ВЫБОР ПРАВИЛЬНОГО АЛГОРИТМА ДЛЯ РЕШЕНИЯ ЗАДАЧ

Марчук К.А.[1)], Е.М. Гецман[2)]

1) студент Белорусского национального технического университета, г. Минск, Беларусь, kirillka.mar002@gmail.com

2) старший преподаватель Белорусского национального технического университета, г. Минск, Беларусь, hetsman@bntu.by

Аннотация: необходимость выбора правильного алгоритма при решении любых задач на примере задачи повышенного уровня, сравнение алгоритмов и выявление их недостатков, а также их быстродействие при различных данных.

Ключевые слова: алгоритм, числа Фибоначчи, производительность, период Пизано.

SELECTING THE RIGHT ALGORITHM FOR SOLVING PROBLEMS

K.A. Marchuk[1)], E.M. Hetsman[2)]

1) student of the Belarusian National Technical University, Minsk, Belarus, kirillka.mar002@gmail.com

2) Senior Lecturer, Belarusian National Technical University, Minsk, Belarus, hetsman@bntu.by

Abstract: the need to choose the correct algorithm for solving any problems on the example of a higher-level problem, comparing algorithms and identifying their shortcomings, as well as their performance for different data.

Key words: algorithm, Fibonacci numbers, performance, Pisano period.

Введение

В современном мире программы являются неотъемлемой частью различных устройств и систем, и многие программисты часто сталкиваются с ситуациями, когда вычислительной мощности оборудования недостаточно для успешной работы. Зачастую, ошибкой является неудачный алгоритм [1]. Алгоритм – это определенная последовательность действий для решения задачи.

Разберем важность выбора алгоритма для решения любых задач и в качестве примера возьмем задачу по программированию повышенной сложности: огромное число Фибоначчи по модулю.

Основная часть

Числа Фибоначчи – это целые натуральные числа, расположенные в числовой последовательности таким образом, что каждое последующее число является суммой двух предыдущих чисел, при этом в этом числовом ряде проявляются уникальные свойства, выраженные в постоянных отношениях между отдельными членами последовательности (рис. 1) [2].

Рисунок 1 – Алгоритм для нахождения числа Фибоначчи

Данная задача имеет множество решений, однако не все алгоритмы удовлетворяют своей производительностью. Одним из примитивных алгоритмов является нахождение n–ое числа Фибоначчи и определения остатка от деления на m.

```
import time as t
def fibonacci(n):
    if n in (1, 2):#если число равно 1 или 2 возвращаем 1
        return 1
    return fibonacci(n - 1) + fibonacci(n - 2)# создаем рекурсию


n,m = map(int,input("Введите числа n и m ").split()) # Ввод данных
start = t.time()# начало выполнения программы
print("Искомый остаток от деления = ",fibonacci(n) % m)
end = t.time()
print("Время выполнения программы  = ",  end - start)
```

Рисунок 2 – Код первого алгоритма

Алгоритм неплох для нахождения числа Фибоначчи по m, лишь при малых значениях n. Исходя из таблицы 1, можно сделать вывод, что первый алгоритм неприемлем начиная с 100000, так как время выполнения кода уже более 1 секунды и в последующем время на выполнение будет расти.

Таблица 1-Производительность программы от n

Время, с	Число Фибоначчи
0.015599966049194336	10
0.015599966049194337	100
0.015599966049194338	1000
0.062400102615356445	10000
1.762803077697754	100000
41.91727352142334	500000

Низкая скорость алгоритма объясняется экспоненциальной скоростью роста чисел Фибоначчи (рис. 3), в следствии чего алгоритм порождает огромное дерево рекурсивных вызовов (рис. 4). Дерево рекурсивных вызовов – показывает из скольких вызовов были произведены другие вызовы. Для того чтобы посчитать F_n – ое число Фибоначчи наш алгоритм делает 2 рекурсивных вызова F_{n-1} и F_{n-2} , а те в свою очередь делают по 2 рекурсивных вызова и так далее.

Экспоненциальная скорость роста

$$F_{20} = 6765$$
$$F_{50} = 12586269025$$
$$F_{100} = 354224848179261915075$$
$$F_{1000} = \begin{array}{l}434665576869374564356885276 7\\ 50406258025646605173717804 02\\ 48172908953655541794905189 04\\ 03879840079255169295922593 08\\ 03226347752096896232398733 22\\ 47116164299644090653318793 82\\ 98969649928516003704476137 79\end{array}$$

