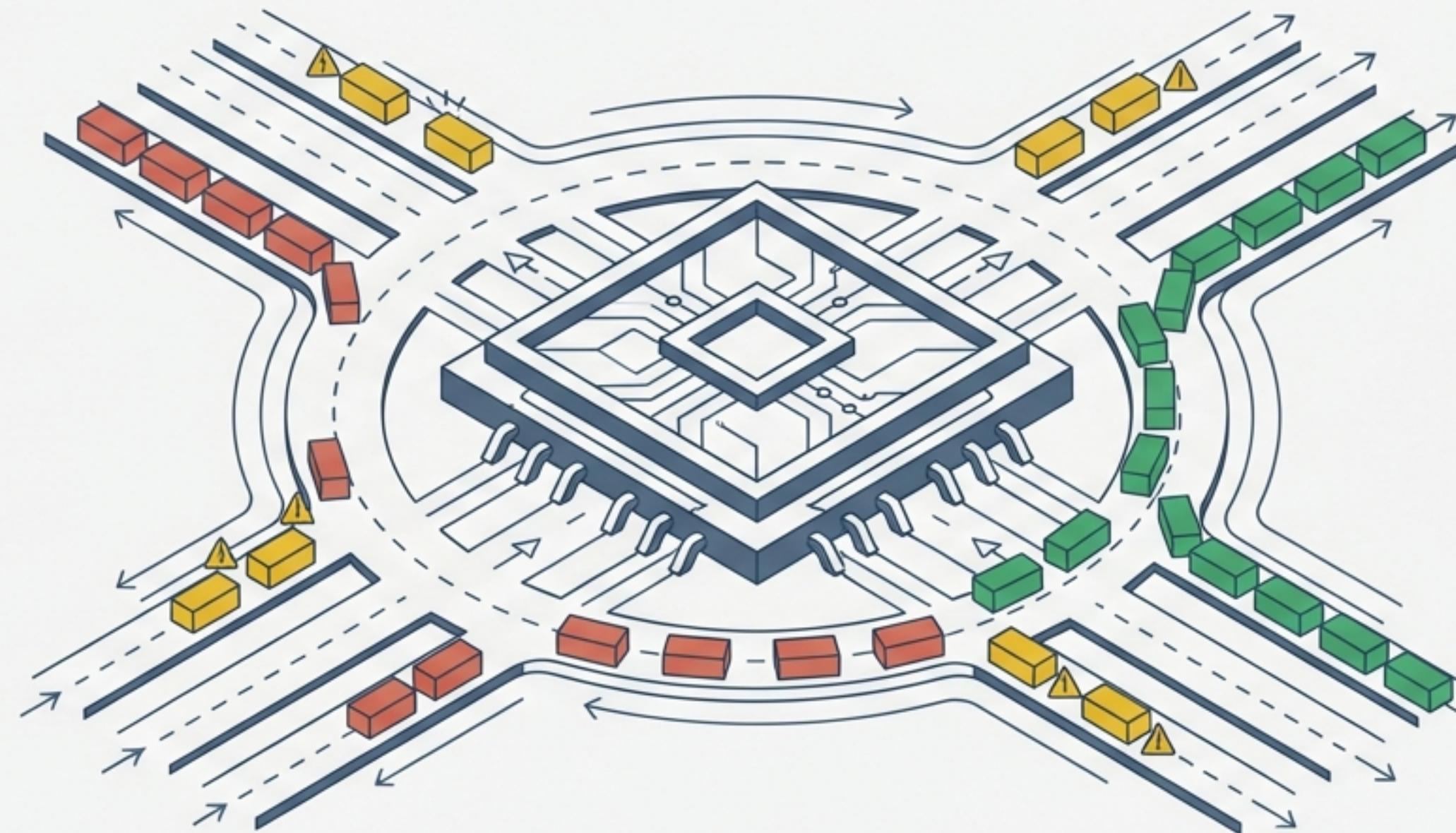


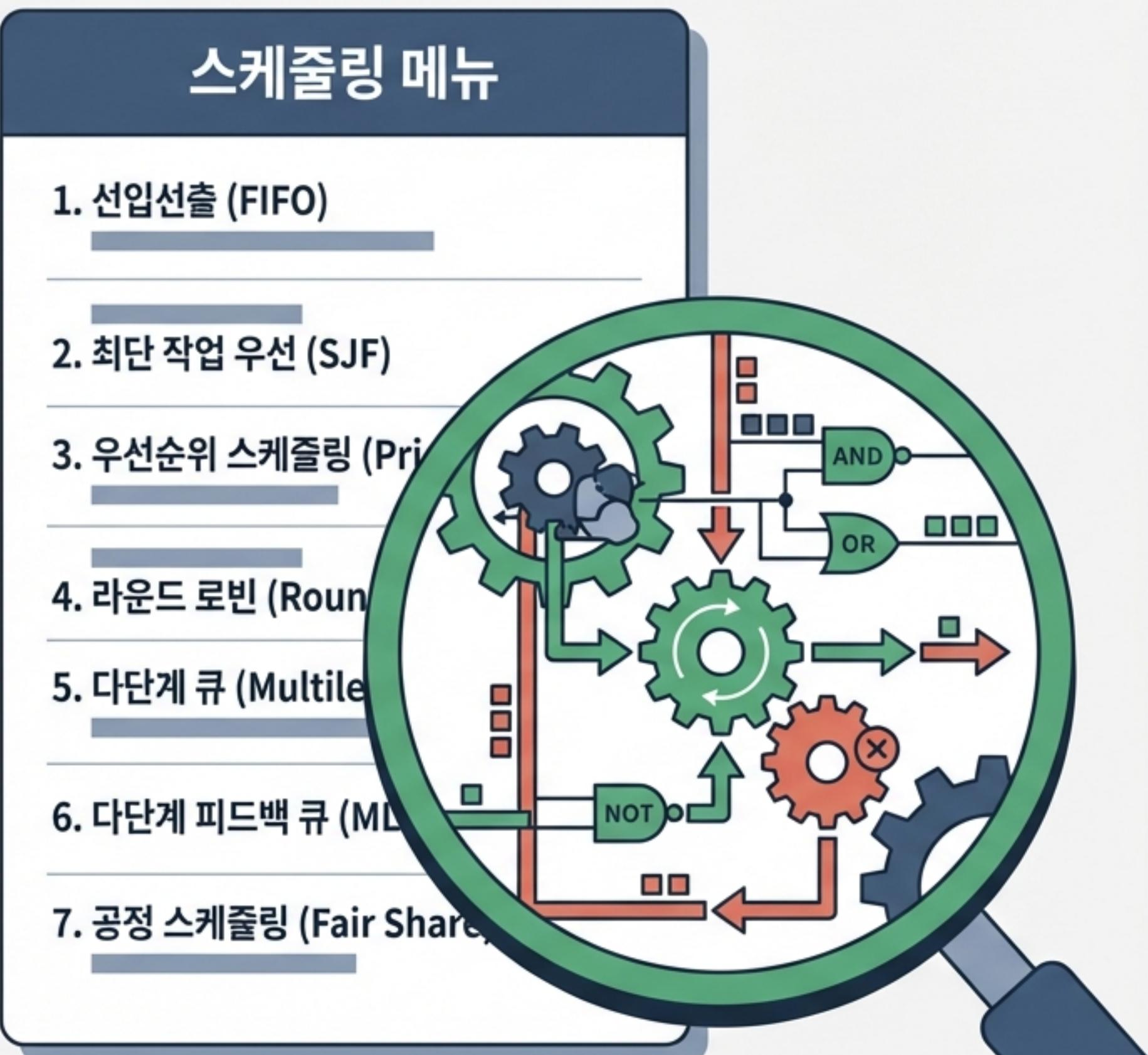
CPU 스케줄링 알고리즘 7선

운영체제는 어떻게 CPU 자원을 가장 효율적으로 배분할까?



