МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Институт №8 "Информационные технологий и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

Курсовая работа
по курсу "Вычислительные системы"
2 семестр

Задание 7
Разреженные матрицы

Автор работы:
студент 1 курса, группы М8О-106Б-20
Леухин М. В.
Преподаватель:
Дубини А. В.

Москва, 2021

Содержание

1 Постановка задачи

Задание: Составить программмму на языке Си с функциями для обработки прямоугольных разреженных матрциц с элементами целого типпа, которая:

1. Вводит матрицы различного размера, представленные во входном текстовом файле в обычном формате (по строкам), с одновременным размещением ненулевых элементов в разреженной матрице в соответствии с заданной схемой.

2. Печатает введённые матрицы во внутреннем представлении согласно заданной схеме размещения и в обычном (естественном) виде.

3. Выполняет необходимые преобразования разреженных матриц (или вычисления над ними) путём обращения к соответствующим функциям

4. Печатает результат преобразования (вычисления) согласно заданной схеме размещения и в обычном виде.

В процедурах и функциях предусмотреть проверки и печать сообщений в случаях ошибок в задании параметров. Для отладки использовать матрицы, содержащие 5–10% ненулевых элементов с максимальным числом элементов 100.

Вариант схемы размещения матриц: один вектор. Ненулевому элементу соответствуют две ячейки: первая содержит номер столбца, вторая содержит значение элемента. "-1"в первой ячейки означает конец строки, а вторая ячейка содержит в этом случае номер следующей хранимой строки. "-1"в обеих ячейках являются признаком конца перечня ненулевых элементов разреженной матрицы.

-1	Номер строки	Номер столбца	Значение	Номер столбца	Значение	...

...

-1	Номер строки	Номер столбца	Значение	...	-1	-1

Преобразование: Найти элемент матрицы, ближайший к заданному значению. Разделить на него элементы строки и столбца, на пересечении которых он расположен. Если таких элементов несколько, обработать все.

2 Основная часть

2.1 Разреженные матрицы

Разреженная матрица — это матрица с преимущественно нулевыми элементами (если же большая часть элементов ненулевые, матрицу называют **плотной**). Разреженные матрицы больших размеров часто возникают при решении таких задач, как дифференциальные уравнения в частных производных. При хранении и преобразовании разреженных матриц в компьютере бывает полезно (а часто и необходимо) использовать специальные алгоритмы и структуры данных, которые учитывают разреженную структуру матрицы. Операции и алгоритмы, применяемые для работы с обычными, плотными матрицами, применительно к большим разрежённым матрицам работают относительно медленно и требуют значительных объёмов памяти. Однако разрежённые матрицы могут быть легко сжаты путём записи только своих ненулевых элементов, что снижает требования к компьютерной памяти.

2.2 Описание алгоритма

Цель: написать функцию, выполняющую поставленную в введении задачу. Удобно будет разделить задание на 2 части: 1) поиск элемента (элементов) с ближайщим к заданному значением и 2) преобразование всех элементов строки и столбца, на пересечении которых он расположен.

1) Нахождение элемента. Здесь всё довольно просто — проходим по нашей матрице и рассматриваем каждый элемент. Заведём переменную dx, в которой будет храниться модуль разницы между заданным значением и последним найденным ближайшим к нему элементом. В начальный момент времени в переменной dx будем хранить значение DBL_MAX из библиотеки *<float.h>*. Стоит учитывать тот факт, что зачений, подходящих под условие, может быть два. Например, пусть дано значение n — тогда нам подходят все элементы со значением $n - dx$ и $n + dx$ при минимальном dx для этой матрицы (очевидно, что если в матрице есть элемент со значением n, то два этих значения совпдаают). Чтобы не проходить по всей матрице 2 раза, изначально заведём массивы индексов — в них будем хранить индексы всех элементов, подходящих под условие. Если находим новый элемент такой, что новое значение dx меньше предыдущего — очищаем массив индексов и добавляем в него индекс нового элемента. В моём примере я использую 4 вектора для индексов: 2 веткора под индексы i и j всех элементов со значением $n + dx$ — *iidf* и *jidf*, и 2 веткора под индексы i и j всех элементов со значением $n - dx$ — *iids* и *jids*. Во время написания этого отчёта до меня дошло, что проще было бы завести некую стрктуру $Point$ и хранить в ней два индекса — в таком случае понадобилось бы лишь 2 вектора структур $Point$ под элементы вида $n + dx$ и вида $n - dx$, что сделало бы структуру программы значительно проще. Но не судьба.

2) Преобразование матрицы. Здесь главная сложность состоит в том, что мы не имеем константного доступа к элементам строки/столбца, как в случае с вариантом хранения матрицы в виде двумерного массива. Поэтому проходим по всей матрице еще раз и рассматриваем каждый элемент. Если очередной элемент имеет хотя бы один индекс, который есть в одном из векторов *iidf* или *jidf* — делим его значение на $n + dx$. Если хотя бы один его индекс есть в веткоре *iids* или *jids* — делим его значение на $n - dx$.

2.3 Оценка сложности

n — количество строк матрицы, m — количество стобцов матрицы, k — количество ненулевых элементов матрицы, q — количество элементов матрицы, ближайщих к заданному числу

- **Чтение матрицы** — $O(n{\times}m)$, т.к. мы в любом случае считываем все элементы исходной матрицы.
- **Печать матрицы в стандартном виде** — $O(n{\times}m)$, т.к. мы в любом случае выводим все элементы матрицы, в том числе и нулевые
- **Печать матрицы во внутреннем представлении** — $O(k)$, т.к. мы печатаем только ненулевые элементы. Дополнительно выводится номер строк и для каждого элемента номер столбца, но это не влияет на сложность.
- **Нахождение элемента (см. 2.2)** — $O(k)$. Мы просматриваем каждый элемент матрицы и для ненулевых проводим ряд операций, который выполняются за линейное время.
- **Преобразование матрицы (см 2.2)** — $O(k{\times}q)$. Для каждого ненулевого элемента просматриваем, подходит ли он под условие (линейный поиск в массиве индексов). Если подходит — производим деление, не влияющее на сложность.

Сложность хранения матрицы по памяти — $O(k)$, т.к. храним только ненулевые элементы.

3 Вывод

Использованные приёмы позволяют использовать значительно меньше памяти для хранения огромных разреженных матриц, а также достаточно быстро получать доступ к их элементам. Естественно, эффективность использования таких матриц обратно пропорционально зависит от заполненности — то есть чем меньше ненулевых элементов, тем лучше.

4 Список источников

- Методические указания к выполнению курсовых работ. Зайцев В. Е.
- Записи с семинара, посвящённого КП7. Дубинин А. С.