Просматривается массив JB. Если соответствующий элемент массива IP ненулевой, т.е. позиции элементов в векторах **a** и **b** совпадают, то вычисляются произведения **ai** * **bi** и накапливаем в сумму произведений **h** до тех пор, пока не будет исчерпан массив JB. В результате

JB(1) = 4, IP(4) = 0 - действий нет;

JB(2) = 3, IP(3) = 3.

Следовательно, третий элемент массива AN не является нулевым, т.е. AN(3) = 1,5, а BN(2) = 1,2. Получаем произведение BN(2) * AN(3) = 1,8 и накапливаем результат в сумму произведений **h**.

Количество операций в данном алгоритме пропорционально числу ненулевых элементов, не считая засылки N нулей в массив IP.

Применение этого алгоритма особенно эффективно в ситуации, когда вектор нужно скалярно умножить на несколько векторов. В этом случае массив IP заполняется только один раз и затем используется для вычисления всех требуемых скалярных произведений. Эта ситуация возникает при необходимости перемножить разреженную матрицу и разреженный вектор.

Задание

Разработать программу умножения или сложения разреженных матриц. Предусмотреть возможность ввода данных, как с клавиатуры, так и использование заранее подготовленных данных. Матрицы хранятся и выводятся в форме трех объектов. Для небольших матриц можно дополнительно вывести матрицу в виде матрицы. Величина матриц - любая (допустим, 1000*1000). Сравнить эффективность (по памяти и по времени выполнения) стандартных алгоритмов обработки матриц с алгоритмами обработки разреженных матриц при различной степени разреженности матриц.

Указания к выполнению работы

Все логически завершенные фрагменты алгоритма (ввод, вывод, обработка и т.п.) необходимо оформить как подпрограммы.

При разработке интерфейса программы следует предусмотреть:

- указание формата и диапазона вводимых данных,
- указание операции, производимой программой,
- наличие пояснений при выводе результата,
- указание формата выводимых данных
- возможность заполнения матриц вручную (даже при большой размерности, например, 1000*1000).

При тестировании программы необходимо:

- проверить правильность ввода
- проконтролировать правильность вывода данных (т.е. их соответствие требуемому формату);
- проверить правильность выполнения операций;
- обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);

- обеспечить вывод сообщений при нулевых результате или вывод нулевого результата при ненулевом входе;
- обеспечить возможность ввода данных и вывода результата как при малых матрицах, так и при больших (например, 1000 * 1000).
- сравнить время выполнения стандартного алгоритма обработки матриц и алгоритма обработки разреженных матриц при различной заполненности матриц.
- сравнить объем требуемой памяти для реализации стандартного алгоритма обработки матриц и алгоритма обработки разреженных матриц при различном проценте заполнения матриц.

Следует также протестировать программу при полной загрузке системы, то есть при полном заполнении матриц. Программа должна адекватно реагировать на неверный ввод, пустой ввод и выход за границы матрицы или вектора. Необходимо тщательно следить за освобождением динамической памяти при окончании программы.

Содержание отчета

Отчет по лабораторной работе должны содержать:

1) описание условия задачи;
2) описание ТЗ;
3) описание алгоритма (в любом виде);
4) набор тестов, с указанием, что проверяется;
5) таблицы с результатами измерений времени и памяти при различных используемых форматах хранения и алгоритмах обработки матриц в их различном процентном заполнении нулями.

На основе полученных измерений в отчете по лабораторной работе должен быть сделан **вывод,** о том, в каких случаях применения какого алгоритма и способа хранения целесообразно для обработки матрицы, какую выгоду дает тот или иной алгоритм. Обратить особое внимание на тестирование программы.

Отчет по лабораторной работе должен содержать ответы на следующие вопросы:

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

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

3. Каков принцип обработки разреженной матрицы?

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

Отчет представляется в электронном или печатном виде.

Список рекомендуемой литературы

1. *Писсанецки С.* Технология разреженных матриц.: Пер. с англ. - М.: Мир, 1988.
2. *Тьюарсон Р.* Разреженные матрицы.: Пер. с англ. - М.: Мир, 1977.