УДК 004.422.63

МОДЕЛИРОВАНИЕ НЕКОТОРЫХ МЕТОДОВ ПРЕДСТАВЛЕНИЯ *n* FIFO-ОЧЕРЕДЕЙ В ПАМЯТИ ОДНОГО УРОВНЯ [1]

А. В. Соколов А. В. Драц
Институт прикладных математических исследований Карельского научного центра РАН,
Петрозаводский государственный университет
email: avs@krc.karelia.ru email: adeon88@mail.ru

Аннотация. Во многих приложениях требуется работа с несколькими FIFO-очередями, расположенными в общем пространстве памяти. Для этого применяют различные программные или аппаратные решения. В данной работе предлагаются математические модели для последовательного циклического и связанного способов представления n FIFO-очередей в общей памяти. В обоих способах представления для каждой очереди нужны два указателя на начало и конец, но для первого способа элементы равных длин располагаются циклически в последовательных адресах выделенного очереди участка памяти, а во втором каждая очередь представлена в виде односвязного списка элементов, и переполнение памяти наступает тогда, когда список свободных элементов пуст и требуется включить элемент в какую-либо очередь. В этих способах представления операции включения и исключения элементов выполняются за время O(1). В последовательном способе мы имеем потери, в случае когда одна очередь переполнилась, а в других остаётся свободное место. В связном способе потерь нет до тех пор, пока не исчерпана вся свободная память в выделенном для очередей участке памяти, но требуется дополнительная память на связи в каждом элементе списка.

В статье решается задача нахождения оптимального разделения памяти между очередями в случае последовательного циклического представления очередей и задача анализа метода представления очередей в виде связных списков. В качестве математических моделей предложены случайные блуждания по целочисленным решёткам в различных областях *n*-мерного пространства. В качестве критерия оптимальности рассматривается минимальная средняя доля времени, которое системы проводит в состояниях «сброса хвоста». Это эквивалентно минимизации средней доли потерянных элементов. Эту величину разумно минимизировать, когда переполнение очереди является не аварийной, а стандартной ситуацией. В некоторых приложениях при переполнении

[1] Работа поддержана грантом РФФИ № 12-01-00253-а и Программой стратегического развития Петрозаводского государственного университета в рамках реализации комплекса мероприятий по развитию научно-исследовательской деятельности.

очереди работа программы заканчивается, и тогда в качестве критерия оптимальности надо рассматривать максимальное среднее время до переполнения памяти. То есть если очередь занимает всю предоставленную ей память, то все последующие элементы, поступающие в неё, отбрасываются до тех пор, пока не появится свободная память (т.е. до тех пор, пока не произойдёт исключение элемента из очереди). Эта схема работы применяется, например, в работе сетевых маршрутизаторов в том случае, когда по мере увеличения трафика очередь на исходящем интерфейсе маршрутизатора заполняется пакетами. Такое поведение маршрутизатора называется «сбросом хвоста». Потери пакетов приводят к нежелательному результату, поэтому число таких ситуаций необходимо свести к минимуму. Для решения задач используется аппарат регулярных цепей Маркова.

Ключевые слова и фразы: FIFO-очереди; случайные блуждания; цепи Маркова; оптимальные динамические структуры данных.

1 ВВЕДЕНИЕ

Во многих приложениях требуется работа с несколькими FIFO-очередями, расположенными в общем пространстве памяти. Для этого применяют различные программные или аппаратные решения [1–3]. В настоящей работе предлагаются математические модели для последовательного циклического и связанного способов представления очередей [1]. В обоих способах представления для каждой очереди нужны два указателя на начало и конец, но для первого способа элементы равных длин располагаются циклически в последовательных адресах выделенного очереди участка памяти, а во втором каждая очередь представлена в виде односвязного списка элементов, и переполнение памяти наступает тогда, когда список свободных элементов пуст и требуется включить элемент в какую-либо очередь. В обоих способах операции включения и исключения выполняются за время O(1). В качестве критерия оптимальности рассматривается минимальная доля потерянных элементов при бесконечном времени работы очередей. Эту величину разумно минимизировать, когда переполнение очереди является не аварийной, а стандартной ситуацией (здесь мы подчёркиваем, что в некоторых приложениях при переполнении очереди работа программы заканчивается, и тогда в качестве критерия оптимальности надо рассматривать максимальное среднее время до переполнения памяти). То есть если очередь занимает всю предоставленную ей память, то все последующие элементы, поступающие в неё, отбрасываются до тех пор, пока не появится свободная память (т. е. до тех пор, пока не произойдёт исключение элемента из очереди). Такая схема работы применяется, например, в работе сетевых маршрутизаторов [3] в том случае, когда по мере увеличения трафика очередь на исходя-

щем интерфейсе маршрутизатора заполняется пакетами. Такое поведение маршрутизатора называется «сбросом хвоста». Потери пакетов приводят к нежелательному результату, поэтому число таких ситуаций необходимо свести к минимуму.

