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 «Программное обеспечение ЭВМ и информационные технологии»

Исследование быстродействия умножения матриц на языке Common Lisp

Студент Виноградов А. О.

Группа ИУ7-66Б

Преподаватель Строганов Ю.В.

Москва — 2023 г.

Введение

Матричная алгебра имеет обширные применения в различных отраслях знания – в математике, физике, информатике, экономике. Например, матрицы используется для решения систем алгебраических и дифференциальных уравнений, нахождения значений физических величин в квантовой теории, шифрования сообщений в Интернете.

Важной стороной работы с матрицами в программировании является оптимизация матричных операций (умножение, сложение, транспозиция и так далее), так как во многих задачах размеры матриц могут достигать больших значений. В данной работе пойдет речь об оптимизации операции умножения матриц на языке Common Lisp.

Целью данной работы является реализация и сравнительный анализ функций перемножения матриц с использованием хвостовой рекурсии и функционалов.

1 Аналитическая часть

Лисп (англ. Lisp) — семейство языков программирования, программы и данные в которых представляются в виде списков.

Функционал в языке Лисп — функция, которая принимает или возвращает другую функцию.

Рекурсия — ссылка на объект при описании самого объекта. Рекурсивная функция — функция, вызывающая саму себя. Отдельно выделяют хвостовую рекурсию — наиболее оптимизированный вариант рекурсии, при котором рекурсивный вызов — последнее действие, выполняемое программой в рамках функции.

Вывод

В данном разделе были рассмотрены основные термины, используемые в работе.

2 Технологическая часть

В данном разделе рассмотрены средства реализации, а также представлены листинги реализаций алгоритмов умножения матриц — рекурсивного и с использованием функционалов.

2.1 Разработка алгоритмов

В листингах 2.2 — 2.3 приведены реализации алгоритмов умножения матриц.

Листинг 2.1 – Функции реализующие умножение матриц с использованием хвостовой рекурсии

```
(defun getrow(n matrix)
    (nth n matrix)
)

(defun _getcol(n mat res)
  (cond
    ((null mat) res)
    (T (_getcol n (cdr mat) (cons (nth n (car mat)) res)))
  )
)

(defun getcol(n matrix)
  (reverse (_getcol n matrix ()))
)

(defun rowXcol(row col)
  ( apply #'+ 0 (mapcar #'* row col) )
)
```

Листинг 2.2 – Функции реализующие умножение матриц с использованием хвостовой рекурсии (продолжение)

```
(defun _rowXmatrix(row matrix matlen res counter)
  (cond
    ((= matlen counter) res)
    (T
      (_rowXmatrix
        row
        matrix
        matlen
        (cons (rowXcol row (getcol (- matlen counter 1) matrix))
           res)
        (+ 1 counter)
      )
    )
  )
)

(defun rowXmatrix(row matrix matlen)
  (_rowXmatrix row matrix matlen () 0))

(defun matrixmul(mat1 mat2 res n)
  (cond
    ((= n 0) res)
    (T
      (matrixmul
        mat1
        mat2
        (cons
          (rowXmatrix (getrow (- (length mat1) n) mat1) mat2
             (length mat2))
          res
        )
        (- n 1)
      )
    )
  ))

(defun matmul (mat1 mat2)
  (reverse
    (matrixmul mat1 mat2 () (length mat1))
  ))
```

Листинг 2.3 – Функция умножения матриц с использованием функционалов

```
(defun rowXcol(row col)
  ( apply #'+ 0 (mapcar #'* row col) )
)

(defun rowXmatrix(row matrix)
  (mapcar #'(lambda (mat_col)
        (rowXcol row mat_col)
      ) (transpose matrix)
  )
)

(defun matmul (mat1 mat2)
 (
    mapcar #'(lambda (row1)
        (rowXmatrix row1 mat2)
        ) mat1
  )
)

(defun transpose (matrix)
    (apply #'mapcar #'list matrix)
)
```

3 Исследовательская часть

Ниже приведены технические характеристики устройства, на котором было проведено измерение времени работы ПО:

- операционная система Ubuntu 22.04;
- оперативная память 8 Гб 2133 МГц;
- процессор Intel Core i5-8300H с тактовой частотой 2.30 ГГц.

3.1 Время выполнения реализаций алгоритмов

Для замеров времени использовалась функция замера процессорного времени get-intrnal-run-time языка Common Lisp.

Функция используется дважды — в начале и в конце замера времени, затем значения начальное значение вычитается из конечного.

