Applied Mathematics, 2011, 2, 1297-1302
doi:10.4236/am.2011.210180 Published Online October 2011 (http://www.SciRP.org/journal/am)

Optimal Implementation of Two FIFO-Queues in Single-Level Memory

Elena A. Aksenova, Andrew V. Sokolov
Institute of Applied Mathematical Research, *Karelian Research Center*,
Russian Academy of Sciences, *Petrozavodsk*, *Russia*
E-mail: *aksenova@krc.karelia.ru*, *avs@krc.karelia.ru*
Received November 19, 2010; *revised January* 11, 2011; *accepted January* 18, 2011

Abstract

This paper presents mathematical models and optimal algorithms of two FIFO-queues control in single-level memory. These models are designed as two-dimensional random walks on the integer lattice in a rectangular area for consecutive implementation and a triangle area for linked list implementation.

Keywords: FIFO-Queues, Random Walks, Markov Chains, Consecutive Implementation, Linked List Implementation, Paged Implementation

1. Introduction

Let two stacks grow towards each other in the shared memory of size *m*. D. Knuth set the task of constructing a mathematical model of this process [1]. In [2-6] a mathematical model of the process was constructed as two-dimensional random walk in a triangle with two reflecting and one absorbing barriers. In [7-8] we consider the problem of one and two stacks managing in two-level memory. This paper proposes mathematical models and optimal algorithms of two FIFO-queues control [9]. These data structures are often used in control and automation systems.

In the case of single-level memory, several allocation methods of FIFO-queues in the memory may be used [1]. The first method (Garvic's algorithm) is the consecutive allocation of one queue after another. In this case we have memory losses due to the fact that when any queue overflow, other queue can have free memory units. The linked list implementation is the second method. In this case any number of lists can coexist inside a shared area of memory until the list of free memory isn't exhausted. On the other hand, this method requires an additional link field for each element. The third method is storage of elements in the linked lists of fixed size pages. In this case we have memory losses of the first and the second types.

On the basis of proposed models the methods of queues implementation are compared, the optimality criterion—maximizing of the average number of operations, performed with queues until memory overflow. Execution time of each operation is constant and it isn't included in the optimality criterion. However, it is important to take into account the specific of hardware and software in the queues implementation methods. In [1] provides such time assessment of element insertion in the stack. When consecutive implementation is used, it equals 12 cycles, and when linked list—it equals 17 cycles. For the operation of element deletion this assessments are equal approximately.

2. Consecutive Implementation of Two FIFO-Queues

Assume that we want to work with two consecutive circular FIFO-queues in the single-level memory of size *m*. Denote the insertion and deletion probabilities of the elements in the first queue by p_1 and q_1, in the second queue—by p_2 and q_2, the probability of operation which doesn't change the queue length (for example only reading)—by *r*, where $p_1 + q_1 + p_2 + q_2 + r = 1$. Denote the current lengths of queues by x_1 and x_2, the length of memory, separated to the first queue—by *s*, to the second queue—by *n-s* (**Figure 1**). The value $n = m - 2$ is the total maximal size of two queues, because one memory element remains free [1] for the circular realization of FIFO-queue.

As a mathematical model we have two-dimensional random walk on the integer lattice in the bounded area $0 \le x_1 < s + 1$, $0 \le x_2 < n - s + 1$ (**Figure 2**) with the transition probabilities : p_1—to the right, q_1—to the left, p_2—

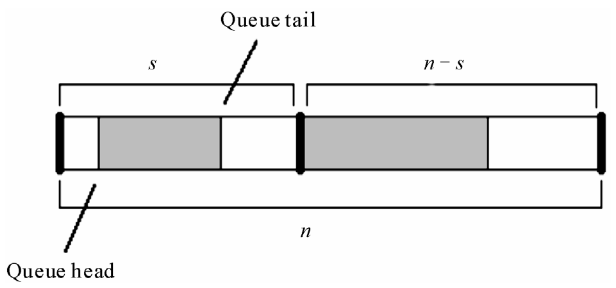

Figure 1. Consecutive allocation.