Мы в этой работе строим математические модели в виде случайных блужданий по целочисленной решётке. Первоначально такие модели в виде случайного блуждания в треугольнике [4–7] были построены для решения задачи анализа процесса работы с двумя стеками, растущими навстречу друг другу, поставленной в [1]. В этих моделях предполагается, что на каждом шаге дискретного времени с заданными вероятностями происходят некоторые операции с очередями. Время выполнения операций – это не случайная величина, а константа, поэтому фиксированным является и шаг времени. В [8] был рассмотрен случай последовательного представления очереди для n=2, в [9] была предложена модель, описывающая последовательное представление очередей для n=3. В [10–12] рассмотрены модели работы с n очередями для разных способов представления очередей и критериев оптимальности.

В данной статье рассмотрены последовательный и связанный способы представления FIFO-очередей для случая произвольного n. Необходимо определить, как распределить память между очередями в последовательном способе организации, и какой из способов организации очередей является оптимальным.

Будем придерживаться следующих обозначений:

- m – размер памяти;
- n – количество очередей в быстрой памяти;
- p_i – вероятность включения элемента в i-ю очередь;
- q_i – вероятность извлечения элемента из i-й очереди;
- r – вероятность того, что не произойдёт операции включения или извлечения;
- k_i – размер памяти, выделенной для очереди с номером i при последовательном представлении;
- x_i – текущая длина очереди с номером i;
- l – отношение размера узла к размеру указателя (для связанного представления);
- P^* – доля времени, которую проводит очередь в состоянии «сброса хвоста».

2 ПОСЛЕДОВАТЕЛЬНОЕ ПРЕДСТАВЛЕНИЕ ОЧЕРЕДЕЙ

2.1. Общие положения

Рассмотрим n FIFO-очередей, расположенных в быстрой памяти раз-

мера m. Для последовательного представления каждой очереди необходимо выделить k_i единиц памяти, где $k_1+\ldots+k_n=m$. Если очередь занимает всю предоставленную ей память, то все последующие элементы, поступившие в неё, отбрасываются до тех пор, пока не появится свободная память.

В качестве математической модели рассмотрим случайное блуждание по целочисленному n-мерному параллелепипеду, ограниченному гиперплоскостями $x_i=0$, $x_i=k_i$, $1 \le i \le n$. Количество состояний равно $\prod_{i=1}^{n}(k_i+1)$. $(x_1,\ldots,x_n)$ – состояние системы $x_1 = k_1+1$, …, $x_n = k_n+1$– состояния «сброса хвоста».

Переход системы из состояния $(x_1,\ldots,x_n)$ в состояние $(x_1',\ldots,x_n')$ происходит в соответствии со следующими правилами:

$$(\ldots,x_i,\ldots,x_j,\ldots)\underset{\rightarrow}{p_i}\begin{cases}(\ldots,x_i+1,\ldots,x_j,\ldots) & 0\le x_i\le k_i, x_j\le k_j\\ (\ldots,x_i+1,\ldots,x_j-1,\ldots) & 0\le x_i\le k_i, x_j=k_j+1\\ (\ldots,x_i,\ldots,x_j,\ldots) & x_i=k_i+1, x_j\le k_j\\ (\ldots,x_i,\ldots,x_j-1,\ldots) & x_i=k_i+1, x_j=k_j+1\end{cases}$$

$$(\ldots,x_i,\ldots,x_j,\ldots)\underset{\rightarrow}{q_i}\begin{cases}(\ldots,x_i,\ldots,x_j,\ldots) & x_i=0, x_j\le k_j\\ (\ldots,x_i,\ldots,x_j-1,\ldots) & x_i=0, x_j=k_j+1\\ (\ldots,x_i-1,\ldots,x_j,\ldots) & 1\le x_i\le k_i, x_j\le k_j\\ (\ldots,x_i-1,\ldots,x_j-1,\ldots) & 1\le x_i\le k_i, x_j=k_j+1\\ (\ldots,x_i-2,\ldots,x_j,\ldots) & x_i=k_i+1, x_j\le k_j\\ (\ldots,x_i-2,\ldots,x_j-1,\ldots) & x_i=k_i+1, x_j=k_j+1\end{cases}$$

