An Algorithm for Locating Adjacent Storage Blocks in the Buddy System

James A. Hinds
State University of New York at Buffalo

A simple scheme for the determination of the location of a block of storage relative to other blocks is described. This scheme is applicable to the buddy type storage allocation systems.

Key Words and Phrases: dynamic storage allocation, buddy system, generalized Fibonacci sequences
CR Categories: 3.89, 4.32, 4.39

Two storage allocation schemes that have been presented by Knowlton [1] and Hirschberg [2] are similar in many basic features, and both are referred to as buddy systems. The purpose of both these methods is to keep track of a large memory pool used to satisfy storage requests.

These two systems have the following basic actions in common:

Step A. To satisfy a storage request:

1. The smallest block of storage that is at least as big as the request is selected as the candidate block.
2. The candidate is checked for size and, if large enough, is split into two smaller blocks (buddies); otherwise the candidate block is returned as the block satisfying the request and the algorithm terminates.
3. One of the buddies (the smaller of the two, if possible) is selected as the new candidate and the other is inserted into the free storage pool. The algorithm then proceeds from Step A. 2.

Step B. To return a block to the storage pool:

1. The buddy of the newly returned block is located.
2. The buddy is inspected to see that it is whole (not split into subbuddies) and free (not allocated). If both conditions are met, then the buddy is removed from free storage and merged with the newly returned block to create a larger block. This larger block is then taken as the newly returned block and execution proceeds with Step B.1.
3. If it is impossible to merge, then the newly returned block is returned to the free storage area and the algorithm terminates.

In both of these schemes it is important to notice that the merges exactly correspond to the splits. For example, if a block at storage location 30 of length 10 is split into buddies starting at 30 and 36 with lengths 6 and 4 respectively, the corresponding merge must be done using exactly these two blocks. In this case it is impossible to merge the block at 30 with length 6 with any other block except that block starting at location 36 and only when that block has a length of 4. This means that sufficient information must be carried in each block to indicate the relative location of the proper buddy.

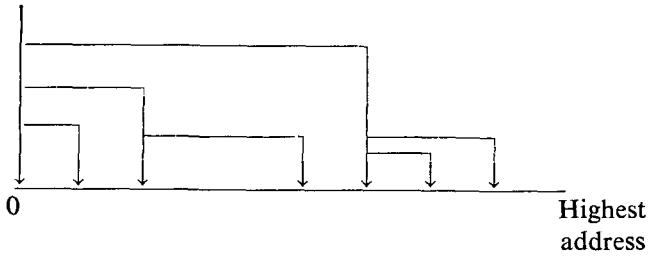
In Knowlton's original buddy system the blocks had lengths of powers of two, and so the addresses of the blocks were an encoding of the relative locations of the blocks. Hirschberg's system used blocks whose length were elements of the familiar Fibonacci series. To locate the buddy of a block, an auxiliary list was consulted that gave the possible starting locations of blocks. Both of the techniques used are special purpose and break down for other sequences of block sizes.

Knowlton allowed block sizes $\text{SIZE}_n = \text{SIZE}_{n-1} + \text{SIZE}_{n-1}$; $\text{SIZE}_0 = 1$. Hirschberg allowed blocks of sizes $\text{SIZE}_n = \text{SIZE}_{n-1} + \text{SIZE}_{n-2}$; $\text{SIZE}_0 = 3$, $\text{SIZE}_1 = 5$. This paper describes a very simple and efficient technique for locating buddies that allows sizes to be elements of an arbitrary (but fixed) generalized Fibonacci sequence. These are sequences of the form $\text{SIZE}_n = \text{SIZE}_{n-1} + \text{SIZE}_{n-k}$ (the series is completely determined by picking k and the first k values of SIZE). The binary buddy system of Knowlton has been shown to be a practical system for storage management. The Fibonacci scheme of Hirschberg has been analyzed statistically and has been shown

Author's address: State University of New York at Buffalo, Department of Computer Science, 4226 Ridge Lea Road, Amherst, NY 14226.