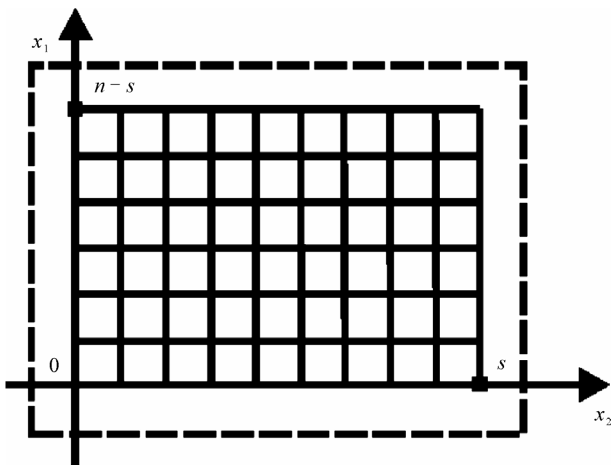

Figure 2. Random walk area.

upward, q_2—downward, r—to remain on the place. The random walk begins at the state $x_1 = x_2 = 0$, where the lines $x_1 = -1$ and $x_2 = -1$ are reflecting barriers, the lines $x_1 = s + 1$ and $x_2 = n - s + 1$—absorbing barriers. The problem is to choose the value s so that the expectation of the random walk time, until the absorption occurs, would be maximal. Number the points of walk area top-down and right to left beginning by 0. These points correspond to the nonrecurring states of the Markov chain [10]. We have $(s + 1)(n - s + 1)$ such states.

The tasks are solved with the apparatus of absorbing Markov chains. The proposed algorithm of states numbering makes possible to estimate the transition matrix, corresponding to the Markov chain.

Theorem 2.1 *At the given states numbering, memory size m and value s the matrix Q has the structure*:

$$Q_{z\times z} = \begin{pmatrix} A_{k\times k} & B_{k\times k} & & \\ C_{k\times k} & \cdots & \cdots & \\ & \cdots & A_{k\times k} & B_{k\times k} \\ & & C_{k\times k} & A'_{k\times k} \end{pmatrix},$$

where

$$z = (s+1)(n-s+1),\ k = n-s+1,$$

the submatrixes have the structures:

$$A_{k\times k} = \begin{pmatrix} r & q_2 & & \\ p_2 & \cdots & \cdots & \\ & \cdots & r & q_2 \\ & & p_2 & r+q_2 \end{pmatrix},$$

$$B_{k\times k} = \begin{pmatrix} q_1 & & \\ & \cdots & \\ & & q_1 \end{pmatrix},$$

$$C_{k\times k} = \begin{pmatrix} p_1 & & \\ & \cdots & \\ & & p_1 \end{pmatrix},$$

the submatrix $A'_{k\times k}$ *has the same structure as* $A_{k\times k}$, *but value* q_1 *is added to the each element of the main diagonal.*

The Proof is based on the mathematical induction method.

3. Linked List Implementation of Two FIFO-Queues

For linked list implementation it is enough to have one-linked list of elements (**Figure 3**). Each queue element consists of two memory units of the same size. The first unit contains stored information and the second unit contains a pointer to the following element of list.

Denote the current lengths of queues by x_1 and x_2, the pointers to the first queue elements—by F_1 and F_2, the pointers to the last queue elements—by R_1 and R$_2$. Assume that m is divisible by 2. In such queue implementation $m/2$ memory units are spent for the pointers, $m/2$ memory units are spent for the stored information. As a mathematical model we have two-dimensional random walk on the integer lattice in the region $0 \le x_1 + x_2 < m/2 + 1$, where the border $x_1 + x_2 = m/2+1$ is an absorbing barrier (it means that the overflow of some queue occurs), the borders $x_1 = -1$ and $x_2 = -1$—are reflecting barriers (it means that the underflow of some queue occurs) (**Figure 4**).

The random walk begins at the state $x_1 = x_2 = 0$. Necessary to find an average time of random walk until the absorption occurs. It is denoted by T. Then we should compare it with the average time of consecutive and paged implementation.

4. Paged Implementation of Two Queues

In this way the queues are implemented as a linked list of the same size pages. The page size equals x memory units. Assume that m is divisible by x, and then the number of pages equals m/x. The number of pointers equals m/x,

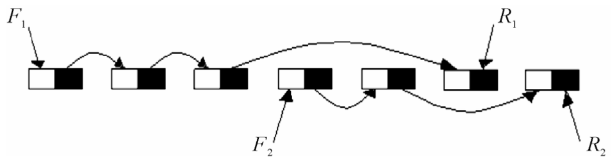

Figure 3. Linked list implementation.