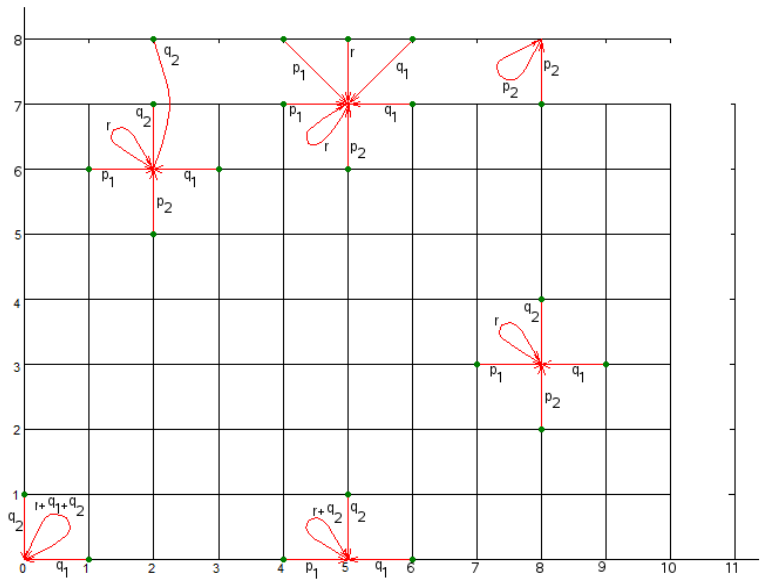

Рисунок 1. Переходы между состояниями
в случае последовательного представления очередей

$$(\dots, x_i, \dots, x_j, \dots)\underset{\rightarrow}{r}\begin{cases}(\dots, x_i, \dots, x_j, \dots) & 0 \le x_i \le k_i, x_j \le k_j\\ (\dots, x_i, \dots, x_j - 1, \dots) & 0 \le x_i \le k_i, x_j = k_j + 1\\ (\dots, x_i - 1, \dots, x_j, \dots) & x_i = k_i + 1, x_j \le k_j\\ (\dots, x_i - 1, \dots, x_j - 1, \dots) & x_i = k_i + 1, x_j = k_j + 1\end{cases}$$

Теорема 1. Пусть $(k_1, \dots, k_n)$ – фиксированное разбиение памяти между очередями, у очередей с номерами $1 \le i \le s$ справедливо $p_i \ne q_i$, а для очередей с номерами $s+1 \le i \le n$ справедливо $p_i = q_i$. Тогда доля времени, которую система проводит в состоянии «сброса хвоста», равна

$$P_c = \sum_{i=1}^{n} \frac{q_i - p_i}{\left(\frac{q_i}{p_i}\right)^{k_i+1} - 1} + \sum_{i=s+1}^{n} \frac{p_i}{k_i + 1}$$

Доказательства теорем приведены в работах [10,11].

2.2. Оптимальное разбиение памяти. Случай равных вероятностей

Теорема 2. Пусть у всех очередей равны вероятности включения и исключения элементов $p_i=q_i$ $(1 \le i \le n)$. Тогда:

$P_c^* \ge \sum_{i=1}^{n} \frac{p_i}{k_i+1} = \frac{\left(\sum_{i=1}^{n} \sqrt{p_i}\right)^2}{m+n}$, где $k_i^* = \frac{(m+n)\sqrt{p_i}}{\sum_{l=1}^{n} p_l} - 1$

$(k_1^*, \dots, k_n^*) = \operatorname{argmin}_{k_1 + \dots + k_n = m} \sum_{i=1}^{n} \frac{p_i}{k_i+1}$

При этом $\left|k_i^* - \overline{k_i}\right| < 1$, где $(\overline{k_1}, \dots, \overline{k_n})$ – оптимальное разбиение памяти $(k_i^* \in R, \overline{k_i} \in \{N \cup 0\})$.

Чтобы найти оптимальное разбиение памяти, можно округлить числа k_i^* вниз, а оставшуюся часть памяти (размером меньше n единиц) распределить между очередями по методу, описанному в следующем разделе.

