Низкий уровень применения электронного обучения связан с консервативной позицией академического сообщества и (или) ограниченной готовностью студентов к некоторым современным образовательным технологиям. Образовательная организация вынуждена не только повышать заинтересованность и квалификацию педагогических работников области информационных технологий обучения, но и мотивировать и включать в процессы электронного обучения студентов. Хорошим механизмом в достижении поставленных результатов развития электронного обучения станет качественное локальное регулирование, позволяющее реализовывать методы смешанного обучения (сочетание традиционного и электронного обучения), развивать и совершенствовать методы электронного обучения, предусматривать оценку эффективности применяемых методов обучения.

Список литературы

1. Основные направления деятельности Правительства Российской Федерации на период до 2024 года : утверждены Правительством РФ 29.09.2018 г.

УДК [511.17:512.643:519.217.2]:004.438

Ощепков Е. Д.

ПРОБЛЕМА КОЛЛАТЦА И ВЫЧИСЛЕНИЯ В JULIA

Евгений Дмитриевич Ощепков

магистрант

oshchepkov.04.05.1997@gmail.com

Кубанский государственный университет (КубГУ), Россия, Краснодар

THE PROBLEM OF COLLATZ AND COMPUTING IN JULIA

Evgeny Dmitrievich Oshchepkov

FGBOU VO «Kuban state University»

Аннотация*. В данной статье обсуждается так называется проблема Коллатца, а также сравниваются вычисления на языках Python и Julia.*

Abstract. *This article discusses the so-called Collatz problem and compares calculations in Python and Julia.*

Ключевые слова: *Проблема Коллатца, сиракузская проблема, теория чисел, Python, Julia.*

Keywords: *Problem of Collatz, Syracuse problem, theory of number, Python, Julia.*

Введение

Для начала стоит обсудить что такое проблема Коллатца. Мы берем произвольное нечетное число A и заменяем его на 3*A+1 — это уже четное число. После этого делим его на степени двойки пока не получится нечетное число B. Отображение A → B называется отображением Коллатца. Требуется доказать, что если взять достаточное количество суперпозиций этого отображения, то каким бы не было исходное нечетное число A в итоге всегда получится 1. В работе проведен детальный анализ этой проблемы как средствами прямых вычислений, так и анализом цепочек, как цепей Маркова. Вычисления матриц Маркова производилось средствами языков Python и Julia.

Цепью Маркова называют такую последовательность случайных событий, в которой вероятность каждого события зависит только от состояния, в котором процесс находится в текущий момент и не зависит от более ранних состояний.

Матрица предельных переходов — Матрица, в которой все строки одинаковы. Стационарные состояния — одна строка из матрицы предельных переходов.

Инструментарий

Язык программирования Julia — высокоуровневый высокопроизводительный язык программирования с динамической типизацией, созданный для математических вычислений.

Прежде чем делать программу, которая получает матрица переходов марковского процесса для нужного нам модуля, нужно показать как получается минимальная из них на ручных вычислениях.

Лемма:

Пусть у нас есть два состояния системы — это числа вида $6k+1$ и $6k+5$, $k \in Z$.

Тогда матрица переходов марковского процесса имеет вид:

$$\frac{1}{3}\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$$

Доказательство:

Мы имеем два состояния системы. Разберём два случая:

Случай 1:

Рассмотрим число $6k+1$.

В выражении $3a+1$ заменяем a на наше число: $3(6k+1)+1=18k+4$. Делим на 2, получаем $9k+2$. k может быть либо чётным, либо нечётным, т.е. обе вероятности равны 1/2.

Рассмотрим оба случая.

$$k = 2m \Rightarrow 18m+2 \Rightarrow 9m+1$$
$$k = 2m+1 \Rightarrow 18m+11 = 6(3m+1)+5$$

Во втором случае мы получили число второго вида с вероятностью 1/2.

Далее берём выражение $9m+1$ и рассматриваем дальше.

$$m = 2n \Rightarrow 18n+1 = 6(3k)+1$$
$$m = 2n+1 \Rightarrow 18n+10 \Rightarrow 9n+5$$

Получили число первого вида с вероятностью 1/4.

$$k = 2t \Rightarrow 18t+5 = 6(3t)+5$$
$$k = 2t+1 \Rightarrow 18t+14 \Rightarrow 9t+7$$

Получили число второго вида с вероятностью 1/8.

$$t = 2p \Rightarrow 18p+1 \Rightarrow 6(3p)+1$$
$$t = 2p+1 \Rightarrow 18p+10 \Rightarrow 9k+5$$

Получили число первого вида с вероятностью 1/16.

