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»

Лабораторная работа № 3
по дисциплине «Функциональные и логические языки программирования»

Тема Пролог. Динамическая БЗ.

Студент Одинцов Е.В.

Группа ИУ7-53БВ

Преподаватели Строганов Ю.В.

Москва, 2024

Содержание

ВВЕДЕНИЕ

Целью данной работы является создание программы на Prolog для нахождения значения элемента последовательности Фибоначчи и факториала. Указанные последовательности должны быть написаны двумя способами:

— без использования динамической базы знаний (мемоизации);

— с использованием динамической базы знаний (с мемоизацией).

1 Аналитическая часть

Числа Фибоначчи определяются рекурсивной формулой:

$$F(0) = 0, \quad F(1) = 1, \quad F(n) = F(n-1) + F(n-2), \quad n > 1$$

Программа для вычисления чисел Фибоначчи на Prolog реализована двумя способами: без мемоизации и с использованием мемоизации.

Рекурсивное вычисление Фибоначчи

В рекурсивной версии каждый вызов функции для вычисления $F(n)$ вызывает два дополнительных рекурсивных вызова для $F(n-1)$ и $F(n-2)$, что приводит к экспоненциальной сложности $O(2^n)$.

Листинг 1.1 — Рекурсивное вычисление Фибоначчи

```
fib(0, 0).
fib(1, 1).
fib(N, F) :-
    N > 1,
    N1 is N - 1,
    N2 is N - 2,
    fib(N1, F1),
    fib(N2, F2),
    F is F1 + F2.
```

Фибоначчи с мемоизацией

Для оптимизации программы используется мемоизация — техника запоминания ранее вычисленных значений. В Prolog для этого используется динамическая база знаний с помощью предиката `assertz/1`, который добавляет факт в базу знаний.

Листинг 1.2 — Фибоначчи с мемоизацией

```
:- dynamic fib_mem/2.

fib_mem(0, 0).
fib_mem(1, 1).
fib_mem(N, F) :-
    N > 1,
    N1 is N - 1,
    N2 is N - 2,
    fib_mem(N1, F1),
    fib_mem(N2, F2),
```

```
    F is F1 + F2,
    assertz(fib_mem(N, F)).
```

Использование мемоизации снижает сложность до линейной — $O(n)$, поскольку каждое значение $F(n)$ вычисляется только один раз.

Факториал

Факториал числа n определяется формулой:

$$n! = \begin{cases} 1, & \text{если } n = 0 \\ n \cdot (n-1)!, & \text{если } n > 0 \end{cases}$$

Рекурсивное вычисление факториала

Программа для вычисления факториала также реализована с использованием рекурсии.

Листинг 1.3 — Рекурсивное вычисление факториала

```
fact(0, 1).
fact(N, F) :-
    N > 0,
    N1 is N - 1,
    fact(N1, F1),
    F is F1 * N.
```

Факториал с мемоизацией

Аналогично Фибоначчи, факториал можно оптимизировать с помощью мемоизации:

Листинг 1.4 — Факториал с мемоизацией

```
:- dynamic fact_mem/2.

fact_mem(0, 1).
fact_mem(N, F) :-
    N > 0,
    N1 is N - 1,
    fact_mem(N1, F1),
    F is N * F1,
    assertz(fact_mem(N, F)).
```

2 Технологическая часть

2.1 Используемые технологии

1) Язык: SWI-Prolog (реализация языка программирования Prolog)
2) IDE:
    — TexStudio,
    — Visual Studio Code.

ЗАКЛЮЧЕНИЕ

В ходе работы были реализованы алгоритмы для вычисления чисел Фибоначчи и факториалов двумя способами: рекурсивным и с мемоизацией. Применение мемоизации существенно оптимизировало производительность программы за счет устранения избыточных вычислений.

Данная работа демонстрирует эффективность использования мемоизации для оптимизации рекурсивных алгоритмов, что особенно полезно при работе с большими значениями аргументов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SWI-Prolog. Доступно на: https://www.swi-prolog.org/. [Дата обращения: сентябрь 2024].

2. Статья о DC-grammar. Доступно на: https://en.wikipedia.org/wiki/Definite_clause_grammar. [Дата обращения: сентябрь 2024].