2.3. Оптимальное разбиение памяти. Общий случай

Введём обозначение:

$$Z_m = \min_{k_1 + \dots + k_n = m} \left(\sum_{i=1}^{n} p_i^* (k_i) \right)$$

где Z_m – доля времени, которое процесс проводит в состоянии «сброса хвоста» при оптимальном разбиении памяти размера m между очередями, $p_i^*(k_i)$ – доля времени, в течение которого происходит потеря элементов только для i-й очереди при размере памяти k_i.

Рассмотрим рекуррентную формулу

$$Z_m = Z_{m-1} - \max_{1 \le i \le n} (p_i^*(k_i) - p_i^*(k_i + 1))$$

Начальное значение Z_0 равно:

$$Z_0 = \sum_{i=1}^{n} \frac{q_i - p_i}{\left(\frac{q_i}{p_i}\right)^{0+1} - 1} = \sum_{i=1}^{n} p_i$$

То есть при отсутствии памяти любая попытка включения элемента в одну из очередей будет приводить к его потере.

Был реализован эффективный алгоритм решения задачи на основе предложенной модели динамического программирования, который за время O(mn) вычисляет оптимальное разбиение памяти и долю времени, проведённого в состояниях «сброса хвоста». Также была построена математическая модель этого процесса в виде случайного блуждания по целочисленному n-мерному параллелепипеду с вершиной в начале координат, рёбрами, параллельными осям координат, и известными длинами рёбер. Гиперплоскости соответствуют состояниям «сброса хвоста». Была предложена нумерация состояний, и на её основе разработан алгоритм генерации соответствующей цепи Маркова и решения задачи с использованием результатов теории регулярных цепей Маркова [13]. Этот метод решения задачи будет подробно изложен в следующем разделе на примере анализа связанного представления очередей. Метод динамического программирования в данном случае приводит к более эффективному алгоритму.

3 СВЯЗАННОЕ ПРЕДСТАВЛЕНИЕ ОЧЕРЕДЕЙ

При связанном представлении каждая очередь хранится в виде связанного списка. Часть памяти тратится на указатели, l – отношение размера указателя к размеру информационной части. $M = m(1 - 1/l)$ – размер памяти, который тратится на размещение информационных частей, он же равен общему количеству элементов, которые можно разместить в памяти. Например, если элемент списка объявлен как

```
struct node{DataT info; node * link;};
```

то `sizeof(info)= 4*sizeof(link); sizeof(info) = 4` (байта), и размер памяти равен 40 байт, то m=10, l=1/4 и M=m/(1+l)=8. (Единица памяти равна 4 байта.)

В качестве математической модели рассмотрим блуждание по целочисленной n-мерной пирамиде с рёбрами $0 \le x_1 \le M$, $0 \le x_2 \le M$,…, $0 \le x_n \le M$ и основанием $x_1+x_2+\ldots+x_n=M$. Для каждого состояния $(x_1, \ldots, x_n)$ на плоскости $x_1+x_2+\ldots+x_n=M$, т.е. когда вся память уже занята, введём состояние, соответствующее «сбросу хвоста». В это состояние можно попасть в случае попытки включить элемент в любую из

очередей, когда вся память занята. Переход процесса из состояния $(x_1, \dots, x_n)$ определяется по следующим правилам:

$$(\dots, x_i, \dots, x_j, \dots)\underset{\rightarrow}{p_i}\begin{cases}(\dots, x_i+1, \dots, x_j, \dots) & 0 \le x_1 + \dots + x_n \le M \\ (\dots, \overline{x_i}, \dots, \overline{x_j}, \dots) & x_1 + \dots + x_n = M\end{cases}$$

$$(\dots, x_i, \dots, x_j, \dots)\underset{\rightarrow}{q_i}\begin{cases}(\dots, x_i-1, \dots, x_j, \dots) & x_i > 0 \\ (\dots, \overline{x_i}, \dots, \overline{x_j}, \dots) & x_i = 0\end{cases}$$

$$(\dots, x_i, \dots, x_j, \dots)\underset{\rightarrow}{r}(\dots, x_i, \dots, x_j, \dots)(\dots, \overline{x_i}, \dots, \overline{x_j}, \dots)\underset{\rightarrow}{p_i}(\dots, \overline{x_i}, \dots, \overline{x_j}, \dots)$$

$$(\dots, \overline{x_i}, \dots, \overline{x_j}, \dots)\underset{\rightarrow}{q_i}\begin{cases}(\dots, x_i-1, \dots, x_j, \dots) & x_i > 0 \\ (\dots, \overline{x_i}, \dots, \overline{x_j}, \dots) & x_i = 0\end{cases}(\dots, \overline{x_i}, \dots, \overline{x_j}, \dots)\underset{\rightarrow}{r}(\dots, x_i, \dots, x_j, \dots)$$

