Memory Allocation : Garbage Collection

① The maintenance of linked lists in memory assumes the possibility of inserting new nodes into the lists and hence requires some mechanism which provides unused memory space for the new nodes.

② Similarly, some mechanism is required whereby the memory space of deleted nodes becomes available for future use.

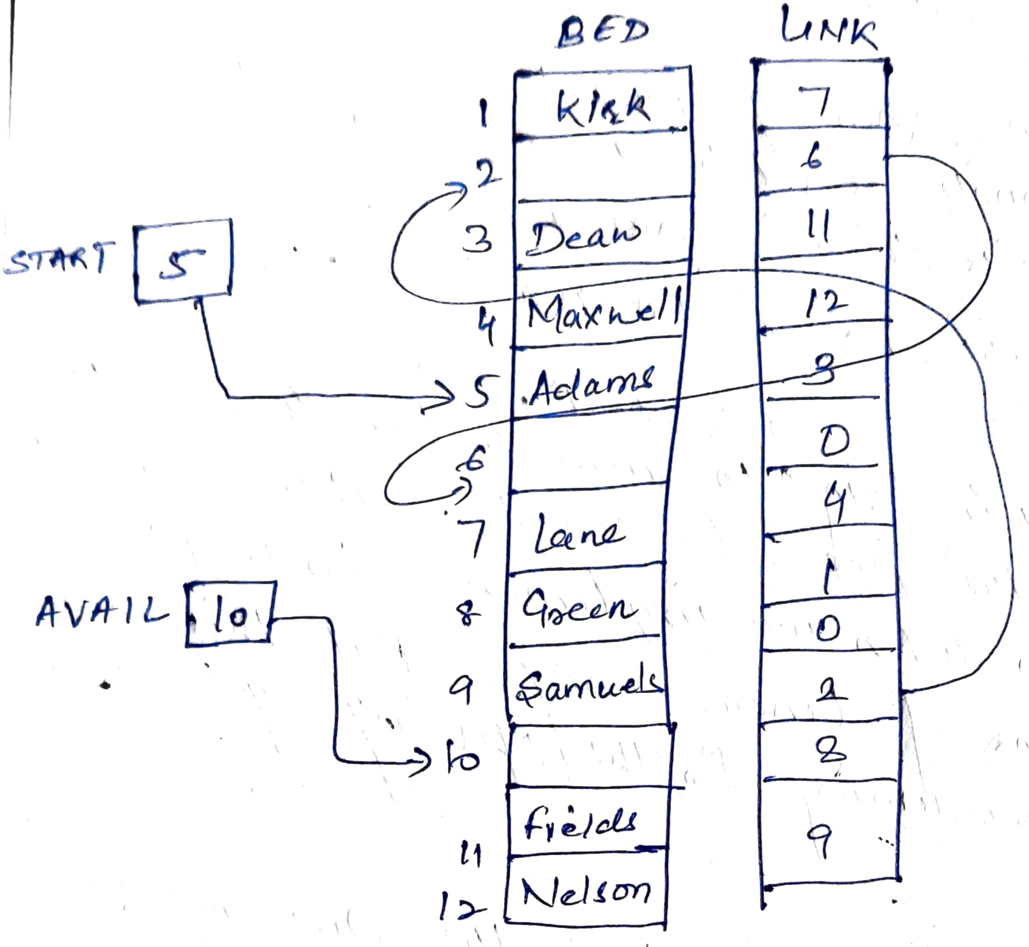
③ Together with the linked lists in memory, a special list is maintained which consists of unused memory cells. This list, which has its own pointer, is called the list of available space or the free storage list or the free pool.

Suppose our linked lists are implemented by parallel arrays and suppose insertions and deletions are to be performed on our linked lists. Then unused memory cells in the arrays will also be linked together to form a linked list using AVAIL as its pointer variable. Such a data structure will frequently be denoted by writing

LIST (INFO, LINK, START, AVAIL)