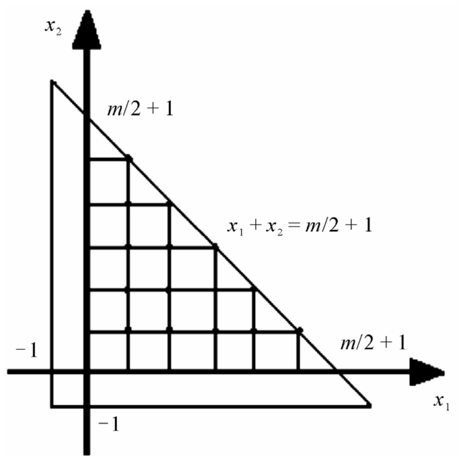

Figure 4. Random walk area.

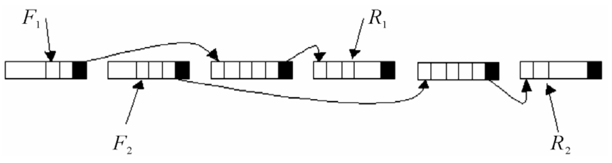

Figure 5. Paged implementation.

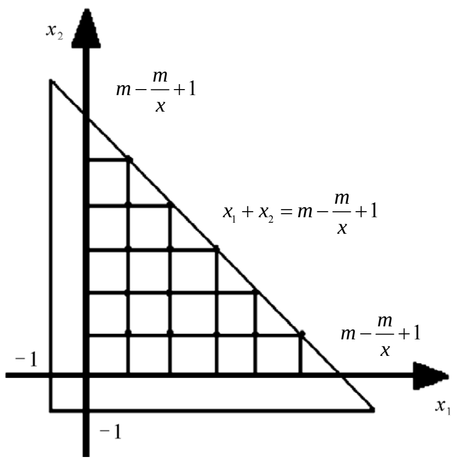

Figure 6. Random walk area.

$x-1$ memory units of each page are used for the information storing and one unit is used for the pointer storing (**Figure 5**). Also, assume that a control algorithm of free page list exists. The algorithm gives new page to the overflowing queue. If the page, which contains a queue beginning, becomes empty, that this page returns to the free page list. If the free page list is empty and the tail page of some queue overflows, the queue overflow occurs. Notice, if $x = 2$, it is the linked list implementation, *i.e.* possible to consider the linked list implementation of queues as a special case of the paged implementation.

Denote the current lengths of queues by x_1 and x_2, the pointers to the first queue elements—by F_1 and F_2, the pointers to the last queue elements—by R_1 and R_2. As a mathematical model we have two-dimensional random walk on the integer lattice in the region $0 \le x_1 + x_2 < m - m/x + 1$, where the border $x_1 + x_2 = m - m/x + 1$ is an absorbing barrier, the borders $x_1 = -1$ and $x_2 = -1$ – are reflecting barriers (**Figure 6**). The random walk begins at the state $x_1 = x_2 = 0$.

The task is to find an average time of random walk. Examine the process of random walk as a finite uniform Markov chain. Let's number the nonrecurring states so like it is presented on the picture below (**Figure 7**), *i.e.* bottom-up, from the left to the right. Determine the scheme of random walk with the probabilities of transitions so: p_1—to the right, q_1—to the left, p_2—upward, q_2—downward, r – to remain on the place. The number of memory units without pointers equals $n = m - m/x$. Then the number of the nonrecurring states of Markov chain

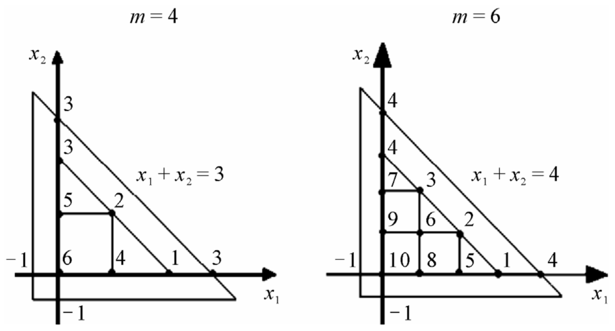

Figure 7. Random walk area.

equals $(n + 1)(n + 2)/2$.