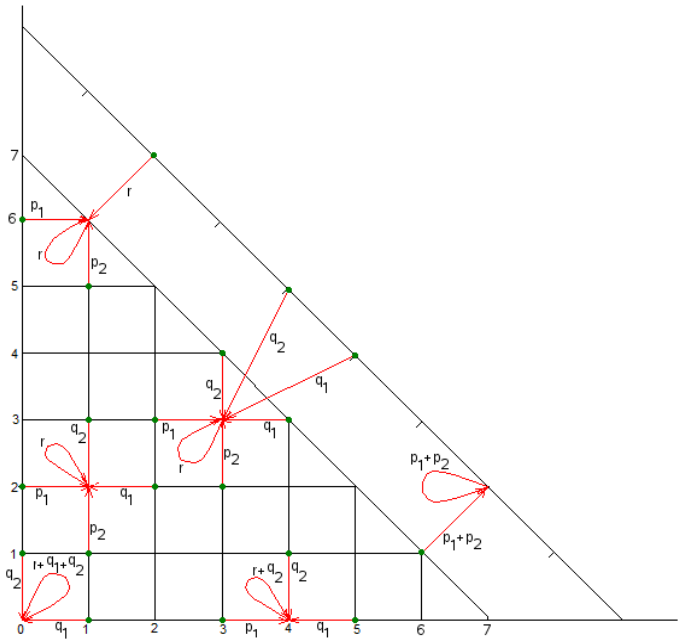

Рисунок 2. Переходы между состояниями
в случае связанного представления очередей

Обозначим $y_i = p_i/q_i$.

Теорема 3. Доля времени, которую система проводит в состоянии «сброса хвоста» при связанном представлении при условиях $y_i \neq 1$ и $y_i \neq y_j$, при $i \neq j$ равна

$$P_l = (p_1 + \dots + p_n)\frac{\sum_{i=1}^{n} \dfrac{y_i^{M+n-1}}{\prod_{\substack{j=1 \\ j\neq i}}^{n}(y_i - y_j)}}{\sum_{i=1}^{n} \dfrac{y_i^{M+n}}{(y_i-1)\prod_{\substack{j=1 \\ j\neq i}}^{n}(y_i - y_j)} + \dfrac{1}{\prod_{j=1}^{n}(1-y_j)}}$$

Далее рассмотрим общий случай. Пусть есть:

k_0 очередей с вероятностями $\frac{p_{i_0}}{q_{i_0}} = \cdots = \frac{p_{i_{k_0}}}{q_{i_{k_0}}} = x_0 = 1$,

k_1 очередей с вероятностями $\frac{p_{i_1}}{q_{i_1}} = \cdots = \frac{p_{i_{k_1}}}{q_{i_{k_1}}} = x_1$

…

k_s очередей с вероятностями $\frac{p_{i_s}}{q_{i_s}} = \cdots = \frac{p_{i_{k_s}}}{q_{i_{k_s}}} = x_s$

$k_0+\ldots+k_s = n$.

Теорема 4. Доля времени, которую система проводит в состоянии «сброса хвоста» при связанном представлении равна

$$P_l = \frac{(p_1 + \cdots + p_n)}{k_0} \frac{\frac{\partial^{n-s-1}}{\partial^{k_0-1} y_0 \partial^{k_1-1} y_1 \ldots \partial^{k_s-1} y_s} \left\{ \sum_{i=1}^{n} \frac{y_i^{M+n-1}}{\prod_{\substack{j=1 \\ j\neq i}}^{n} (y_i - y_j)} \right\}}{\frac{\partial^{n-s}}{\partial^{k_0} y_0 \partial^{k_1-1} y_1 \ldots \partial^{k_s-1} y_s} \left\{ \sum_{i=1}^{n} \frac{y_i^{M+n}}{\prod_{\substack{j=1 \\ j\neq i}}^{n} (y_i - y_j)} \right\}}$$

4 СРАВНЕНИЕ ПОСЛЕДОВАТЕЛЬНОГО И СВЯЗАННОГО ПРЕДСТАВЛЕНИЙ

В этом разделе приводятся результаты сравнения последовательного и связанного представлений. Результаты справедливы в предельной форме, когда $m \to \infty$, но, как показывают результаты тестов, они имеют место в допредельной форме, когда размер памяти довольно мал (около 10-20 единиц). Для вычислений использовалась система векторной алгебры Maxima [14]. Рассмотрим несколько случаев зависимостей между вероятностными характеристиками.