Замеры проводились для квадратных матриц размерами 100, 200,...,600. Матрицы заполнялись случайными числами от 0 до 1000.

В таблице 3.1 представлены замеры времени работы для каждого из алгоритмов.

Таблица 3.1 – Результаты замеров времени работы реализаций алгоритмов для различных размерностей матриц (в мс)

Размерность	Рекурсия	Функционалы
100	67,7	16,2
200	1025,2	124,9
300	4905,7	470,4
400	14990,9	1243,5
500	36474,5	2428,4
600	76315,9	4572,5

На рисунке 3.1 приведена графическая интерпретация результатов измерения.

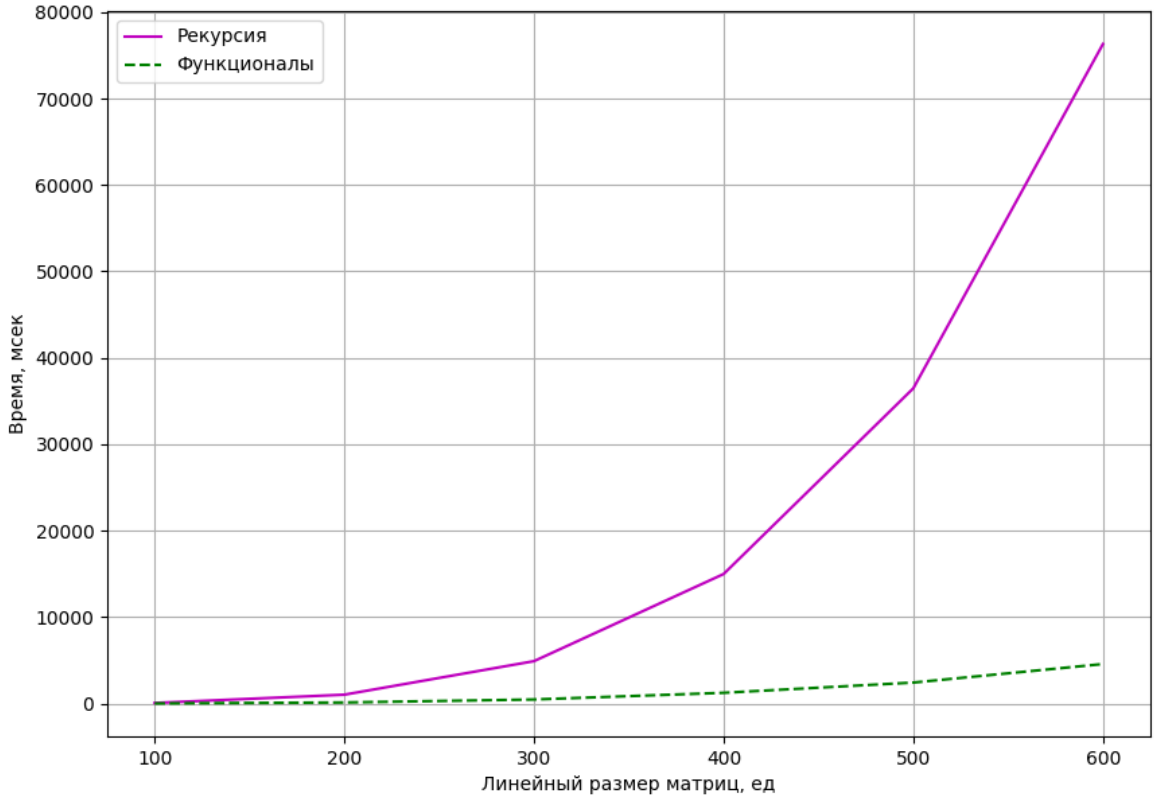

Рисунок 3.1 – Графическая интерпретация результатов измерения

Из полученных результатов можно сделать вывод, что реализация алгоритма с использованием функционалов работает быстрее рекурсивной реализации.

Вывод

В результате экспериментов было выявлено, что реализация алгоритма умножения матриц с использованием функционалов работает быстрее рекурсивной реализации. Для квадратных матриц размерностью 600×600 реализации дают следующие результаты:

– рекурсивный алгоритм — 76,3 с;

– алгоритм с использованием функционалов — 4,5 с;

Таким образом, использование функционалов является позволяет ускорить выполнение алгоритма умножения матриц. Для матриц размерностью 600×600 рекурсивная реализация проигрывает более чем в 16 раз.