암기보다는 '아이디어'와 '트레이드오프'에 집중하세요

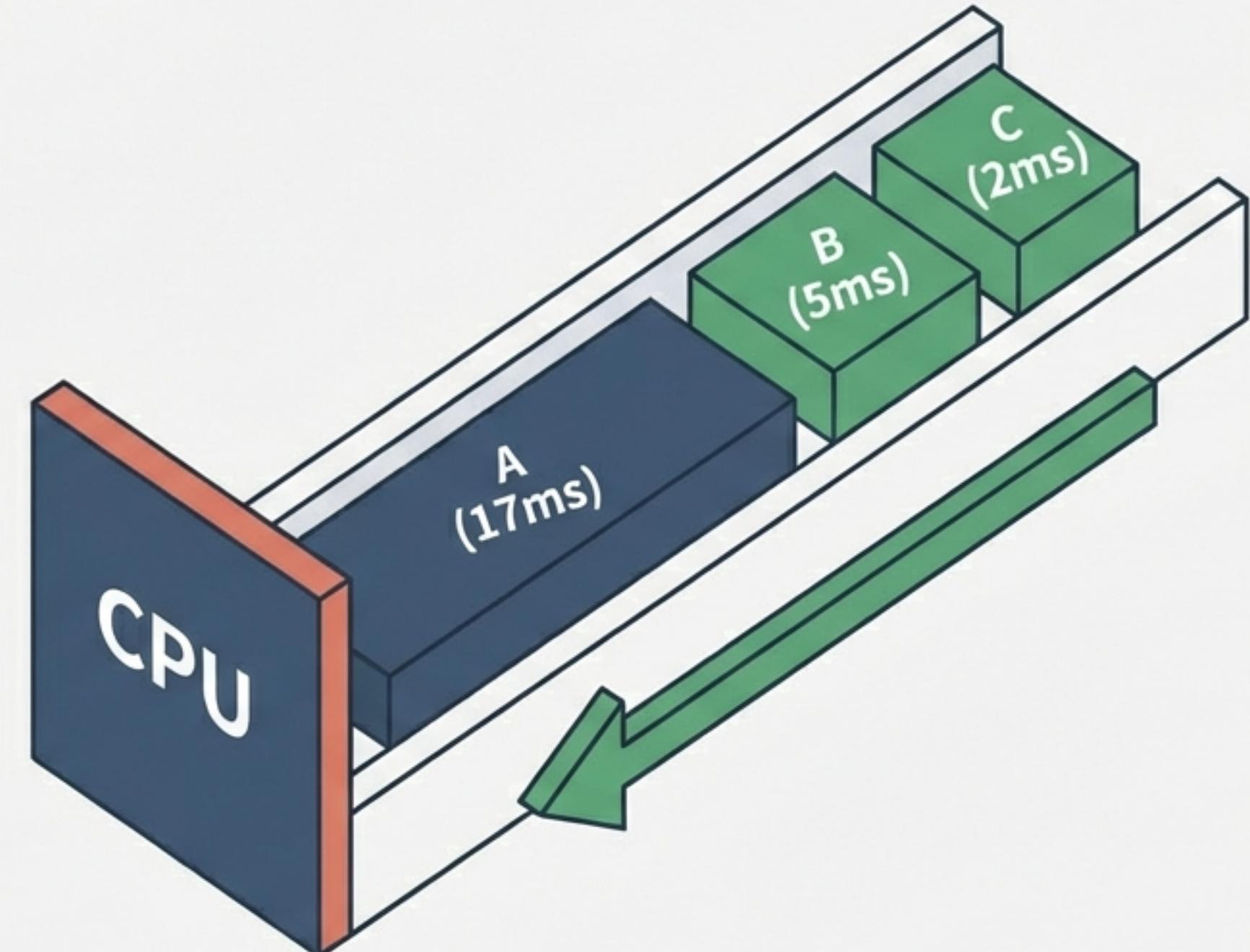
- 운영체제마다 사용하는 알고리즘은 다르지만, 핵심 아이디어는 공유됩니다.
- 단순한 방식에서 시작해, 발생한 문제를 해결하며 발전해 나가는 과정을 따라가 봅니다.