1. $p_1 > q_1$ и $p_1/q_1 > p_i/q_i$, $2 \le i \le n$.

Последовательное представление:

$$lim_{m\to\infty} \frac{q_i - p_i}{\left(\frac{q_i}{p_i}\right)^{k_i+1} - 1} = \begin{cases} p_i - q_i & p_i > q_i \\ 0 & p_i < q_i \end{cases}$$

$lim_{m\to\infty} \frac{p_i}{k_i+1}$=0

Связанное представление:

$$\lim_{m\to\infty} P_l^* = (p_1 + \cdots + p_n) \frac{y_i+1}{y_i} = (p_1 + \cdots + p_n) \left(1 - \frac{q_1}{p_1}\right).$$

$p_i \left(1 - \frac{q_1}{p_1}\right) > 0$, $2 \le i \le n$, поскольку $p_1 > q_1$.

$$p_i\left(1-\frac{q_1}{p_1}\right)=p_i-p_i\frac{q_1}{p_1}>p_i-q_i\ .$$

Следовательно, $p_i\left(1-\frac{q_1}{p_1}\right)>max(p_i-q_i,0)$.

Просуммируем последнее неравенство по i от 2 до n и прибавим p_1-q_1:

$$\lim_{m\to\infty}P_c^*=\sum_{i=1}^{n}\max(p_i-q_i,0)\le(p_1+\cdots+p_n)\left(1-\frac{q_1}{p_1}\right)=\lim_{m\to\infty}P_l^*,$$

т.е. $P_c^*\le P_l^*$ даже при небольших размерах памяти.

2. $p_i=q_i=\frac{1}{2n}$, $1\le i\le n$.

$P_c^*\le\sum_{i=1}^{n}\frac{p_i}{k_i+1}=\sum_{i=1}^{n}\frac{\frac{1}{2n}}{\frac{m}{n}+1}=\frac{1}{2}\frac{n}{m+n}$, $P_l^*=\frac{n}{M+n}$.

Следовательно, $P_c^*\le P_l^*$.

3. $p_i<q_i$, i=1,…,n и $p_1/q_1>p_i/q_i$, $2\le i\le n$.

Как следует из вычисления оптимального разбиения памяти между очередями, все очереди будут проводить примерно одну и ту же долю времени в состоянии «сброса хвоста», т.е.

$$\frac{q_i-p_i}{\left(\frac{q_i}{p_i}\right)^{k_i+1}}\approx\frac{q_j-p_j}{\left(\frac{q_j}{p_j}\right)^{k_j+1}}\quad i\ne j.$$

Используя равенство k_1+…+k_n=m, можно найти примерное значение величин k_1,…,k_n:

$$P_c^*=\frac{1}{\exp\left(\frac{m-n}{\sum_{i=1}^{n}\frac{1}{\log\left(\frac{q_1}{p_1}\right)}}-\log(n)\right)}=O\left(\exp\left(-\frac{m}{\sum_{i=1}^{n}\frac{1}{\log\left(\frac{q_1}{p_1}\right)}}\right)\right)$$

Поведение функции $P_l\ (M)$ будет определяться величиной $\sum_{i=1}^{n}\frac{y_i^{M+n-1}}{\prod_{\substack{j=1\\j\ne i}}^{n}(y_i-y_j)}$. Поскольку знаменатель $\frac{y_i^{M+n}}{(y_i-1)\prod_{\substack{j=1\\j\ne i}}^{n}(y_i-y_j)}\to 0$, и величина $\frac{1}{\prod_{j=1}^{n}(1-y_j)}$ является константой, то

$$P_c^*=O\left(\left(\frac{p_1}{q_1}\right)^M\right)=O\left(\exp\left(-m(1-\tfrac{1}{l})\log\left(\frac{q_1}{p_1}\right)\right).\right.$$