Examine the transition probability matrix Q, describing transitions from the nonrecurring states to the nonrecurring ones. At the given numbering the matrix Q has a block structure. Obviously, that the matrix structure of the linked list implementation coincides with the matrix structure of the page implementation.

Theorem 4.1 *At the given states numbering, memory size m and value s the matrix Q has the structure*:

$$Q_z = \begin{pmatrix} A_k & B_k & & & & \\ C_k & A_{k-1} & B_{k-1} & & & \\ & C_{k-1} & \cdots & \cdots & & \\ & & \cdots & A_{2\times 2} & B_{2\times 1} \\ & & & C_{1\times 2} & r+q_1+q_2 \end{pmatrix},$$

$$z = (n+1)(n+2)/2,$$

where the submatrixes have the dimensions $A_k = A_{k\times k}$, $B_k = B_{k\times(k-1)}$, $C_k = C_{(k-1)\times k}$ *and the structures*:

$$A_k = \begin{pmatrix} r+q_2 & & & & \\ & r & & & \\ & & \cdots & & \\ & & & r & \\ & & & & r+q_1 \end{pmatrix},$$

$$A_{1\times 1} = \left(r + q_1 + q_2\right),$$

$$B_k = \begin{pmatrix} q_1 & & \\ q_2 & \cdots & \\ & \cdots & q_1 \\ & & q_2 \end{pmatrix},$$

$$C_k = \begin{pmatrix} p_1 & p_2 & & \\ & \cdots & \cdots & \\ & & p_1 & p_2 \end{pmatrix},$$

$$k = n+1, n, \cdots, 3, 2.$$

The Proof is based on the mathematical induction method.

Denote by $F(x)$ the average number of memory units, which actually used for stored data, if overflow occurs. Let's suppose that pages stored the queue beginning and the queue end, are half-filled, if the overflow occurs. The number of these pages equals 3. Then $F(x) = m - m/x - 3(x-1)/2$, $m > 0$, $x > 0$. Lets find a maximum of $F(x)$. The derivative $F'(x) = m/x^2 - 3/2 = 0$, the root $x^* = \sqrt{6m}/3$.

The second derivative $F''(x) = -2m/x^3 < 0$, $m > 0$, $x > 0$, then x^* is the maximum of $F(x)$. Consider the values $\lfloor x^* \rfloor$ and $\lceil x^* \rceil$ as an optimal page size, because the mathematical model is discrete.

5. Decision of the Task

For decision of the task we shall find the fundamental matrix $N = (I - Q)^{-1}$ [10], where I is an unitary matrix. In the cases of linked list and page implementations we work with the general resource of the memory. It implies that when the overflow of some queue occurs, all memory units are filled. Indeed, page, which contains the queue beginning, can't be filled fully. In this case another queue can't use this page, but pages, which contains its beginning and end, are not filled fully too. The overflowed queue can't use these pages. Actually the random walk occurs in smaller volume of the memory. So the calculated average time is the upper estimate for real random walk time. It is denoted by $T_{\max}$. Also, possible to compute the lower estimate of real random walk time if consider that when the overflow of some queue occurs, pages, which contain the beginning of the overflowed queue and the beginning and the end of another queue, are stored on one unit.

Then consider the random walk in triangular area $0 \le x_1 + x_2 < m - m/x + 1$ for the upper estimate, and $0 \le x_1 + x_2 < m - m/x - 3(x-2) + 1$ for the lower estimate, where the summand $3(x-2)$ is three pages, which stored on one unit. Notice that the linked list implementation hasn't such losses. For any memory size m we define when the random walk area for the linked list implementation is less than the random walk area for the page implementation in the case of lower estimate of the average time. Consider the inequality $m - m/x - 3(x-2) > m/2$, $m > 0$, $x > 2$.

If x is an arbitrary page size, get $2 < x < m/6$, *i.e.* the random walk area in the case of linked list implementation is always less than the random walk area in the case of paged implementation for lower estimate, if x satisfies the inequality above. Therefore, if $2 < x < m/6$, the average time of random walk until absorption in the case of linked list implementation is less than in the case of paged implementation for the lower estimate.

