Реферат по статье “Система обслуживания с ветвящимися потоками вторичных требований.”

Предмет: Теория случайных процессов

Александр Сергеевич Баклашов

Содержание

List of Figures

1 Введение

При проектировании различных систем — от автоматических систем управления (АСУ) до вычислительных систем, значительную роль играют приоритетные модели теории массового обслуживания. Среди таких систем можно выделить системы с различными динамическими ориентациями и режимами переключения, которые выделяются в контексте исследования моделей, включающих в себя несколько типов требований. В таких моделях разнообразие типов требований подразумевает необходимость применения соответствующих системных ориентаций и режимов переключения при их изменении.

Помимо этого, в рассматриваемых системах стоит задача оптимизации приоритетного обслуживания, заключающаяся в разработке оптимальных стратегий обслуживания, которые могли бы обеспечить эффективное управление различными типами требований в системе.

В данной работе рассматривается задача определения оптимальной дисциплины обслуживания в системе с несколькими типами требований, имеющих различные приоритеты и поступающих как извне (первичных), так и в результате обслуживания (вторичных) и ветвящимися потоком вторичных требований. Модели такого типа становятся актуальными при исследовании работы ЭВМ в различных режимах, исследовании информационно-поисковых и других различных систем.

Приведем конкретные примеры, где такие системы становятся значимыми. Предположим, мы рассматриваем работу компьютера в пакетном режиме. В этом случае после начальной обработки пакета данных определяется количество программ в нем. Каждая программа может запросить разнообразные ресурсы, такие как вызов стандартных программ, доступ к оперативной памяти, или обращение к внешним устройствам для получения дополнительной информации.

Еще один пример связан с задачей поиска информации в массивах данных. После анализа некоторого массива можно обнаружить, что необходимая инфор-

мация на самом деле содержится в одном из других массивов.

Эти примеры демонстрируют, насколько важным является эффективное управление разнообразными запросами и ресурсами в системах с ветвящимися потоками требований, исследование и оптимизация которых имеют непосредственное прикладное значение. Организация работы таких систем также включает оперативное определение приоритетов обслуживания требований. Данная работа демонстрирует, что относительно линейного функционала потерь оптимальной является именно приоритетная дисциплина, для которой приводится алгоритм её построения.

2 Постановка задачи. Предположения

В работе “Система обслуживания с ветвящимися потоками вторичных требований” рассматривается однолинейная система обслуживания (система обслуживания (СО), в которой все поступающие заявки хранятся в одной очереди), выполняющая r типов операций. Длительности выполнения отдельных операций являются независимыми случайными величинами (СВ) с функциями распределения (ФР) $\beta_i(\cdot)$, при этом $\beta_i(0) = 0$, имеющими первые два момента b_i, b_{i2}. Первичные требования на выполнение операции типа i образуют пуассоновский поток с интенсивностью λ_i, $\lambda_i \geq 0$, $i = 1, r$. Помимо этого, имеют место случаи, в которых на некоторые, но не на все операции первичных требований не поступает. В результате выполнения операции типа i вызвавшее ее требование считается обслуженным, но с вероятностью $q_i(n) = q_i(n_1, ..., n_r)$ возникает набор $n = (n_i, ..., n_r)$ вторичных требований на выполнение операций различных типов, которые мгновенно поступают в очереди (неограниченные) для требований соответствующего типа.

Непосредственно вслед за этим по набору $l = (l_1, ..., l_r)$, где l_i — число требований в i- й очереди, с помощью функции управления $u(l)$ выбирается очередное требование на обсуживание. При этом, если $u(l) = i$, то будет обслуживаться требование типа i.

Предполагается, что простои прибора при наличии требований не допускаются, т.е. любое требование, заставшее прибор свободным, немедленно начинает обслуживаться и, если $u(l) = i$, то $l_i > 0$ (начинает обслуживаться требование типа i). В момент, когда система освобождается и нельзя направить требование на обслуживание, мы доопределяем $u(0) = 0$.