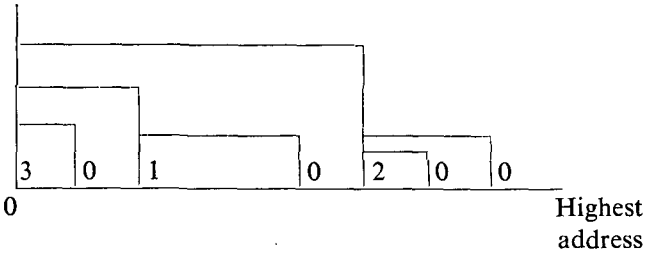
to be superior to the original binary system. The benefits to be obtained from the use of extended Fibonacci series have not been proven, but preliminary investigations have been favorable.

Suppose that an arbitrary Fibonacci sequence has been picked and is to be used on a storage pool. At some point in time the pool may be broken up as shown:



Each vertical line indicates the start of a block, each horizontal line indicates a split and also points to the right-hand buddy. For simplicity the free status of a block has been omitted from the diagram. This diagram indicates exactly what merges would have to take place and what blocks are to be inspected to achieve the merges. It seems obvious that if sufficient information could be encoded into the blocks at the time of a split then the merging process would be made easy.

The information to be coded into each block is this: the count of how many horizontal lines branch off from each vertical line. This is equivalent to the number of splits a block has undergone since it was created (note that right-hand buddies have a count of zero and left-hand buddies have strictly positive counts). The resulting storage block would appear as follows:



Let us call this count the Left-Buddy-Count or LBC. The LBC is utilized with several other attributes of a block. The other attributes of a block are: FREE, a boolean describing the availability of a block, and SIZE, which is an index into a VECTOR containing the actual size of the block (this vector contains the elements of the generating Fibonacci series). In addition, the constant k (the value of k in the generating relation $SIZE_n = SIZE_{n-1} + SIZE_{n-k}$) is used to locate buddies and assign proper sizes during a split. A final assumption is that when splitting a block of SIZE = L, the block of SIZE = L − 1 is the left.

The LBC determines the relative location of a buddy (if LBC = 0 then the buddy is on the left). The SIZE will indicate the starting address of a buddy on the right (address of block + $VECTOR_{SIZE}$ = address of buddy), on the other hand, if the buddy is on the left, then its length must be $VECTOR_{SIZE+k-1}$, hence its location is known. The FREE bit of the buddy will prevent a merge if the buddy is allocated and the SIZE field of the buddy will agree with the predicted size of the buddy only if it has not been split up into smaller blocks, (if it has, then a merge is not possible).

To assign the proper LBC to a block, let the LBC of the entire storage pool be 0 at the beginning. During any split of a parent block, assign to the newly created blocks (here called right and left):

$LBC_{right} := 0$, for the block on the right, and
$LBC_{left} := LBC_{parent} + 1$, for the block on the left.

For a merge the reverse relation is used:

$LBC_{parent} := LBC_{left} - 1$.

Earlier it was claimed that the LBC concept was a simple and efficient method to locate a buddy of a block. The determination of the right or left handedness of a block is simply a test for an LBC = 0. The address of the buddy is calculated by adding or subtracting the proper element from a vector to the starting address of the block. The generation of the LBC involves only a count up or down by one at any single step. This is very comparable to the original efficiency of Knowlton's bit inversion process for calculating the address of a buddy. In addition the LBC is vastly more adaptable to different storage sizes than the systems of Knowlton or Hirschberg.

The details of the management of supporting and auxiliary features are described more fully in algorithms contained in line [3].

Acknowledgment. I wish to thank David Gomberg for his assistance in the preparation of this article.

Received March 1974; revised August 1974

References

1. Knowlton, K.C. A fast storage allocator. *Comm. ACM 8*, 10 (Oct. 1965), 623–625.
2. Hirschberg, Daniel S. A class of dynamic memory allocation algorithms. *Comm. ACM 16*, 10 (Oct. 1973), 615–618.
3. Hinds, James A. A design for the buddy system with arbitrary sequences of block sizes. Tech. Rep. 74, State U. of New York at Buffalo, Mar. 27, 1974.