Memory Allocation Techniques:

These techniques are algorithms that satisfy the memory needs of a process. They decide which hole from the list of free holes must be allocated to the process.

In fixed partitioning with equal size partitions, these algorithms are not applicable, because all the partitions are of the same size.

There are primarily three techniques for memory allocation.

① First-Fit Allocation

① This algorithm searches the list of free holes & allocates the first hole in the list that is big enough to accomodate the desired process.

② searching is stopped when it finds the first-fit hole.

③ The next time, searching is resumed from the first hole.

* Next-Fit Allocation:

① In this method searching will be done as per the first-fit allocation only.

② only the difference is, next time, searching is resumed from the last occupied hole.

It may be possible that first-fit hole is very large, compared to the memory required by the process, resulting in wastage of memory.

② Best-fit Allocation:

① This algorithm takes care of memory storage & searches the list, by comparing memory size of the process to be allocated with that of free holes in the list.

② The smallest hole that is big enough to fulfil the process ~~to be alloced~~ is allocated.

③ It incurs cost of searching all the entries in the list.

④ It also causes internal fragmentation.

③ Worst-fit Allocation:

① It is reverse of the best-fit algorithm.

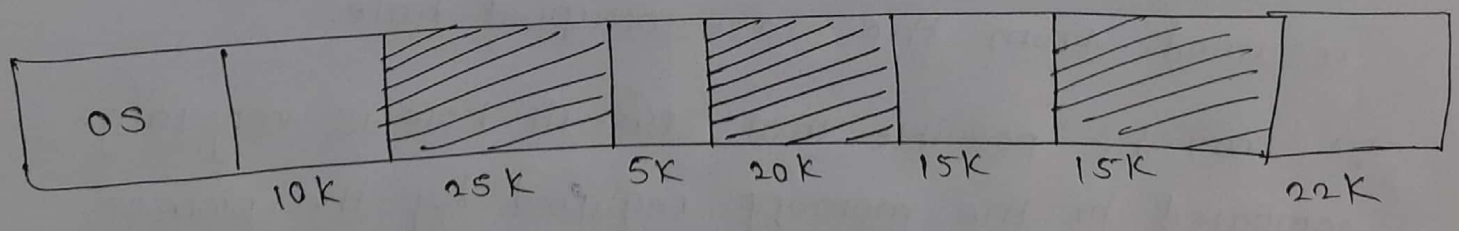
② It searches the list for the largest hole.

③ Leftover space from this large hole may be allocated to the process that fit.

④ It incurs overhead of searching the list for the largest hole.

example:

consider memory allocation scenario as shown below.



□ - hole ▤ occupied by process

Allocate memory for additional request of 4K & 10K. Compare the memory allocation using first fit, best fit, worst fit allocation methods, in terms of internal fragmentation.

⟹ ① First-Fit Allocation:

It allocates the first hole in the list i.e. big enough. Hole of 10K is allocated to the process of 4K, leaving a hole of 6K in memory. The next request is of 10K. So the next hole in the list is of 5K and it cannot accomodate the process. Hence, the next available hole of 15K is allocated, leaving a hole of 5K.

It leaves fragmentation of 6K + 5K = 11K.

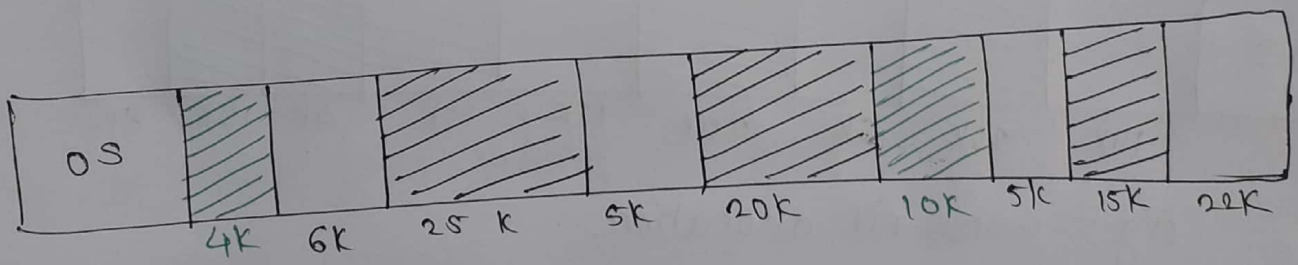


fig: First-Fit Allocation.

② Best-Fit Allocation:

It allocates the smallest hole in the list that is big enough. Hole of 5K is allocated to the process of 4K, leaving a hole of just 1K. The next request is of 10K. After comparing the size of all the holes, holes of 10K is allocated.

Fragmentation 0K + 1K = 1K

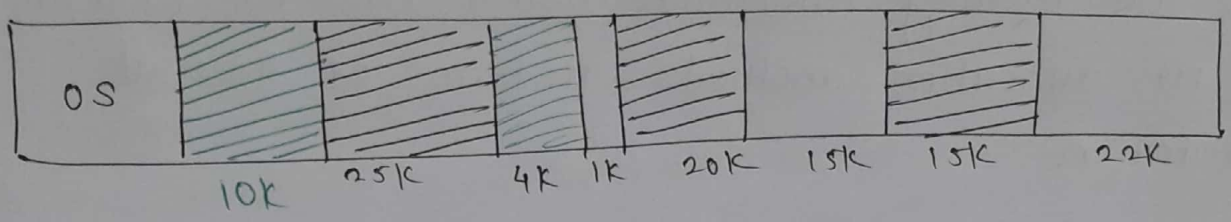


fig: Best-Fit Allocation

③ worst-Fit Allocation:

It allocates the largest hole in the list. For the process of 4K, the largest hole of 22K is allocated. This leaves a hole of 18K in the memory. The next request is of 10K. comparing the size of all holes, 18K hole is largest hole in the list. Hence it is allocated to the process.

Fragmentation = 22K - 4K - 10K = 08K.

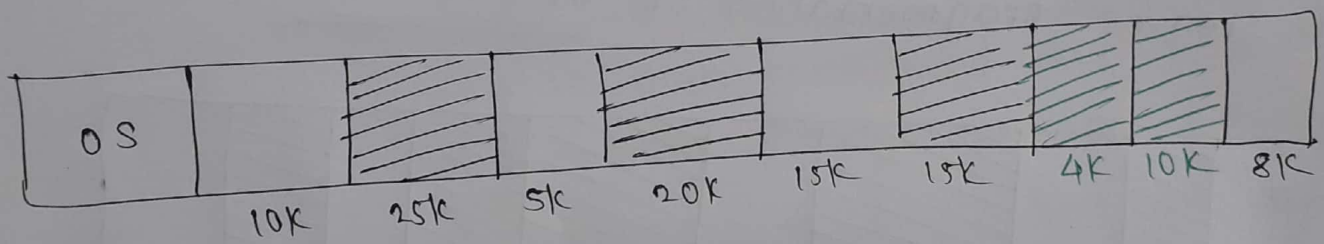


fig: worst-Fit allocation.

By comparing all the algorithms on the basis of internal fragmentation, the best-fit allocation is the best method.