У нас получилось выражение $9k+5$, как ранее мы уже выяснили, оно переходит в число второго вида с вероятностью 1/32.

Далее алгоритм зацикливается.

Таким образом у нас получаются две бесконечные суммы чисел.

Для первого числа:

$$\frac{1}{4}+\frac{1}{16}+\frac{1}{64}+\ldots=\frac{1}{4}\left(1+\frac{1}{4}+\frac{1}{16}+\ldots\right)$$

Для второго числа:

$$\frac{1}{2}+\frac{1}{8}+\frac{1}{32}+\ldots=\frac{1}{2}\left(1+\frac{1}{4}+\frac{1}{16}+\ldots\right)$$

Нетрудно заметить, что перед нами бесконечно убывающая геометрическая прогрессия.

Вычисляем суммы по формуле $S=\frac{b_1}{1-q}$, и получаем 1/3 и 2/3 соответственно. Эти два числа и задают первую строку матрицы.

Случай 2:

Точно так же рассмотрим второе число $6k+5$.

$$3(6k+5)+1=18k+16\Rightarrow 9k+8$$
$$k=2m\Rightarrow 18m+8\Rightarrow 9m+4$$
$$k=2m+1\Rightarrow 18m+17\Rightarrow 6(3m+2)+5$$

С вероятностью 1/2 получили число второго вида.

$$m=2n\Rightarrow 18n+4\Rightarrow 9n+2$$
$$m=2n+1\Rightarrow 18n+13=6(3n+2)+1$$

С вероятностью 1/4 получили число первого вида

$$n=2t\Rightarrow 18t+2\Rightarrow 9t+1$$
$$n=2t+1\Rightarrow 18t+11=6(3t+1)+5$$

С вероятностью 1/8 получили число второго вида

$$t=2p\Rightarrow 18p+1=6(3p)+1$$
$$t=2p+1\Rightarrow 18p+10\Rightarrow 9p+5$$

С вероятностью 1/16 получили число первого вида.

В данном случае алгоритм зацикливается на два шага позже. Рассмотрим до конца, чтобы убедиться.

$$p = 2l \Rightarrow 18l + 5 = 6(3l) + 5$$
$$p = 2l + 1 \Rightarrow 18l + 14 \Rightarrow 9l + 7$$

С вероятностью 1/32 получили число второго вида.

$$l = 2r \Rightarrow 18r + 7 = 6(3r + 1) + 1$$
$$l = 2r + 1 \Rightarrow 18r + 16 \Rightarrow 9r + 8$$

С вероятностью 1/64 получили число первого вида. И заметили, что наше исходное число было $9k + 8$, т.е. алгоритм зациклился.

Вероятности перехода в числа первого и второго вида точно такие же, как и в первом случае. Поэтому получаем вторую строку матрицы равную первой.

$$\frac{1}{3}\begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$$

Что и требовалось доказать.

Программа

Программа была написана в Jupiter notebook, с помощью пакета IJulia. Делится данная программа на несколько пунктов:

1. Входные данные, в соответствии с рисунком 1.

```
N = 2
D = 1
Step = Array{Int128,2}(undef,(N*(3^D)),3)
Mat1 = [1,5]
zero = Int128
zero = 0
t = [zero,zero]
Fr = Int128[]
Lt = Array{BigInt,2}(undef,(N*(3^(D-1)),(N*(3^(D-1)))))
for i in 1:D
  append!(Fr, [2*3^(i-1)])
end
NN = Fr[end]
```

Рисунок 1 — «Входные данные»

Здесь мы видим константы N и D. Это составные модуля, по которому надо посчитать стационарные состояния. А именно $N*3^D$. Матрица Step — матрица, в которой будут записаны степени для 2, чтобы получить матрицу переходов. Вектор Fr — матрица-вектор, в которую записываются все значения модулей, через которые нужно пройти до итогового модуля. Вектор t —

вспомогательный вектор. Матрица Lt — матрица переходов Марковского процесса.