If $x = \sqrt{6m}/3$, then $m - 3\sqrt{6m} + 12 > 0$. Get $24 < m < \infty$, *i.e.* for $m > 24$ at $x = \sqrt{6m}/3$ the random walk area in the case of linked list implementation is always less than the random walk area in the case of paged implementation for lower estimate. Therefore, for $m > 24$ at $x = \sqrt{6m}/3$ the average time of random walk until absorption in the case of linked list implementation is less than in the case of paged implementation for the lower estimate.

6. Results of Numerical Experiments

For this problems solution the algorithms and computational programs in C++ were developed. Numerical results are presented in **Tables 1-2**. The average time of the random walk is presented in **Table 1** for memory size $m = 12$ in the case, when each consecutive circular FIFO-queue has the same size of memory. In **Table 2** the average time is presented for different queues implementations. In the case of consecutive implementation the memory is divided optimally depending on queues probabilities. In the first five columns there are the queues probabilities, in the following three columns there is the average time of the random walk for A_1—consecutive implementation, A_2—paged implementation, A_3—linked list implementation. The column s is the memory size, assigned to the first queue in the case of consecutive implementation. The column, corresponding to the paged implementation, contains the optimal page size x and the average time of random walk. Obviously that at the page size $x = 2$ the average time of random walk for the page implementation is the same as one for the linked list implementation. The column A_2 contains the upper estimate

Table 1. The average time for a consecutive implementation of the queues, when the memory is divided equally.

p_1	q_1	p_2	q_2	r	s	T
0.5	0	0.5	0	0	5	9.29
0.2	0.2	0.2	0.2	0.2	5	61.34
0.6	0.1	0.1	0.1	0.1	5	11.59
0.5	0.1	0.3	0.1	0	5	13.1
0.4	0.1	0.4	0.1	0	5	13.8
0.4	0.19	0.4	0.01	0	5	13.28
0.3	0.2	0.2	0.3	0	5	38.77
0.01	0.5	0.19	0.3	0	5	304.85
0.19	0.5	0.01	0.3	0	5	1703.45
0.3	0.3	0.2	0.2	0	5	48.93
0.4	0.4	0.1	0.1	0	5	47.52
0.45	0.45	0.05	0.05	0	5	45.64
0.3	0.3	0.25	0.1	0.05	5	29.16
0.3	0.3	0.1	0.25	0.05	5	68.92
0.45	0.05	0.1	0.4	0	5	14.17

Table 2. The average time for different methods of implementation of queues.

p_1	q_1	p_2	q_2	r	A_1		A_2			A_3
					s	T	x	$T_{\max}$	$T_{\min}$	T
0.5	0	0.5	0	0	5	9.29	3	9.0	6.0	7.0
2	0.2	0.2	0.2	0.2	5	61.34	3	73.33	35.76	46.81
0.6	0.1	0.1	0.1	0.1	8	16.31	3	15.57	10.12	11.92
0.5	0.1	0.3	0.1	0	6	13.81	3	13.84	8.93	10.56
0.4	0.1	0.4	0.1	0	5	13.8	3	13.91	8.97	10.61
0.4	0.19	0.4	0.01	0	4	13.44	3	13.65	8.8	10.41
0.3	0.1	0.2	0.3	0.1	6	42.87	3	32.84	25.998	23.94
0.01	0.5	0.19	0.3	0	1	1590.3	3	1353.63	292.27	497.81
0.19	0.5	0.01	0.3	0	8	23720.14	3	28498.54	1541.23	4091.91
0.3	0.3	0.2	0.2	0	5	48.93	3	59.13	28.83	37.74
0.4	0.4	0.1	0.1	0	7	55.16	3	63.61	30.96	40.57
0.45	0.45	0.05	0.05	0	7	65.5	3	69.51	33.72	44.25
0.3	0.3	0.25	0.1	0.05	4	29.2	3	32.38	18.8	23.15
0.3	0.3	0.1	0.25	0.05	7	97.46	3	113.88	49.31	67.5
0.45	0.05	0.1	0.4	0	8	21.46	3	20.86	13.45	15.91

$T_{\max}$ and the lower estimate $T_{\min}$ for the average time of random walk.

