

Part 2: Short-Answer Questions

1. (Exercise 8.1) Explain the difference between internal and external fragmentation.

Internal Fragmentation

- What it is: Wasted space *inside* an allocated memory block.
- Why it happens: When memory is allocated in fixed-size blocks (e.g., 4 KB), and the process doesn't use the entire block.
- Example: A process needs 6 KB, but memory can only allocate 8 KB blocks. The extra 2 KB is unused — that's **internal fragmentation**.
- Analogy: Like buying a 1 TB iPhone but only using 600 GB — the remaining 400 GB is reserved but unused.

External Fragmentation

- What it is: Wasted space *between* allocated memory blocks.
- Why it happens: When memory is allocated dynamically in varying sizes, small gaps form between blocks as memory is allocated and freed.
- Example: Suppose you have free blocks of 2 KB, 3 KB, and 4 KB scattered around, but a process needs 5 KB. Even though you have 9 KB free in total, there's no single continuous 5 KB block.
- Analogy: Like having scattered free time slots in your calendar — even though you have 2 hours free in total, you can't schedule a 1.5-hour meeting if no slot is big enough.

Nice one!

2. (Exercise 8.3) Given six memory partitions of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)? Rank the algorithms in terms of how efficiently they use memory.

- 2.a) Given the following partitions (300KB, 600KB, 350KB, 200KB, 750KB, 125KB) at the beginning. Consider the first-fit algorithm.

First-Fit Algorithm — What to Consider

1. Definition: First-fit places each process in the first memory partition large enough to accommodate it.
2. You go in **order** through the memory partitions each time, and **don't go back** once a partition is taken or too small.
3. Update the partition size (or mark it as used) as soon as a process is placed there.

[0] 300 KB → 185 KB
[1] 600 KB → 100 KB
[2] 350 KB → 150 KB
[3] 200 KB
[4] 750 KB → 392 KB → 17 KB
[5] 125 KB

P1	115	300 → 185 KB
P2	500	600 → 100 KB
P3	358	750 → 392 KB
P4	200	350 → 150 KB
P5	375	392 → 17 KB

∴ All process can be allocate into the main memory.

- 2.b) Given the following partitions (300KB, 600KB, 350KB, 200KB, 750KB, 125KB) at the beginning. Consider the best-fit algorithm.

Best-Fit Algorithm — What to Consider

1. Definition: Best-Fit allocates the **smallest** partition that is large enough for the process.
2. For each process, you check all available partitions, and choose the one that leaves the **least** leftover space.
3. Just like First-Fit, once a partition is allocated, it's **updated** (size reduced or marked as used).

[0] 300 KB
 [1] 600 KB → 100 KB
 [2] 350 KB
 [3] 200 KB → 0 KB
 [4] 750 KB → 392 KB → 17 KB
 [5] 125 KB → 10 KB

P1 115 125 → 10 KB
 P2 500 600 → 100 KB
 P3 358 750 → 392 KB
 P4 200 200 → 0 KB (exact fit)
 P5 375 392 → 17 KB

∴ All process can be allocate into the main memory.

This one
is a perfectionist
like I am.

- 2.c) Given the following partitions (300KB, 600KB, 350KB, 200KB, 750KB, 125KB) at the beginning. Consider the worst-fit algorithm.

Worst-Fit Algorithm — What to Consider

1. Definition: Worst-Fit places each process in the **largest** available partition that's big enough.
2. The logic is like, "Let's leave more space in smaller holes by using up the big chunks first."
3. For each process, you search for the **biggest** available partition that fits.

[0] 300 KB
 [1] 600 KB → 242 KB
 [2] 350 KB → 150 KB
 [3] 200 KB
 [4] 750 KB → 635 KB → 135 KB
 [5] 125 KB

P1 115 750 → 635 KB
 P2 500 635 → 135 KB
 P3 358 600 → 242 KB
 P4 200 350 → 150 KB
 P5 375 Cannot be placed?

∴ Cannot allocate all processes into the main memory.

THE OPPOSITE OF
PERFECTIONIST, HAHA