1. 선입선처리 스케줄링 (FCFS)

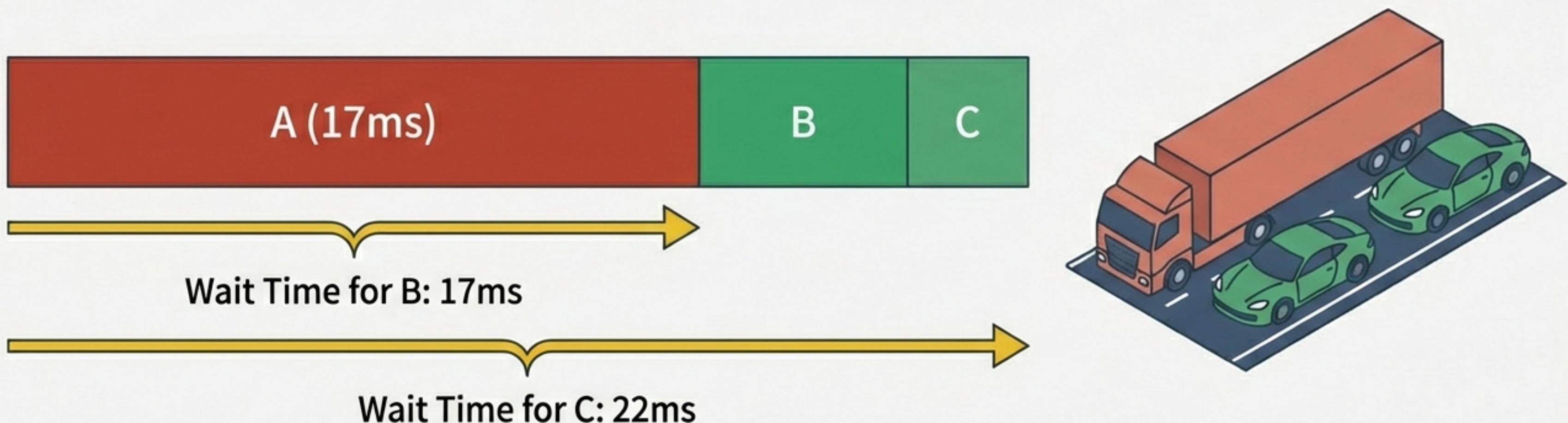
First Come, First Served

- 준비 큐(Ready Queue)에 삽입된 순서대로 CPU를 할당하는 비선점(Non-preemptive) 방식.
- 먼저 요청한 프로세스가 먼저 처리된다.
- 매우 직관적이고 공정해 보임.



FCFS의 치명적 단점: 호위 효과

- 실행 시간이 긴 프로세스가 먼저 도착하면, 뒤에 있는 짧은 프로세스들이 하염없이 기다려야 함.
- 호위 효과 (Convoy Effect): 버스 한 대가 막아서 승용차들이 줄줄이 밀리는 현상과 동일.