В случае, если вероятности включения элементов в структуры данных меньше, чем вероятности исключения, доля времени, которую система проводит в состоянии «сброса хвоста», экспоненциально стремится к 0 для обоих представлений. Предпочтительнее использовать то представление, у которого показатель экспоненты меньше.

r	0.2	0	0	0	0
p_1	0.1	0.44	0.5	0.08	0.15
p_2	0.1	0.02	0.07	0.05	0.08
p_3	0.1	0.02	0.07	0.05	0.05
p_4	0.1	0.02	0.07	0.05	0.02
q_1	0.1	0.44	0.08	0.5	0.35
q_2	0.1	0.02	0.07	0.07	0.2
q_3	0.1	0.02	0.07	0.07	0.1
q_4	0.1	0.02	0.07	0.07	0.05
P_1	0.08	0.1	0.462	0.042	0.0049
P_2	0.08	0.06	0.459	0.039	0.0061
l_1	0.1333	0.1667	0.5964	0.08	0.0067
l_2	0.1143	0.1429	0.5964	0.0677	0.0022
l_4	0.1	0.125	0.5964	0.0589	0.00068
l_8	0.089	0.1111	0.5964	0.0517	0.0000198

Таблица 1. M=16, n=4.

В таблице 1 приведено несколько численных результатов. В строке P_1 указана доля времени, проведённого в состояниях «сброса хвоста», когда память разделена поровну между очередями (поскольку на практике вероятности включения и исключения, которые считались известными, не всегда могут быть вычислены), в строке P_2 указана минимальная доля времени, проведённого в состояниях «сброса хвоста», когда память разделена оптимально. В строке l_1 указана доля времени, проведённого в состояниях «сброса хвоста» для связанного представления – когда на связи тратится 1/2 часть памяти (размер информационной части равен размеру указателя); в строке l_2 – когда на связи тратится 1/3 часть памяти (размер указателя равен 1/2 размера информационной части); в строке l_4 – когда на связи тратится 1/5 часть памяти (размер указателя равен 1/4 информационной части); в строке l_8 – когда на связи тратится 1/9 часть памяти (размер указателя равен 1/8 информационной части).

СПИСОК ЛИТЕРАТУРЫ

1. Д. Кнут. Искусство программирования для ЭВМ. Т. 1. М.: Виль-

ямс, 2001. – 712 с.
2. Р. Седжвик. Фундаментальные алгоритмы на С++. К.: Издательство «Диасофт», 2001. – 687 с.
3. В. Боллапрагада, К. Мэрфи, У. Расс. Структура операционной системы Cisco IOS. М.: Вильямс, 2002. – 197 с.
4. А. В. Соколов. О распределении памяти для двух стеков // Автоматизация эксперимента и обработки данных. 1980. – С.65-71.
5. P. Flajolet. The evolution of two stacks in bounded space and random walks in a triangle // Lec. Notes Computer Sci. Vol. 223, 1986. – P.325-340.
6. G. Louchard, R. Schott, M. Tolley, P. Zimmermann. // Comput. Appl. Math. №53, 1994. P.243-274.
7. Е. А. Аксенова, А. В. Соколов. Оптимальное управление двумя параллельными стеками // Дискретная математика. №1, 2007. – С.67-75.
8. А. В. Соколов, А. В. Тарасюк. Об оптимальном управлении циклическими FIFO-очередями // Системы управления и информационные технологии. №3, 2005. – С.29-33.
9. Е. А. Аксенова. Оптимальное управление FIFO-очередями на бесконечном времени // Межвуз. сб. «Стохастическая оптимизация в информатике». Изд-во С.-Петербургского университета. №2, 2006. С.71-76.
10. Е. А. Аксенова, А. В. Драц, А. В. Соколов. Оптимальное управление n FIFO-очередями на бесконечном времени // Информационно-управляющие системы. №6, 2009. – С. 401-415.
11. A. V. Sokolov, A. V. Drac. The optimal implementation of n FIFO-queues in single-level memory // Proceedings of AMICT 2010-2011 Advances in Methods of Information and Communication Technology, Helsinki. 2012. – P.51-65.
12. A. V. Sokolov, A. V. Drac. The linked list representation of n LIFO-stacks and/or FIFO-queues in the single-level memory // Information Processing Letters. Elsevier Science Publishing Company Inc. Vol. 13, №19-21, 2013. P.832-835.
13. Дж. Кемени, Дж. Снелл. Конечные цепи Маркова. М.: Наука, 1960. – 272 с.
14. Maxima, a Computer Algebra System. [Электронный ресурс] – Режим доступа: `http://maxima.sourceforge.net/`