7. Conclusions

It is possible to draw a conclusion from numerical results, that for linked implementation of the queues, when size of the page $x = 2$, the average time of random walk before overflowing of the memory always less than average time of random walk for consecutive implementation. Also, it is seen that average time for linked list implementation lies between the estimates $T_{\min}$ and $T_{\max}$ of the average time for paged implementation. So for some probabilistic characteristics of the queues the linked list implementation can be better than the page implementation.

Comparing the average time of the random walk for consecutive and paged implementations, notice that the average time for consecutive implementation is basically less than the upper estimate $T_{\max}$ for paged implementation. However, for some probabilistic characteristics of the queues the average time for consecutive implementation is greater than the upper estimate $T_{\max}$ for paged implementation. Also, for practice it can be interesting to analyze the consecutive implementation of the queues, when the memory isn't divided optimum depending on probabilistic characteristics of the queues, but simply—fifty-fifty. It is logically, when we don't know probabilistic characteristics of the queues beforehand. Then as a mathematical model we have two-dimensional random walk on the integer lattice in the square $0 \le x_1 < m/2$, $0 \le x_2 < m/2$, but for the linked list implementation we have two-dimensional random walk on the integer lattice in the triangular area $0 \le x_1 + x_2 < m/2 + 1$. As can be seen from comparison of the lines with numbers 3, 8, 9, 15 in the **Table 1** and in the **Table 2**, the consecutive implementation can be worse than linked.

Though greater part of the triangular area lies inside the square and only two states $(m/2, 0)$ and $(0, m/2)$ are absorbing for the square, but nonrecurring for the triangle, for some probabilistic characteristics of the queues it is enough that the average time of the random walk in the triangle became greater than in the square. For example, in line 3 we have a situation, when the insertion in the first queue occurs with high probability $p_1 = 0.6$, but all rest probabilities is alike. In this case the consecutive implementation is worse than linked. Obviously, this occurs because greater part of the random walk paths is absorbed in the point, which lies outside of absorbing borders of the square, but is absorbing state for the triangle.

8. References

[1] D. E. Knuth, "The Art of Computer Programming," Vol. 1, Addison-Wesley, Reading, 2001.

[2] A. V. Sokolov, "About Storage Allocation for Implementing Two Stacks," *Automation of Experiment and Data Processing*, Petrozavodsk, 1980, pp. 65-71 (in Russian).

[3] A. C. Yao, "An Analysis of a Memory Allocation Scheme for Implementation of Stacks," *SIAM Journal on Computing*, Vol. 10, 1981, pp. 398-403. doi:10.1137/0210029

[4] P. Flajolet, "The Evolution of Two Stacks in Bounded Space and Random Walks in a Triangle," *Lecture Notes in Computer Science*, Vol. 233, 1986, pp. 325-340. doi:10.1007/BFb0016257

[5] G. Louchard, R. Schott, M. Tolley and P. Zimmermann, "Random Walks, Heat Equation and Distributed Algorithms," *Journal of Computational and Applied Mathematics*, Vol. 53, No. 2. 1994, pp. 243-274. doi:10.1016/0377-0427(94)90048-5

[6] R. S. Maier, "Colliding Stacks: A Large Deviations Analysis," *Random Structures and Algorithms*, Vol. 2, No. 4, 1991, pp. 379-421. doi:10.1002/rsa.3240020404

[7] E. A. Aksenova, A. A. Lazutina and A. V. Sokolov, "Study of a Non-Markovian Stack Management Model in a Two-Level Memory," *Programming and Computer Software*, Vol. 30, No. 1, 2004, pp. 25-33. doi:10.1023/B:PACS.0000013438.25368.b4

[8] E. A. Aksenova and A. V. Sokolov, "Optimal Management of Two Parallel Stacks in Two-Level Memory," *Discrete Mathematics and Applications*, Vol. 17, No. 1, 2007, pp. 47-56. doi:10.1515/DMA.2007.006

[9] E. A. Aksenova, A. V. Sokolov and A. V. Tarasuk, "About Optimal Allocation of Two FIFO-Queues in the Bounded Area of Memory," *Control Systems and Information Technologies*, Vol. 3, No. 22, 2006, pp. 62-68 (in Russian).

[10] J. G. Kemeny and J. L. Snell, "Finite Markov Chains, Van Nostrand," Princeton, New Jersey, 1960.