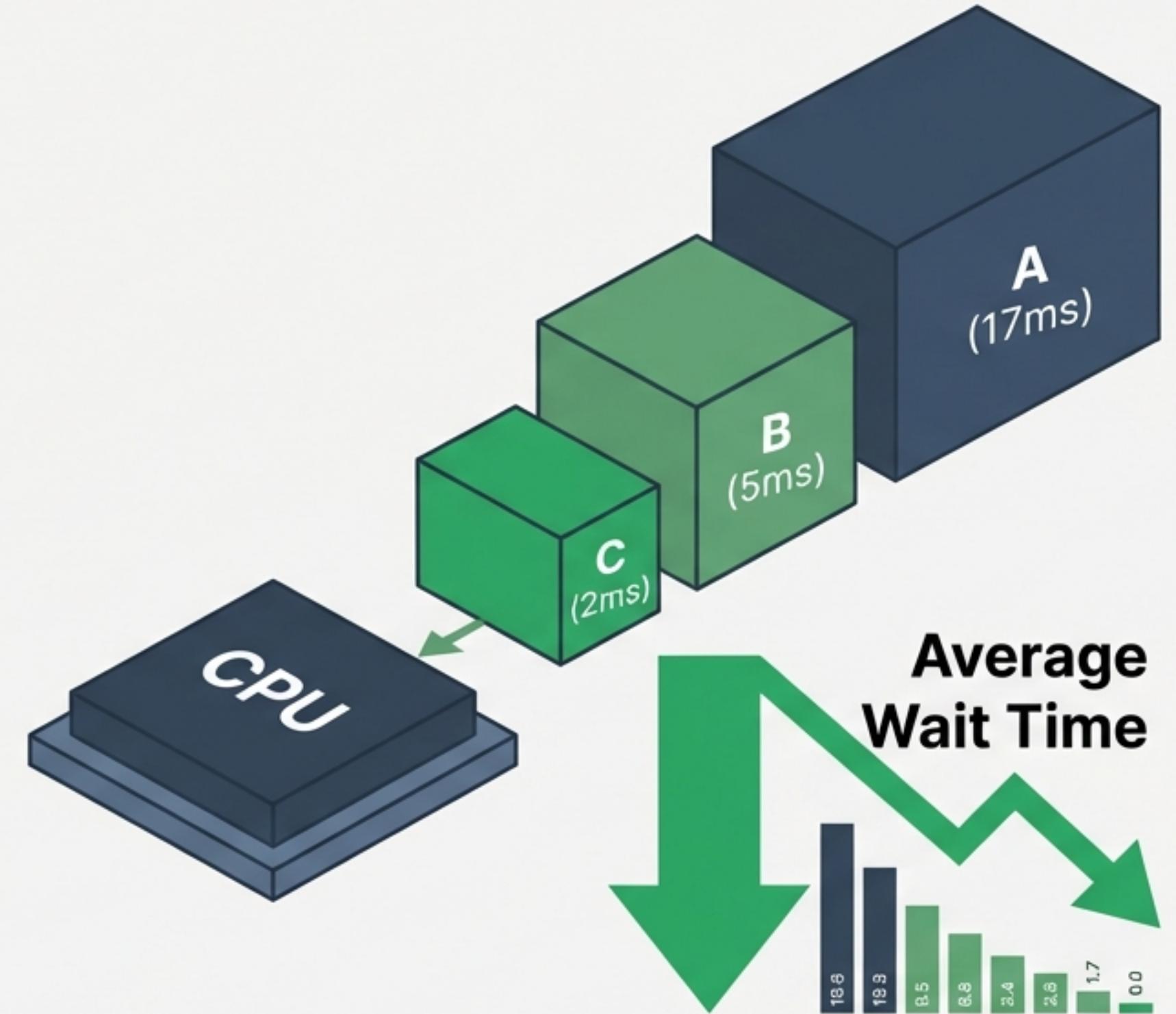


2. 최단 작업 우선 스케줄링 (SJF)

Shortest Job First

- 기다리는 시간을 줄이려면,
짧은 작업부터 먼저 처리하자.
- CPU 사용 시간이 가장 짧은
프로세스부터 실행.
- FCFS의 호위 효과를 해결하고
평균 대기 시간을 최소화함.