2. Полезные функции, в соответствии с рисунком 2.

```
function prin(x,y,z)
   for i = 1:y
      for j = 1:z
         print(x[i,j], " ")
      end
      println()
   end
end
function Amin(A,x,y)
   m = A[1,1]
   for i = 1:x
      for j = 1:y
         m = min(A[i,j],m)
      end
   end
   return m
end
function NOD(x,y)
    while (x!=0) & (y!=0)
        if x > y
            x = x % y
        else
            y = y % x
        end
    end
    return(x+y)
end
```

Рисунок 2 — «Полезные функции»

Здесь мы видим 3 функции:

- Функция prin(x,y,z) – печатает y строк и z столбцов у матрицы x;
- Функция Amin(A,x,y) – возвращает значение минимального элемента матрицы A с x строками и y столбцами;
- Функция NOD(x,y) – ищет НОД(x,y).

3. Поиск элементов матрицы переходов Марковского процесса, в соответствии с рисунком 3.

```
for h in 1:D
   M = N*3
   for s in 1:N
     if (((N in Fr) & (s == 1)))
       for i in 1:(N-(div(N,3)))
         append!(t, [zero])
       end
     end
     for i in 1:N
       t[i] = zero
     end
     P = Mat1[s]
     St = 1
     P = Int128((3*P + 1)/2)
     G = Int128(3*M/2)
     for k in 1:N*(3^(h))
     if (h == D)

       for i in 1:N
         K = BigInt(N - Step[i,1])
         Lt[s,i] = 2^K
           end
       end
   end
   N *= 3
end
```

Рисунок 3 — «Общий вид алгоритма»

```
for k in 1:N*(3^(h))
  if (P%2 == 0)
    if ( (P+G in Mat1) )
          else
              append!(Mat1, [P+G])
          end
          Mat1 = sort(Mat1)
    K = Int128((P+G))%M
    P = Int128(P/2)
    for i in 1:N
      if (Int128(K) == Mat1[i])
        if(t[i] <= 2)
          t[i] += 1
          Step[i,t[i]] = St
          St += 1
                  end
              end
          end
  else
    if ((P in Mat1) )
          else
              append!(Mat1, [P])
          end
          Mat1 = sort(Mat1)
    K = Int128((P)%M)
    P = Int128((P+G)/2)
    for i in 1:N
      if (K == Mat1[i])
        if(t[i] <= 2)
          t[i] += 1
          Step[i,t[i]] = St
          St += 1
                  end
              end
          end
      end
  end
```

Рисунок 4 — «Пропущенный цикл»

Здесь мы видим алгоритм из доказательства Леммы, но в общем виде, с пропуском одного цикла, который находится в отмеченной строке на рисунке 3. Данный цикл указан полностью на рисунке 4.

Можно заметить, что в алгоритме доказательства есть замена, которую можно привести к общему виду:

- Для нечётных чисел:

$$k = 2n \Rightarrow 2nG + P = M(3n + (PdivM)) + P \bmod M$$
$$k = 2n+1 \Rightarrow G(2n+1) + P = 2nG + G + P \Rightarrow nG + \frac{P+G}{2}$$

- Для чётных чисел:

$$k = 2n \Rightarrow 2nG + P = nG + \frac{P}{2}$$
$$k = 2n+1 \Rightarrow G(2n+1) + P = 2nG + G + P =$$
$$= M(3n + ((G+P)divM)) + (G+P) \bmod M$$

Именно эти P,G и M мы видим в программе.

4. Вывод неповторяющихся строк матрицы переходов Марковского процесса, а также коэффициент-знаменатель, в соответствии с рисунком 4.

```
Sq = BigInt(0)
for i in 1:2*3^(D-1)
  Sq += BigInt(Lt[1,i])
end
println("1/",Sq)
if NN > 2
    prin(Lt,div(NN,3),NN)
else
    for i = 1:NN
        print(Lt[1,i], " ")
    end
end
N = div(N,3)
```

Рисунок 5 — «Вывод матрицы»

На самом деле в матрице переходов Марковского процесса всегда только треть неповторяющихся строк, за исключением модуля n=6, которые и имеет смысл выводить.

5. Вычисление и вывод матрицы стационарных состояний, в соответствии с рисунком 5.

```
Nod_in_Matrix_1 = Nod_in_Matrix_2 = Lt
Nod1 = Int128
Nod1 = 1
for k = 1:D-1
    Nod_in_Matrix_2 *= Nod_in_Matrix_1
    Nod1 = NOD(Nod_in_Matrix_2[1,1], Nod_in_Matrix_2[1,2])
    for i = 1:NN
        for j = 1:NN
            Nod1 = NOD(Nod1,Nod_in_Matrix_2[i,j])
        end
    end
    for i = 1:NN
        for j = 1:NN
            Nod_in_Matrix_2[i,j] = div(Nod_in_Matrix_2[i,j],Nod1)
        end
    end
end
println("  Цепь Маркова:")
println()
println(" 1/",sum(Nod_in_Matrix_2[1,:]))
prin(Nod_in_Matrix_2,1,NN)
println()
println("  ВЕРОЯТНОСТЬ ПОЯВЛЕНИЯ ЧИСЕЛ: ")
for i = 1:NN
        println("вида ",NN*3,"k+",Mat1[i]," = ",
        Nod_in_Matrix_2[1,i],"/",sum(Nod_in_Matrix_2[1,:]),
        " = ",Float32(Nod_in_Matrix_2[1,i]/sum(Nod_in_Matrix_2[1,:])),")
end
```

Рисунок 6 — «Стационарные состояния»

Было замечено, что для того, чтобы получить стационарные состояния, не обязательно проверять каждую степень исходной матрицы. Можно сразу возводить в степень D. Также матрица стационарных состояний после каждой новой степени сокращалась на НОД всей матрицы.

Результаты

Кроме модуля n=6, мной вручную было просчитаны матрицы переходов Марковского процесса для модулей 18 и 54, которые не поместятся в данную статью. Также было просчитаны данные матрицы на Python. Вычисления на Python ограничиваются лишь модулем n=54, так как пакет NumPy не поддерживает настолько больших чисел в матрицах, максимально возможный тип, который был найден мной - это натуральные числа более 32 бит.

Вычисления в Julia дали намного больший результат, так как были посчитаны вероятности появления чисел по модулям n=162, n=486 и n=1458.

Вот сами вычисления:

Для модуля *n = 6*:

1/3 (1 2)

Для модуля *n = 18*:

- вычисления на Python:

1/3969 (504 252 126 1008 693 1386)

- вычисления на Julia, с применением деления на НОД матрицы:

1/63 (8 4 2 16 11 22)

Для модуля $n = 54$:

- вычисления на Python:

1/18014192351838207 (661900949988768 593181997539768 16547523749712 878503088108016 1600120807774965 2370254107870908 439251544054008 219625772027004 109812886013502 1323801899977536 360087310832760 720174621665520 1186363995079536 330950474994384 296590998769884 2372727990159072 1185127053935454 3200241615549930)

- Вычисления на Julia, с применением деления на НОД матрицы:

1/262143 (9632 8632 2408 12784 23285 34492 6392 3196 1598 19264 5240 10480 17264 4816 4316 34528 17246 46570)

Заключение

Программа, просчитывающая матрицы Маркова, позволила мне выдвинуть гипотезу, почему так непредвиденно ведёт себя плотность распределения цепочек Коллатца. Также можно заметить, что вычисления в Julia намного лучше, чем в Python, так как на Python уже при модуле 162 не получаются стационарные состояния. А на Julia пока не найден предел. Также вычисления в Julia производятся во много раз быстрее, но точных расчётов не проводилось.

Список литературы

1. Рожков, А. В. Преподавание математики и информатики в ведущих университетах мира и опыт КубГУ / А. В. Рожков, М. В. Рожкова // Университеты в системе поиска и поддержки математически одаренных детей и молодежи : материалы I Всероссийской научно-практической конференции. – Майкоп, 2015. – С. 116–121.

2. Рожков, А. В. Стратегия DPS – Debian-Python-Sage: Проблемно-ориентированные вычислительные среды на открытом коде / А. В. Рожков //

Труды V Международной научно-практической конференции «Информационные технологии в образовании и науке» (ИТОН – 2016). – Казань : КФУ, 2016. – С. 172–179.

3. Рожков, А.В. Экспериментальная (вычислительная) теория чисел / А. В. Рожков, М. В. Рожкова // Новые информационные технологии в образовании и науке : материалы X международной научно-практической конференции, Екатеринбург, 27 февраля – 3 марта 2017 г. // Рос. гос. проф.-пед. ун-т. Екатеринбург, 2017. – С. 413–417.

4. Ощепков, Е. Д. Цепи Маркова и проблема Коллатца / Е. Д. Ощепков // Труды VIII Международной научно-практической конференции «Информационные технологии в образовании и науке» (ИТОН – 2018). – Казань : КФУ, 2018. – С. 197–199.

5. Стахурская, П. А. Вычислительные аспекты экспериментальной теории чисел / П. А. Стахурская // Труды VIII Международной научно-практической конференции «Информационные технологии в образовании и науке» (ИТОН – 2018). – Казань : КФУ, 2018. С. 274–278.