SIMULATION OF SOME METHODS OF REPRESENTATION OF *n* FIFO-QUEUES IN THE SINGLE-LEVEL MEMORY

A. V. Sokolov A. V. Drac
Karelian research center of the Russian Academy of Sciences,
Petrozavodsk State University
email: avs@krc.karelia.ru email: adeon88@mail.ru

Abstract. In many applications there is a problem of representation of multiple FIFO-queues in the single-level memory. In this article we research methods of n FIFO-queues allocation in the shared memory. There are two fundamentally different ways of organizing work with queues – consecutive and linked list representation. For sequential cyclic queues each of them should be allocated in its own part of memory. In this case we will have losses of memory when any queue overflow and other data structures don't overflow. In linked representation the data structure is stored as a list. In this case any number of elements of the lists can coexist in the memory area until the free memory is exhausted. But on the other hand, this approach requires an additional link field for each element. The problem of optimal memory partition between queues in the case of consecutive circular implementation and the problem of the analysis of linked list implementation are investigated. As mathematical models we proposed random walks on the integer lattice in some regions of n-dimensional space. As the optimality criterion we considered the minimal average part of time which the system is situated in the state of "reset tail". It is reasonably to minimize this value in order to minimize the part of lost elements. This value is reasonably minimized when the overflow queue is the standard situation. This scheme is applied , for example, in the network routers. Such behavior of the router called "reset tail". Packet loss leads to undesirable results, so the number of such situations need to be minimized. To solve this problem we used apparatus of regular Markov chains.

Keywords and phrases: FIFO-queues; random walks; Markov chains; optimal dynamic data structures.

REFERENCES

1. D. E. Knuth. The Art of Computer Programming. Vol. 1. Addison-Wesley, Reading, MA, 2001.
2. R. Sedgwick. Algorithms in C++. Third Edition. Parts 1-4. Addison Wesley Longman, 1999.
3. V. Bollapragada, C. Murphy, R. White. Inside Cisco IOS. Software Architecture. Cisco Press, 2000.
4. A. Sokolov. About storage allocation for implementing two stacks. In: Automation of experiment and data processing. Petrozavodsk. 1980. P. 65-71 (in Russian).
5. P. Flajolet. The evolution of two stacks in bounded space and random walks in a triangle. // Lec. Notes Computer Sci. 1986. Vol. 223. P.325-340.
6. G. Louchard, R. Schott, M. Tolley, P. Zimmermann. Random walks, heat equation and distributed Algorithms. J. Comput. Appl. Math. 1994. Vol. 53. P. 243-274.
7. E. A. Aksenova, A. V. Sokolov. The optimal Management of two Parallel Stacks in Two-Level Memory // Discrete Mathematics and Applications. 2007. Vol. 17,

No. 1., P. 47-56.
8. A. V. Sokolov, A. V. Tarasuk, The optimal control of cyclic FIFO-queues. Control Systems and Information Technologies, Vol, 3, No. 20, P. 29-33 (in Russian).
9. E. A. Aksenova. The optimal control of FIFO-queues for infinite time. Stochastic Optimization in Informatics, 2006. Vol. 2, P. 71-76 (in Russian).
10. E. A. Aksenova, A.V. Drac, A.V. Sokolov. The optimal control of n FIFO-queues for infinite time. Information and Control Systems. Vol. 6, P. 46-54. 2009 (in Russian).
11. A. V. Sokolov, A.V. Drac. The optimal implementation of n FIFO-queues in single-level memory. Proceedings of AMICT 2010-2011 Advances in Methods of Information and Communication Technology, Helsinki-2012, P. 51-65. Electronic resource. Access mode: `https://helda.helsinki.fi/handle/10138/41066`
12. A. V. Sokolov, A. V. Drac. The linked list representation of n LIFO-stacks and/or FIFO-queues in the single-level memory. Information Processing Letters. Elsevier Science Publishing Company Inc, 2013, Vol. 13, Iss. 19-21, P. 832-835.
13. J. G. Kemeny, J. L. Snell. Finite Markov Chains. Van Nostrand, Princeton, New Jersey, 1960.
14. Maxima, a Computer Algebra System. Electronic resource. Access mode: `http://maxima.sourceforge.net/`