Mitigate Fragmentation via BestFit, WorstFit, FirstFit

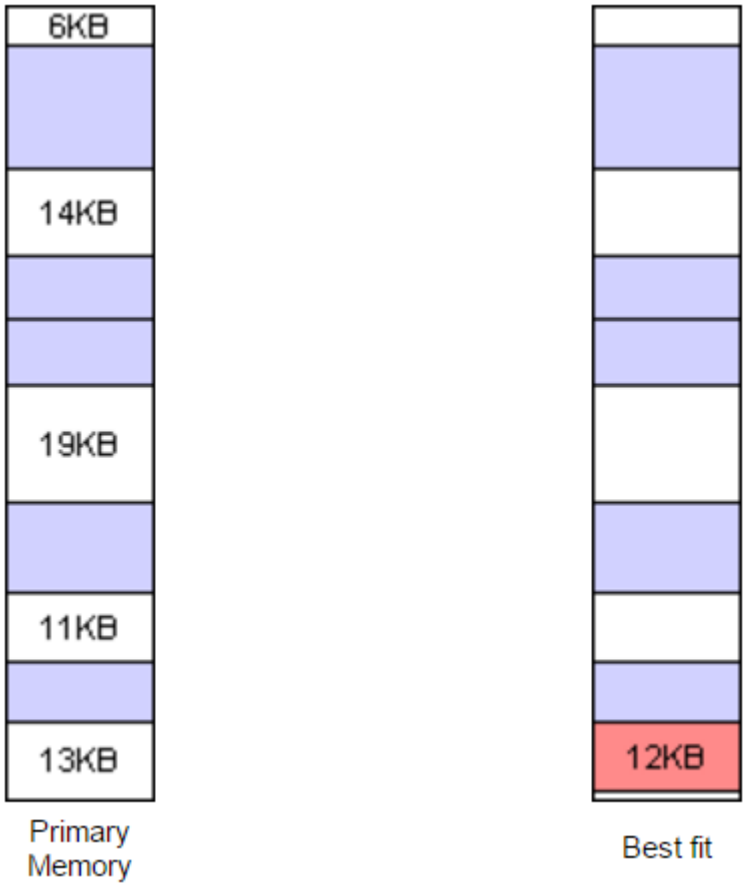
Source:

http://courses.cs.vt.edu/~csonline/OS/Lessons/MemoryAllocation/index.html

Best fit: always place in the smallest hole

The allocator places a process in the smallest block of unallocated memory in which it will fit. For example, suppose a process requests 12KB of memory and the memory manager currently has a list of unallocated blocks of 6KB, 14KB, 19KB, 11KB, and 13KB blocks. The best-fit strategy will allocate 12KB of the 13KB block to the process.

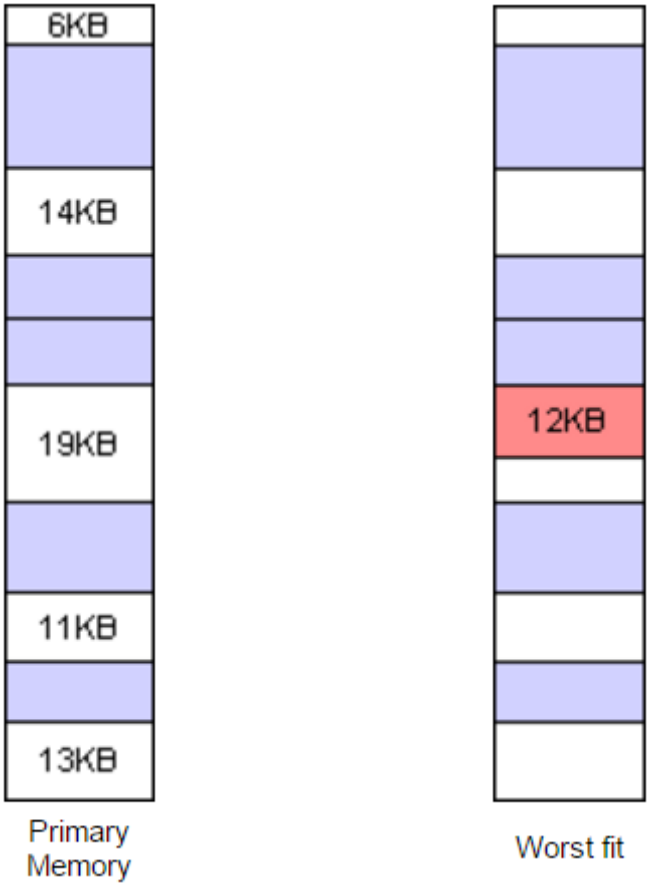
To allocate 12KB:



Worst fit: always place in the largest hole to “leave the largest hole”

The memory manager places a process in the largest block of unallocated memory available. The idea is that this placement will create the largest hold after the allocations, thus increasing the possibility that, compared to best fit, another process can use the remaining space. Using the same example as above, worst fit will allocate 12KB of the 19KB block to the process, leaving a 7KB block for future use.

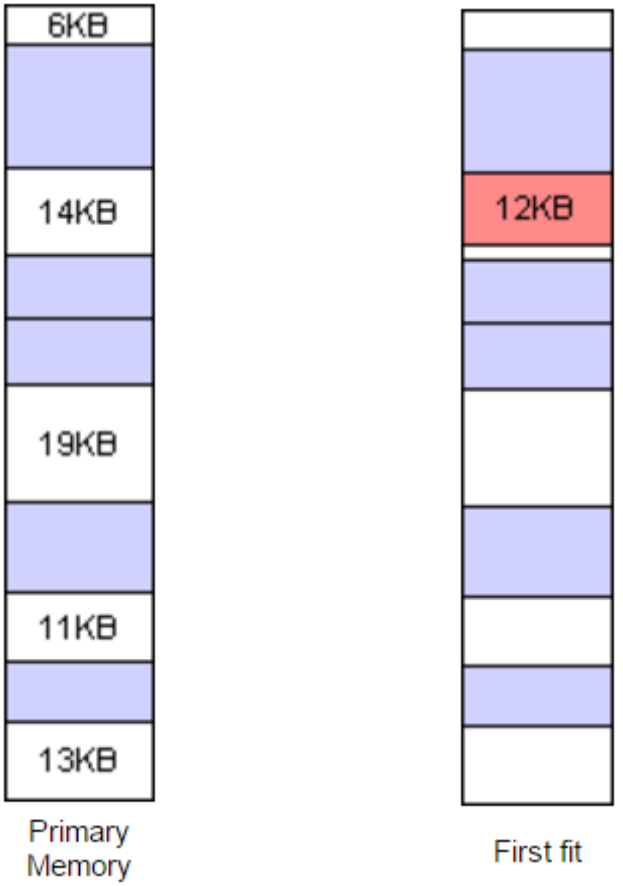
To allocate 12KB:



First fit: always place in the first hole to save analysis time

There may be many holes in the memory, so the operating system, to reduce the amount of time it spends analyzing the available spaces, begins at the start of primary memory and allocates memory from the first hole it encounters large enough to satisfy the request. Using the same example as above, first fit will allocate 12KB of the 14KB block to the process.

To allocate 12KB:



To allocate 12KB: