3.2 Variable Partitioning

- There are three algorithms for searching the list of free blocks for a specific amount of memory.
  - First Fit
  - Best Fit
  - Worst Fit

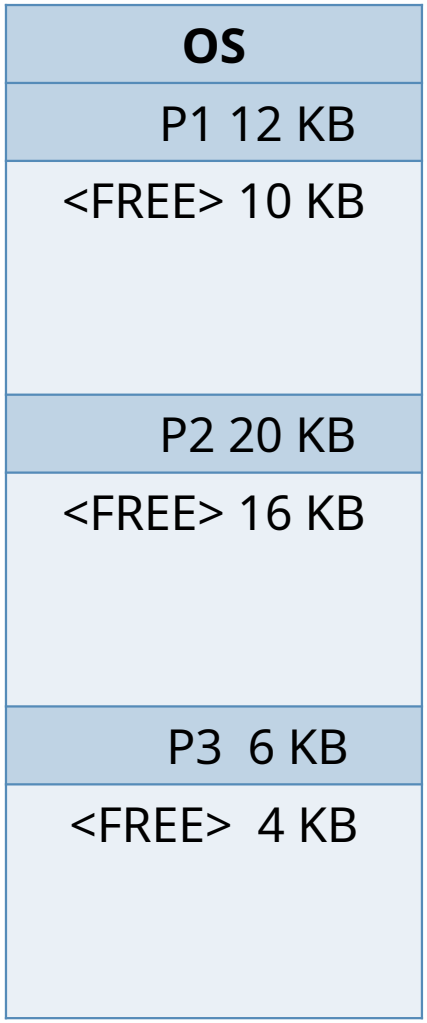
first fit

- First Fit : Allocate the first free block that is large enough for the new process.

- This is a fast algorithm.

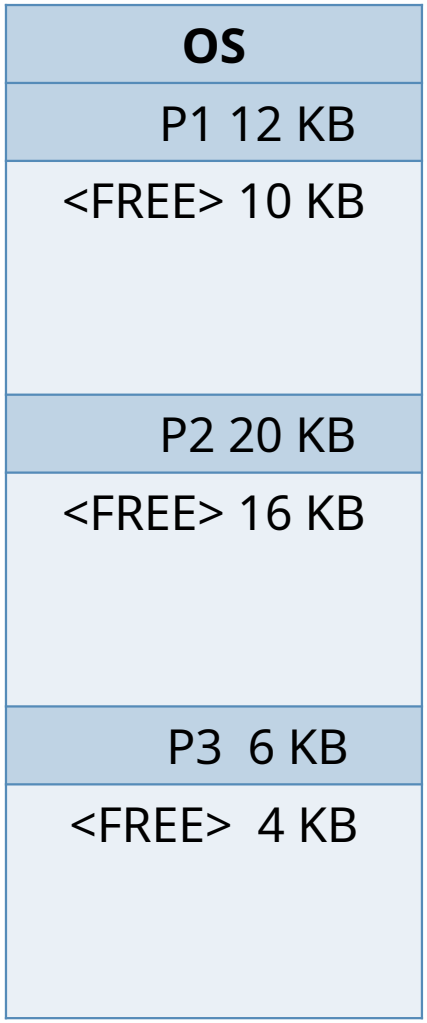
first fit

Initial memory mapping

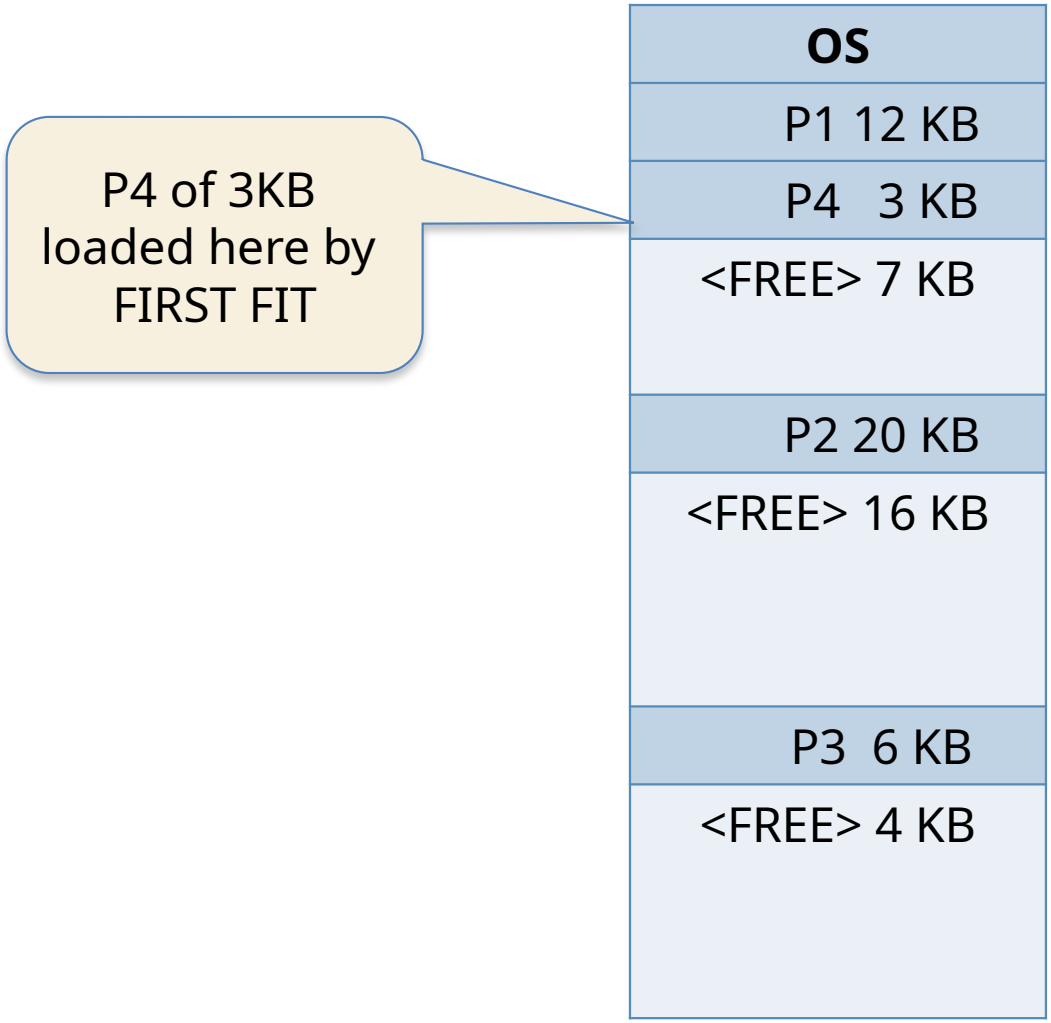


first fit

P4 of 3KB arrives

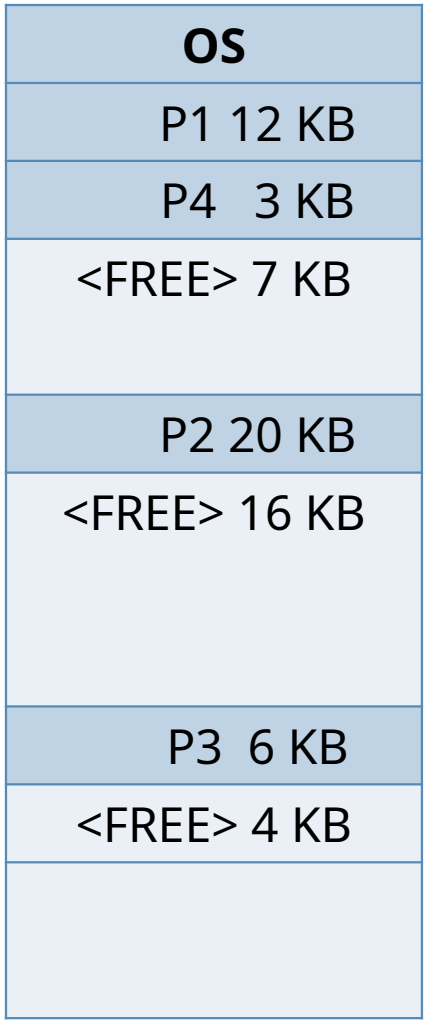


first fit

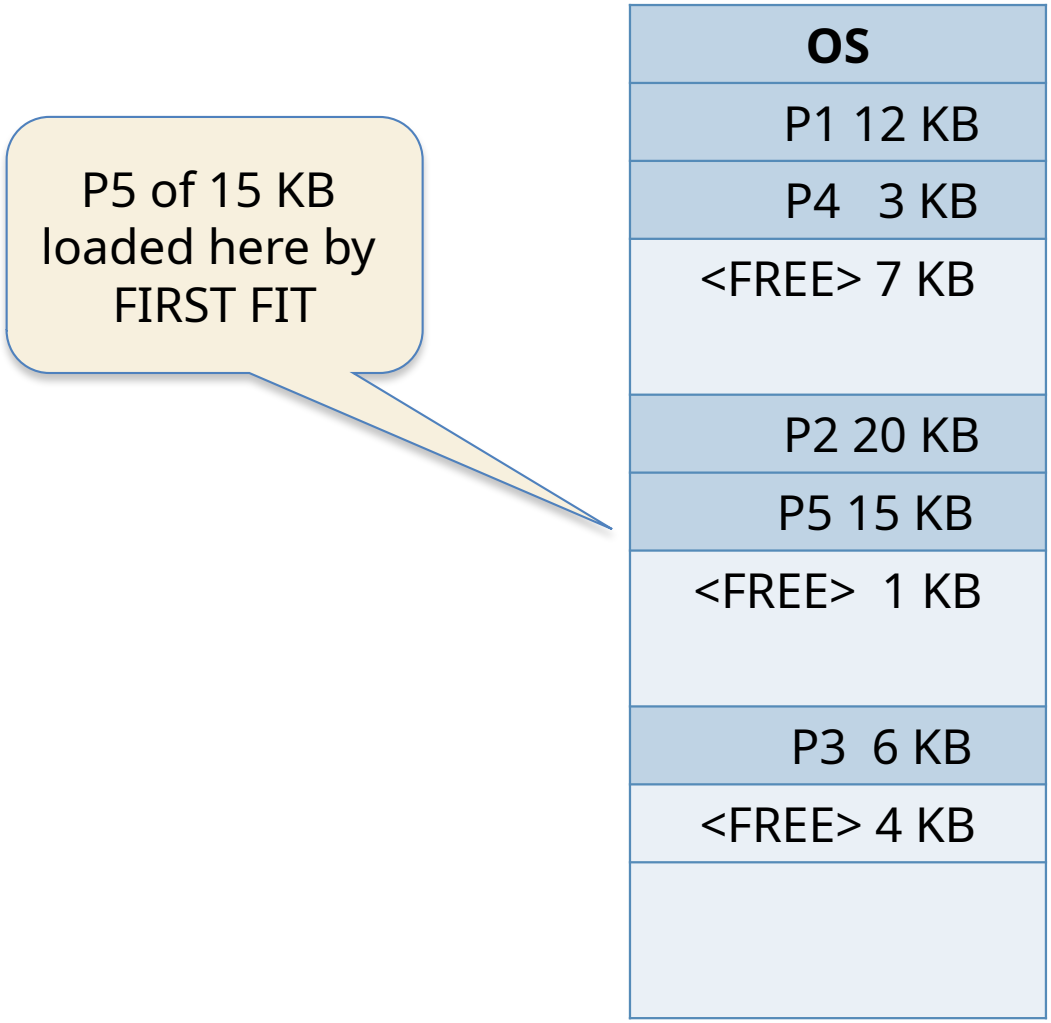

OS
P1 12 KB
P4 3 KB
<FREE> 7 KB
P2 20 KB
<FREE> 16 KB
P3 6 KB
<FREE> 4 KB
P4 of 3KB loaded here by FIRST FIT

first fit

P5 of 15KB arrives



first fit

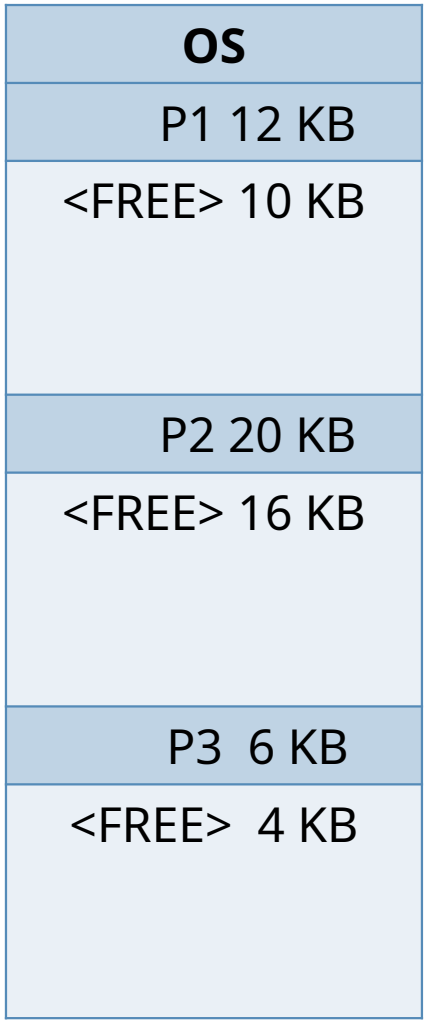

P5 of 15 KB loaded here by FIRST FIT
OS
P1 12 KB
P4 3 KB
<FREE> 7 KB
P2 20 KB
P5 15 KB
<FREE> 1 KB
P3 6 KB
<FREE> 4 KB

Best fit

- Best Fit : Allocate the smallest block among those that are large enough for the new process.
- In this method, the OS has to search the entire list, or it can keep it sorted and stop when it hits an entry which has a size larger than the size of new process.
- This algorithm produces the smallest left over block.
- However, it requires more time for searching all the list or sorting it
- If sorting is used, merging the area released when a process terminates to neighboring free blocks, becomes complicated.

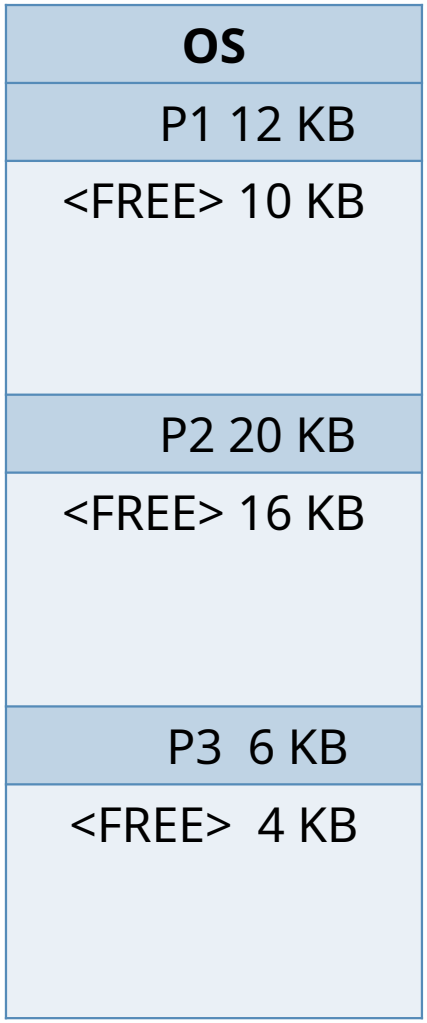
best fit

Initial memory mapping

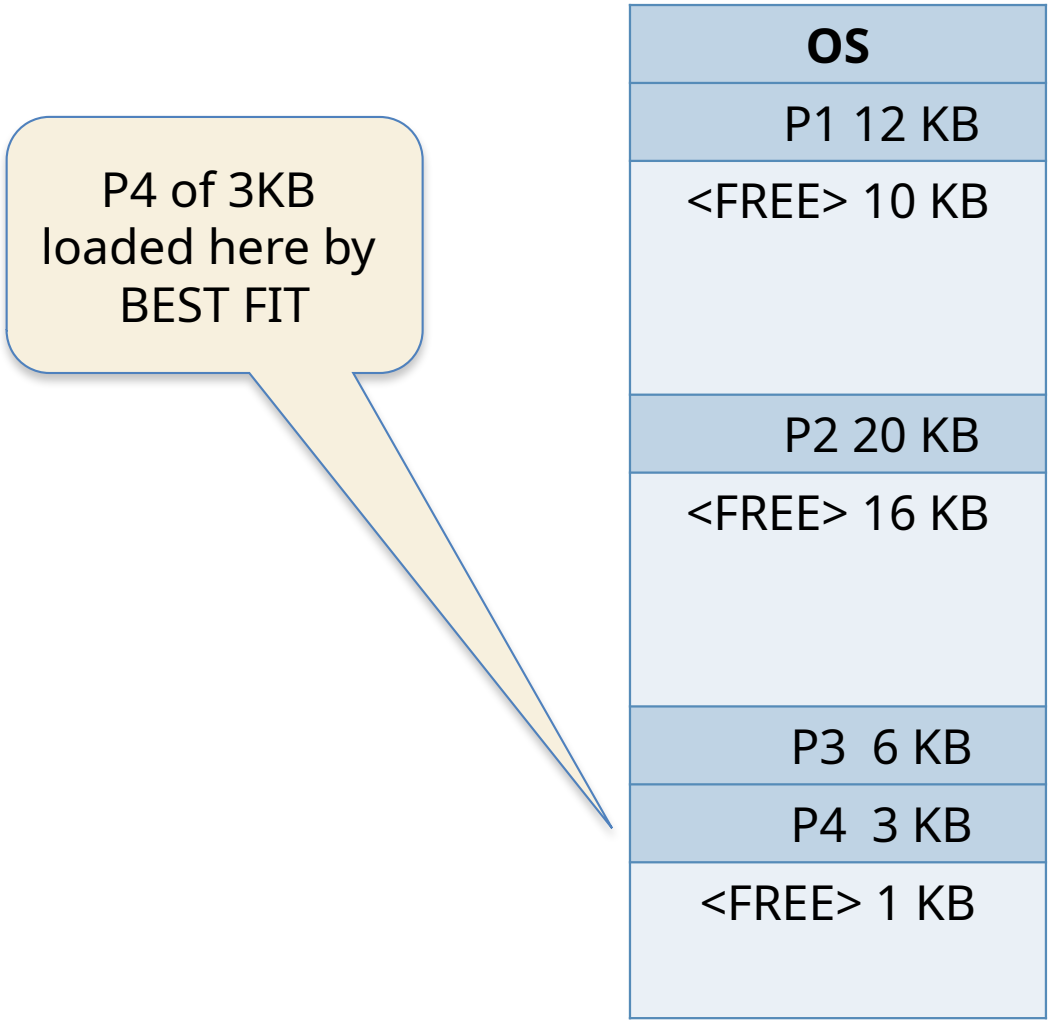


best fit

P4 of 3KB arrives

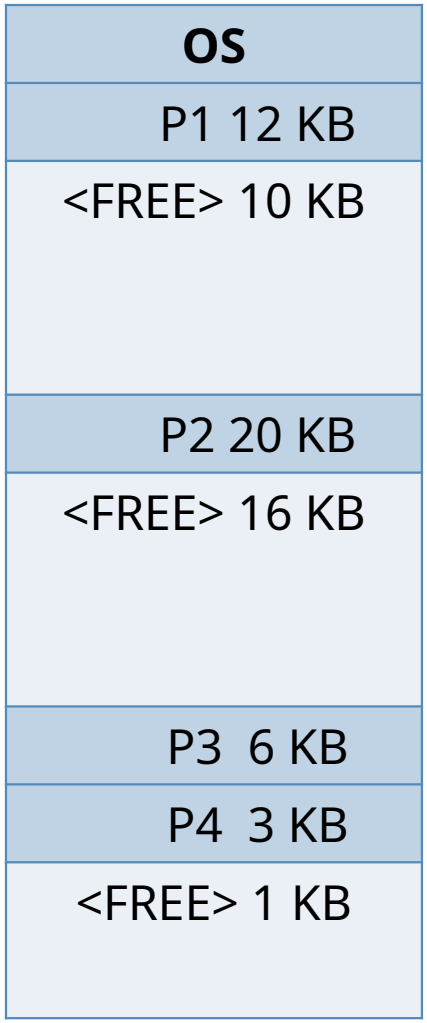


best fit

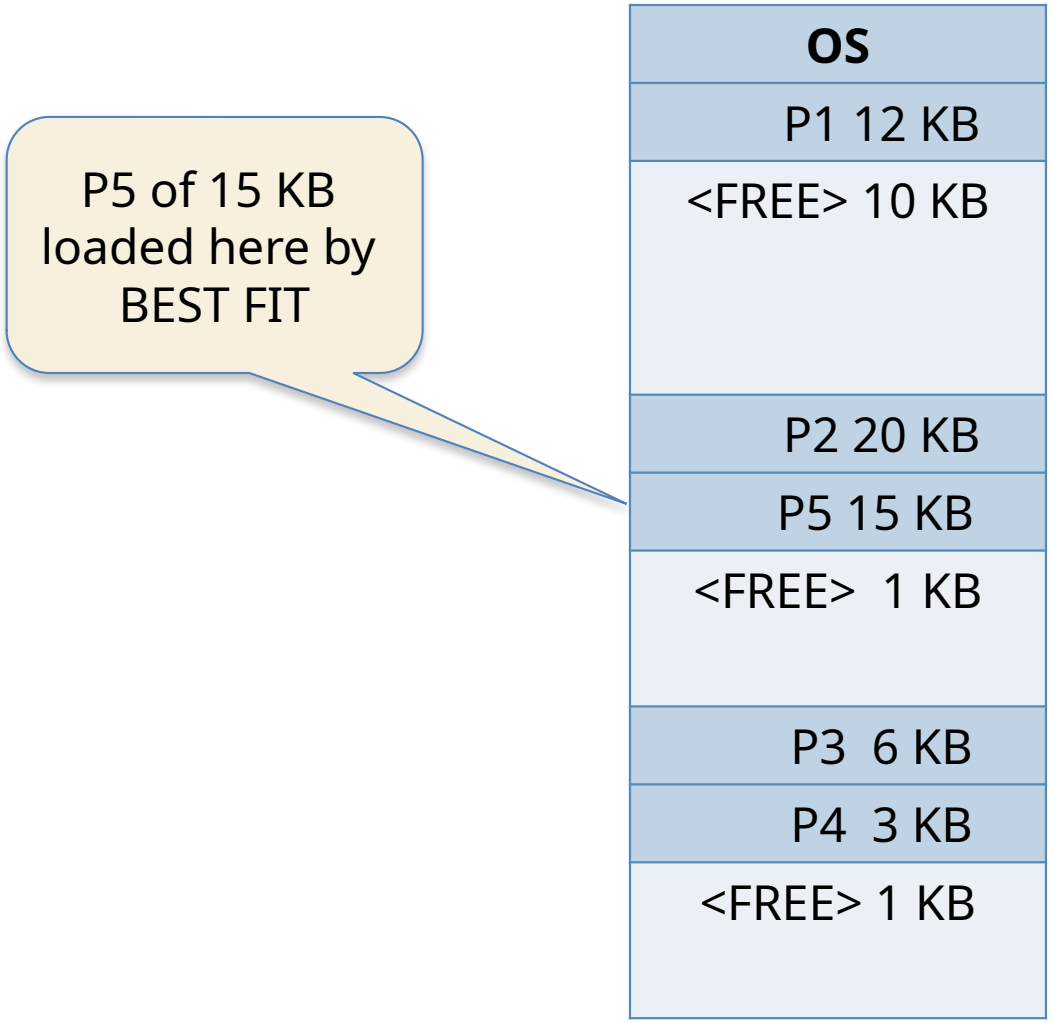

P4 of 3KB loaded here by BEST FIT
OS
P1 12 KB
<FREE> 10 KB
P2 20 KB
<FREE> 16 KB
P3 6 KB
P4 3 KB
<FREE> 1 KB

best fit

P5 of 15KB arrives



best fit

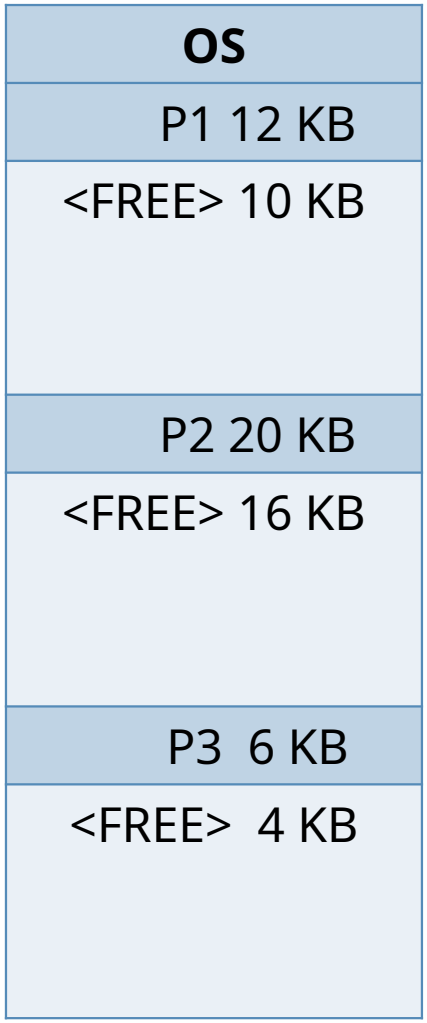

OS
P1 12 KB
<FREE> 10 KB
P2 20 KB
P5 15 KB
<FREE> 1 KB
P3 6 KB
P4 3 KB
<FREE> 1 KB
P5 of 15 KB loaded here by BEST FIT

worst fit

- Worst Fit : Allocate the largest block among those that are large enough for the new process.
- Again a search of the entire list or sorting it is needed.
- This algorithm produces the largest over block.

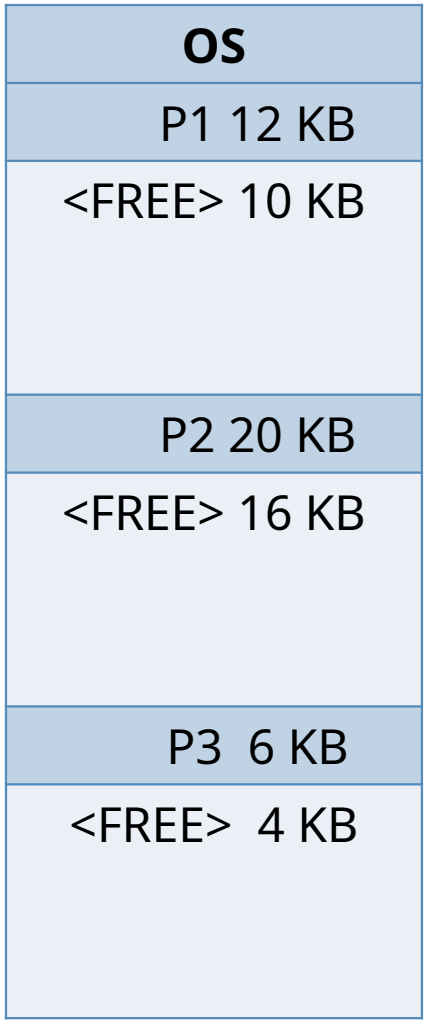
worst fit

Initial memory mapping

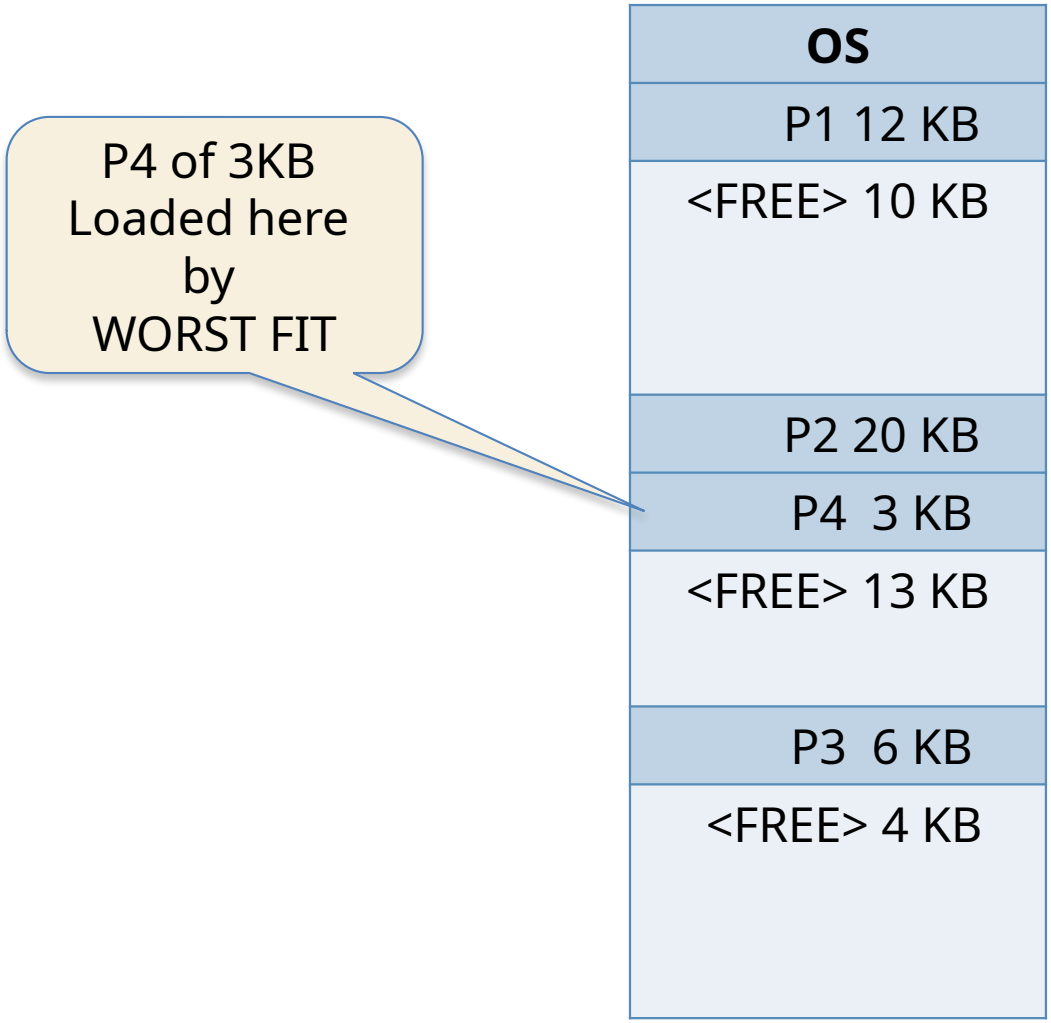


worst fit

P4 of 3KB arrives

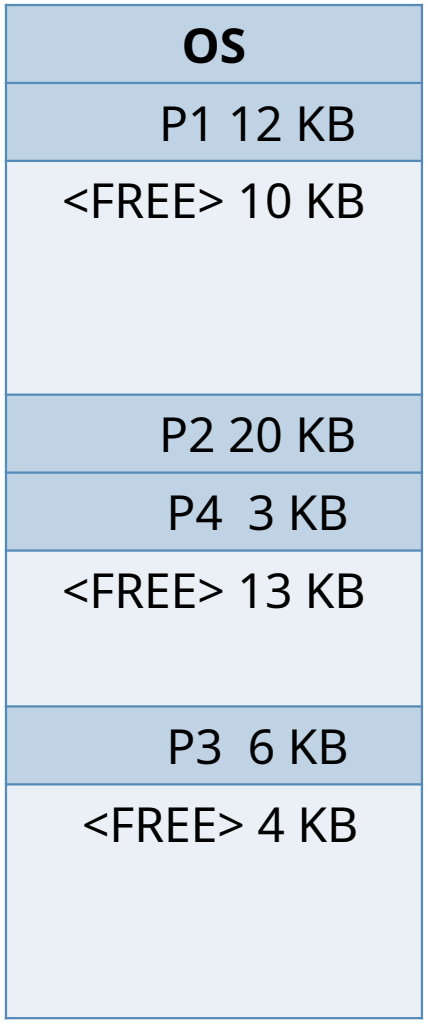


worst fit


P4 of 3KB
Loaded here
by
WORST FIT
OS
P1 12 KB
<FREE> 10 KB
P2 20 KB
P4 3 KB
<FREE> 13 KB
P3 6 KB
<FREE> 4 KB

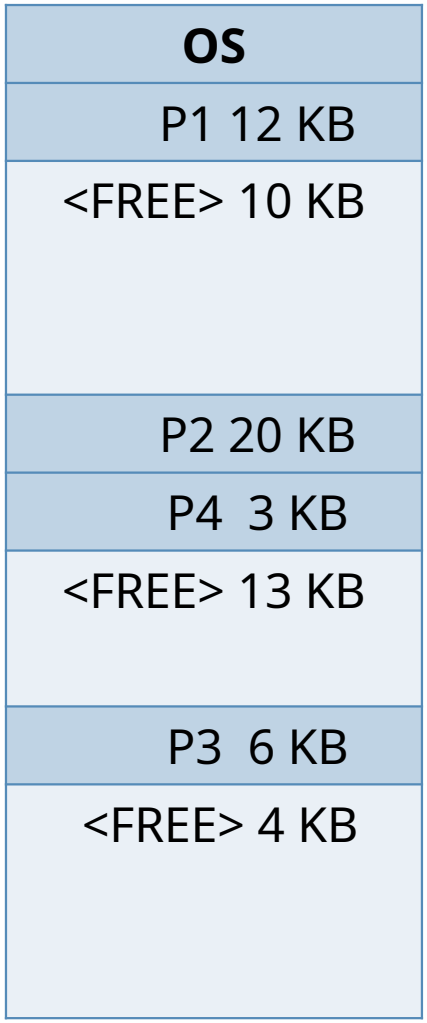
worst fit

No place to load P5 of 15K



worst fit

No place to load P5 of 15K



- Given five memory partitions of 100Kb, 500Kb, 200Kb, 300Kb, 600Kb (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of 212 Kb, 417 Kb, 112 Kb, and 426 Kb (in order)? Which algorithm makes the most efficient use of memory?

- 1. Given five memory partitions of 100Kb, 500Kb, 200Kb, 300Kb, 600Kb (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of 212 Kb, 417 Kb, 112 Kb, and 426 Kb (in order)? Which algorithm makes the most efficient use of memory?

- First-fit:
  - 212K is put in 500K partition
  - 417K is put in 600K partition
  - 112K is put in 288K partition (new partition 288K = 500K - 212K)
  - 426K must wait

- Best-fit:
  - 212K is put in 300K partition
  - 417K is put in 500K partition
  - 112K is put in 200K partition
  - 426K is put in 600K partition

- Worst-fit:
  - 212K is put in 600K partition
  - 417K is put in 500K partition
  - 112K is put in 388K partition
  - 426K must wait

- In this example, best-fit turns out to be the best.

212,417,112,426

100	500	200	300	600		