Для описания поведения системы введём процесс $L(t) = (L_i(t), ..., L_i(t))$, где $L_i(t)$ — число требований типа i в момент t.

Далее зададим c_i — стоимость единицы времени пребывания в системе требования типа r. Задача заключается в определении функции управления, минимизирующей потери в единицу времени в стационарном режиме работы системы.

При сделанных далее предположениях функционал, определяющий эти потери, записывается в виде

$$J = \sum_{i=1}^{r} c_i L_i$$

где L_i — среднее число требований типа i в системе в стационарном режиме.

3 Теоретическое введение

3.1 Алгоритмы Евклида

Алгоритм Евклида — эффективный алгоритм для нахождения наибольшего общего делителя двух целых чисел (или общей меры двух отрезков). Алгоритм назван в честь греческого математика Евклида (III век до н. э.), который впервые описал его в VII и X книгах «Начал». Это один из старейших численных алгоритмов, используемых в наше время.

В самом простом случае алгоритм Евклида применяется к паре положительных целых чисел и формирует новую пару, которая состоит из меньшего числа и разницы между большим и меньшим числом. Процесс повторяется, пока числа не станут равными. Найденное число и есть наибольший общий делитель исходной пары. Евклид предложил алгоритм только для натуральных чисел и геометрических величин (длин, площадей, объёмов). Однако в XIX веке он был обобщён на другие типы математических объектов, включая целые числа Гаусса и полиномы от одной переменной. Это привело к появлению в современной общей алгебре такого понятия, как евклидово кольцо. Позже алгоритм Евклида был обобщён на другие математические структуры, такие как узлы и многомерные полиномы.

Для данного алгоритма существует множество теоретических и практических применений. В частности, он является основой для криптографического алгоритма с открытым ключом RSA, широко распространённого в электронной коммерции. Также алгоритм используется при решении линейных диофантовых уравнений, при построении непрерывных дробей, в методе Штурма. Алгоритм Евклида является основным инструментом для доказательства теорем в современной теории чисел, например таких как теорема Лагранжа о сумме четырёх квадратов и основная теорема арифметики.

4 Выполнение лабораторной работы

4.1 Алгоритм Евклида

4.1.1 Задача

Реализовать алгоритм Евклида

4.1.1.1 Решение

Реализуем алгоритм Евклида (рис. 4.1)

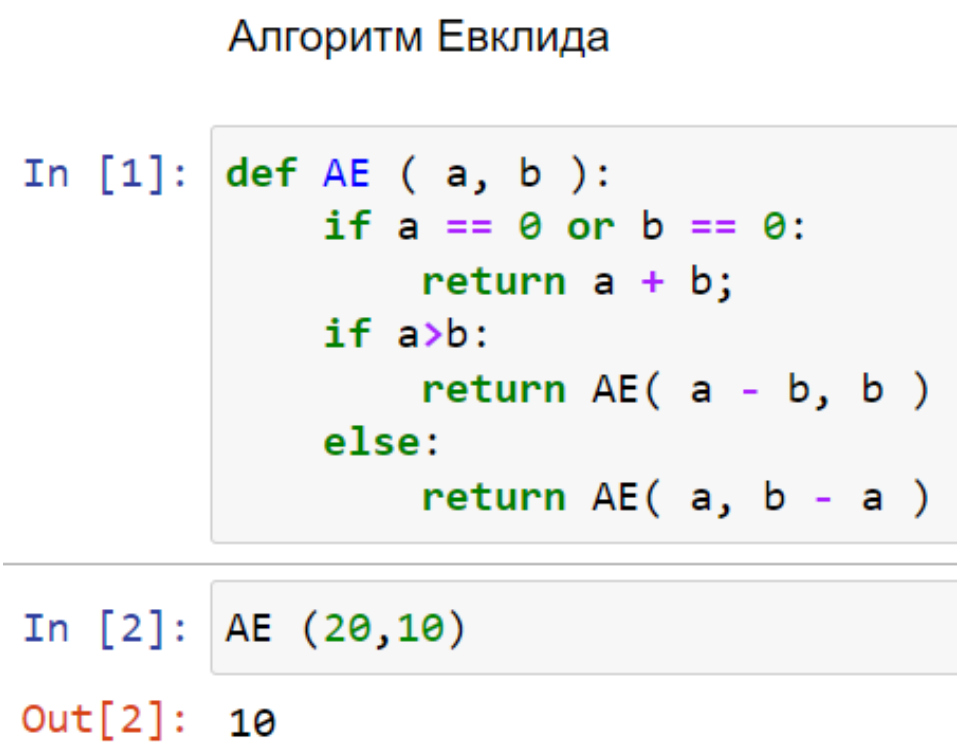

Figure 4.1: Алгоритм Евклида

4.2 Бинарный алгоритм Евклида

4.2.1 Задача

Реализовать бинарный алгоритм Евклида

4.2.1.1 Решение

Реализуем бинарный алгоритм Евклида (рис. 4.2)

Бинарный алгоритм Евклида

In [3]:
```
def BAE ( a, b ):
    g=1
    while True:
        if a%2==0 and b%2==0:
            a = a/2
            b = b/2
            g = g*2
        else:
            u = a
            v = b
            break
    while (u!=0):
        while (u%2 == 0):
            u = u/2
        while (v%2 == 0):
            v = v/2
        if u >= v:
            u = u-v
        else:
            v = v-u
        d = g*v
    return d
```

In [4]:
```
BAE (20,10)
```

Out[4]:
```
10.0
```

Figure 4.2: Бинарный алгоритм Евклида

4.3 Расширенный алгоритм Евклида

4.3.1 Задача

Реализуем расширенный алгоритм Евклида (рис. 4.3)

Расширенный алгоритм Евклида

```
In [5]: def RAE(a, b):
            if b == 0:
                return a, 1, 0

            x1, x0, y1, y0 = 1, 0, 0, 1
            while b > 0:
                q = a // b
                a, b = b, a % b
                x1, x0 = x0, x1 - q * x0
                y1, y0 = y0, y1 - q * y0

            return a,x1,y1

In [6]: RAE(20, 10)

Out[6]: (10, 0, 1)
```

Figure 4.3: Расширенный алгоритм Евклида

4.4 Расширенный бинарный алгоритм Евклида

4.4.1 Задача

Реализуем расширенный бинарный алгоритм Евклида (рис. 4.4)

Расширенный бинарный алгоритм Евклида

In [7]:
```
def RBAE ( a, b ):
    g=1
    while True:
        if a%2==0 and b%2==0:
            a = a/2
            b = b/2
            g = g*2
        else:
            u = a
            v = b
            A = 1
            B = 0
            C = 0
            D = 1
            break
    while (u!=0):
        while (u%2 == 0):
            u = u/2
            if A%2==0 and B%2==0:
                A=A/2
                B=B/2
            else:
                A=(A+b)/2
                B=(B-a)/2
        while (v%2 == 0):
            v = v/2
            if C%2==0 and D%2==0:
                C=C/2
                D=D/2
            else:
                C=(C+b)/2
                D=(D-a)/2
        if u >= v:
            u = u-v
            A = A-C
            B = B-D
        else:
            v = v-u
            C = C-A
            D = D-B
        d = g*v
        x = C
        y = D
    return d,x,y
```

In [8]:
```
RBAE (20, 10)
```

Out[8]:
```
(10.0, 0, 1)
```

Figure 4.4: Расширенный бинарный алгоритм Евклида

5 Выводы

В ходе данной лабораторной работы я рассмотрел и реализовал следующие алгоритмы:

- Алгоритм Евклида;
- Бинарный алгоритм Евклида;
- Расширенный алгоритм Евклида;
- Расширенный бинарный алгоритм Евклида.

6 Библиография

1. Python documentation. [Электронный ресурс]. М. URL: Python documentation (Дата обращения: 28.09.2023).

2. Лабораторная работа №4. Вычисление НОД. - 4 с. [Электронный ресурс]. М. URL: Лабораторная работа №4. Вычисление НОД. (Дата обращения: 19.10.2023).