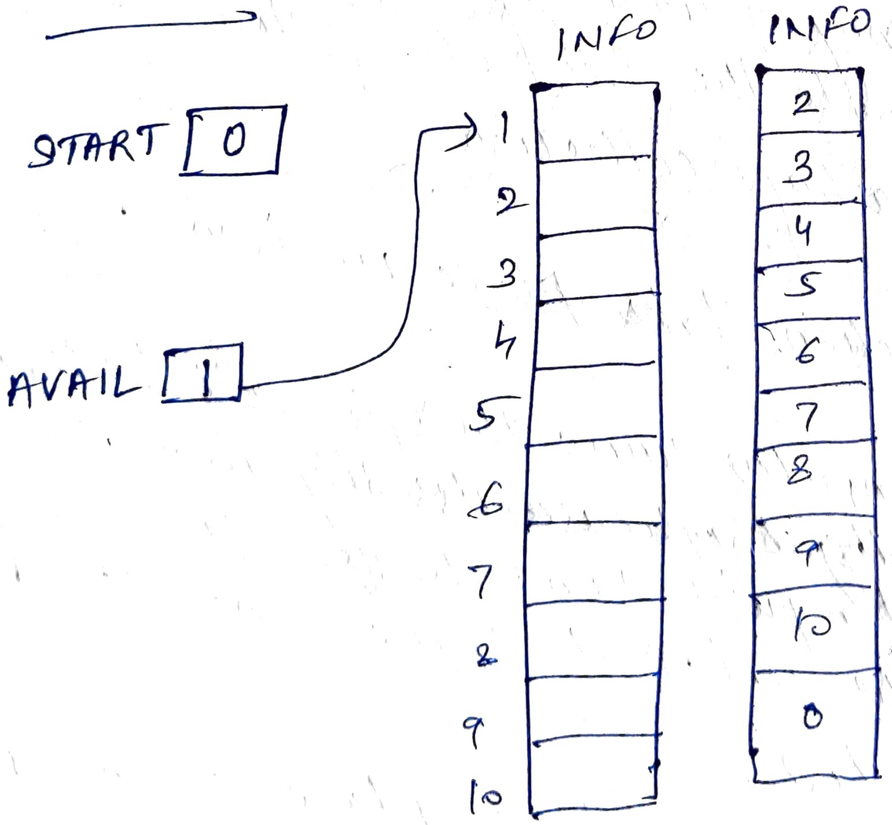
Example

Suppose the list of patients is stored in the linear arrays BED and LINK (so that the patient in bed K is assigned to BED[K]). Then the available space in the linear arrays BED may be linked as in figure. observe that BED[10] is the first available bed, BED[2] is the next available bed, and bed., Hence BED[6] BED[6] is the last available bed. Hence BED[6] has the null pointer in the next pointer field. that is LINK[6]=0.



	BED	LINK
1	Kirk	7
2		6
3	Dean	11
4	Maxwell	12
5	Adams	3
6		0
7	Lane	4
8	Green	1
9	Samuels	0
10		2
11	Fields	8
12	Nelson	9

Example 2



	INFO	INFO
1		2
2		3
3		4
4		5
5		6
6		7
7		8
8		9
9		10
10		0

GARBAGE COLLECTION

① Suppose some memory space becomes reusable because a node is deleted from a list or an entire list is deleted from a program.

② We want the space to be available for future use.

③ One way to bring this about is to immediately reinsert the space into the free storage list. This is what we will do when we implement linked list by means of linear arrays. But this method is time consuming for the operating system of a computer.

④ Another alternative is that the Operating System of a computer may periodically collect all the deleted space onto the free storage list. This is ~~also~~ called as Garbage collection.

Garbage collection take place in two steps.

Ⓐ First the computer runs through all lists, tagging those cells which are currently in use; and then the computer runs through the memory, collecting all the untagged space onto the free-storage list.

The garbage collection may take place when there is only some minimum amount of space

or no space at all left in the free storage (7)
list, or when the CPU is idle and has no time
to do the collection.

Overflow and Underflow

(1) Sometime new data are to be inserted into
a data structure but there is no available space
ie the free storage list is empty. This
situation is usually called overflow.
In such a case, the programmer may then modify
the program by adding space to the underlying
arrays. Observe that overflow will occur
with our linked lists when AVAIL = NULL
and there is an insertion.

(2) The term underflow refers to the situation
where one wants to delete the data from a
data structure that is empty. The underflow
will occur with our linked lists when START =
NULL and there is a deletion.

Garbage compaction

① There happens sometime when we actually run out of memory. most of the memory ~~[illegible]~~ management methods can fail as a result of it.

② This situation can occur either when there is too much of fragmentation or when the entire memory is allocated.

Compaction solve this problem of fragmentation easily by combining all the holes (free space) by moving the allocated blocks to one end of memory.

Compaction has one important limitation-

① it involves updating of pointer. All the pointer ~~to~~ which points to the block need to be updated whenever a block is moved. This is in addition to the cost of copying.

② The main challenge is to find all such pointers. only after which compaction is possible.

In certain cases there are a few ~~possibilities~~ like the pointer pointing to the start of the block and also into its body.

In the case of an executable code, a branch instruction can also point to different locations

Viz only ov...

in the same block. In such cases the compaction(s) can be done in three phases.

(a) In the first phases we determine the distance to which the block will be moved by calculating the new location of each block.

(b) In the second phase we update all the pointers by adding an amount of block which it is pointing into it will be moved.

(c) In the final phase the actual data is moved.