Рисунок 3 – Числа Фибоначчи

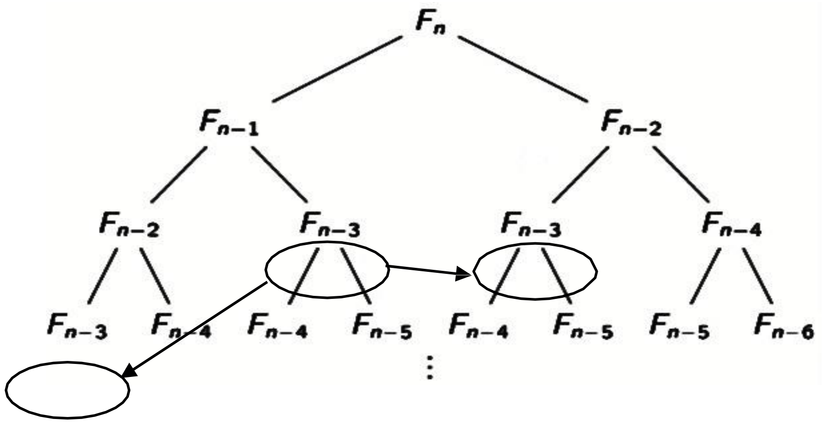

Рисунок 4 - Рекурсивное дерево

Из (рис.4) следует, что алгоритм вычисляет одно и тоже много раз, например F_{n-3} число Фибоначчи он вычисляет 3 раза, а F_{n-4} число Фибоначчи он вычисляет 4 раза. Последовательность Фибоначчи периодична по модулю любого целого положительного числа m, так как среди первых m^2+1 пар чисел найдутся две равные пары $(x_i, x_{i+1}) = (x_j, x_{j+1})$ для некоторых $i \le j$. Поэтому для всех чисел выполняется $x_{i+k} = x_{j+k}$, то есть, последовательность периодична. Выявленный недостаток предлагаемого алгоритма заключается в повторении вычисления чисел Фибоначчи, что в дальнейшем приводит к вынужденному решению в смене исчислений путем применения наиболее удобного и привлекательного алгоритма (рис. 6), основанного на вычислении периода Пизано (рис. 5)[5].

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F_i	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610
F_i mod 2	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0
F_i mod 3	0	1	1	2	0	2	2	1	0	1	1	2	0	2	2	1

Рисунок 5 – Период Пизано для m = 2 и m = 3

При последовательности вычисления нахождения остатка от деления *n*-го числа Фибоначчи на натуральное число *m* достаточно найти все числа периода Пизано для данного m. Затем определить остаток от деления *n* на длину периода, и выбрать число из периода Пизано под этим номером. Следовательно, не нужно считать *n*-ое число Фибоначчи и искать остаток от деления на *m*, а следует первоначально работать с остатками от деления, что является быстрее.

```
import time as t
def fibanachi2(n, m):
    o, i = [0,1], 2
    while not (o[i-2] == 0 and o[i-1] == 1) or i <= 2: #условие выхода из цикла
            o.append((o[i-2] + o[i-1]) % m)# добовляем остатки от деления в массив o
            i+=1
    return o[n % (i - 2)] #определяем остаток соответствующий данному n

n, m = map(int, input("Введите значения n, m = ").split())
start = t.time()# начало выполнения программы
print("Искомый остаток от деления  = ", fibanachi2(n, m))
end = t.time()# конец работы
print("Время выполнения кода  = ", end - start)
```

Рисунок 6 – Код первого алгоритма

Алгоритм, использующий вычисления периода Пизано, подходит для нахождения числа Фибоначчи по m, при всех значениях n. Представив производительность программы по времени (см. таблицу 2), следует вывод, что второй алгоритм подходит лучше для решения данной задачи, так как его скорость исполнения не зависит от числа Фибоначчи.

Таблица 2-Производительность программы от n

Время, с	Число Фибоначчи
0.015600204467773438	10
0.015599966049194336	100
0.015599966049194336	1000
0.015599966049194336	10000
0.015599966049194336	100000
0.015600204467773438	500000

По результатам работы выявлены недостатки описанных алгоритмов для решения задачи на нахождение огромного число Фибоначчи по модулю. Для первого характерны следующие изъяны, заключающиеся в порождении огромного рекурсивного дерева и низкой скорости работы.

Во втором алгоритме устранены недостатки первого, в следствие чего производительность и точность возросла. Как показывает практика, почти всегда можно определить способ улучшить уже разработанный алгоритм программы и тщательно подобрать в начале проектирования, чтобы избежать в дальнейшем неприятных последствий, связанных с доработкой фрагментов программного кода в течение короткого промежутка времени.

Заключение

В завершении подведем итог оптимизации рассмотренных алгоритмов и отметим, что секрет успеха зависит от выбора правильного алгоритма и его эффективности. При выборе алгоритма важно найти узкие места задачи, наиболее влияющие на производительность работы нашей программы, что поможет

существенно выиграть время, а также избежать ненужных ошибок. Идеальных решений не бывает, и разработка алгоритма всегда сопровождается ошибками и недоработками.

Список использованных источников:

1. Горовенко Л.А. Логическое программирование как средство решения задач искусственного интеллекта // Современные проблемы математики и информатики: Сборник научных трудов. Вып 1/ Сост. Н.Г.Дендеберя, С.Г.Манвелов.- Армавир: редакционно-издательский центр АГПУ, 2004. – С. 56-57.
2. Числа Фибоначчи // [электронный ресурс] – Режим доступа: https://hi-news.ru/science/chislo-fibonachchi-pochemu-ono-tak-populyarno-v-prirode.html. Дата доступа: 11.02.2021.
3. Сысоева М.В. Программирование для нормальных с нуля на языке Python / Сысоев И.В. -1-е изд.- Москва: Базальт СПО; МАКС Пресс, 2018. - 180 с.
4. Python для сложных задач: наука о данных и машинное обучение. – СПБ.: Питер, 2018 – 576 с.
5. Период Пизано [Электронный ресурс]/ солнечная энергетика. - Режим доступа: https://ru.qaz.wiki/wiki/Pisano_period. – Дата доступа: 25.03.2021.