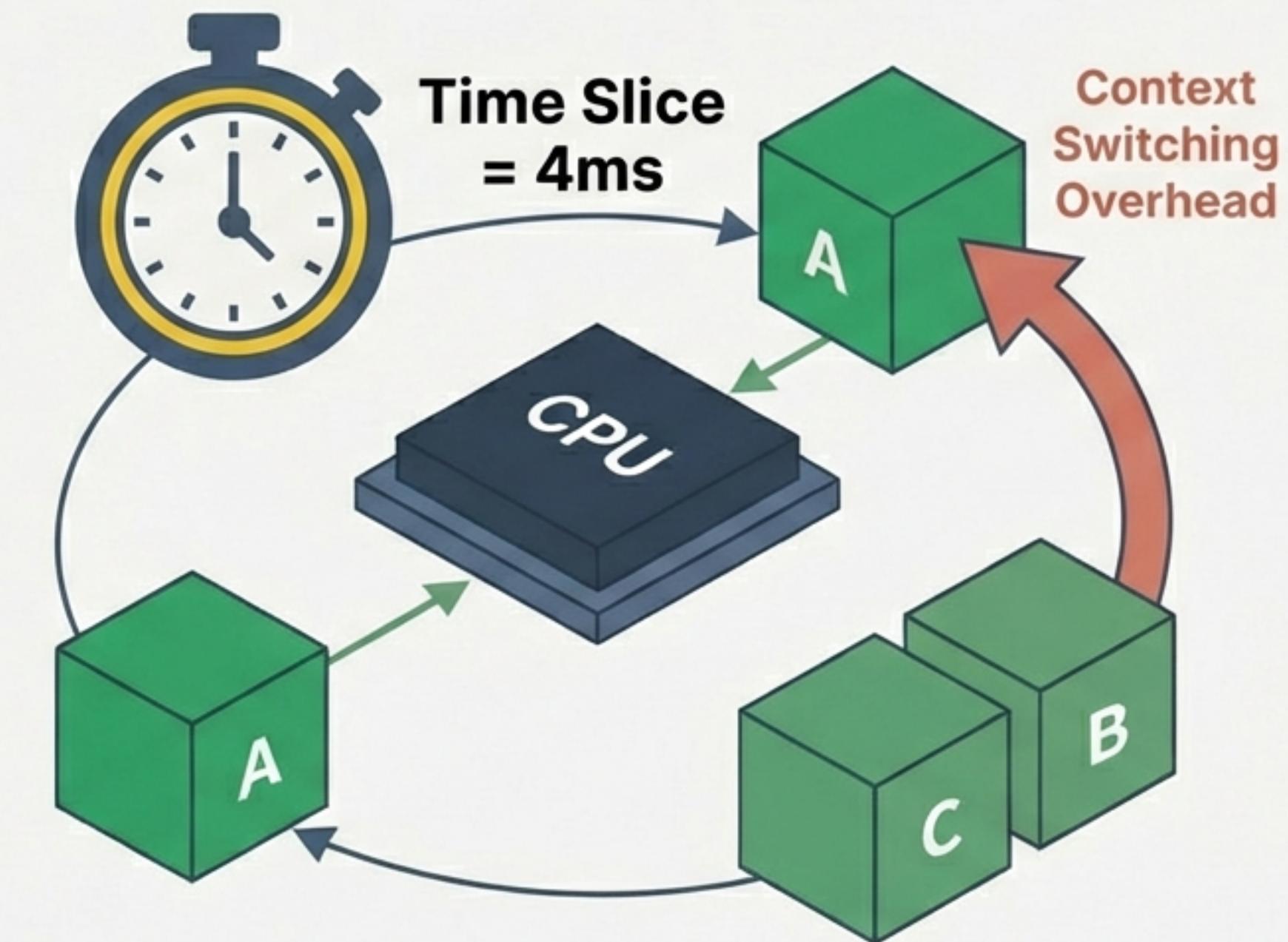
비선점형이 기본이나, 선점형 구현도 가능.



3. 라운드 로빈 스케줄링 (RR)

Round Robin

- FCFS + 타임 슬라이스 (Time Slice)
- 정해진 시간만큼 CPU를 사용하고, 완료되지 않으면 큐의 맨 뒤로 이동.
- 선점형(Preemptive) 스케줄링의 대표적 예시.



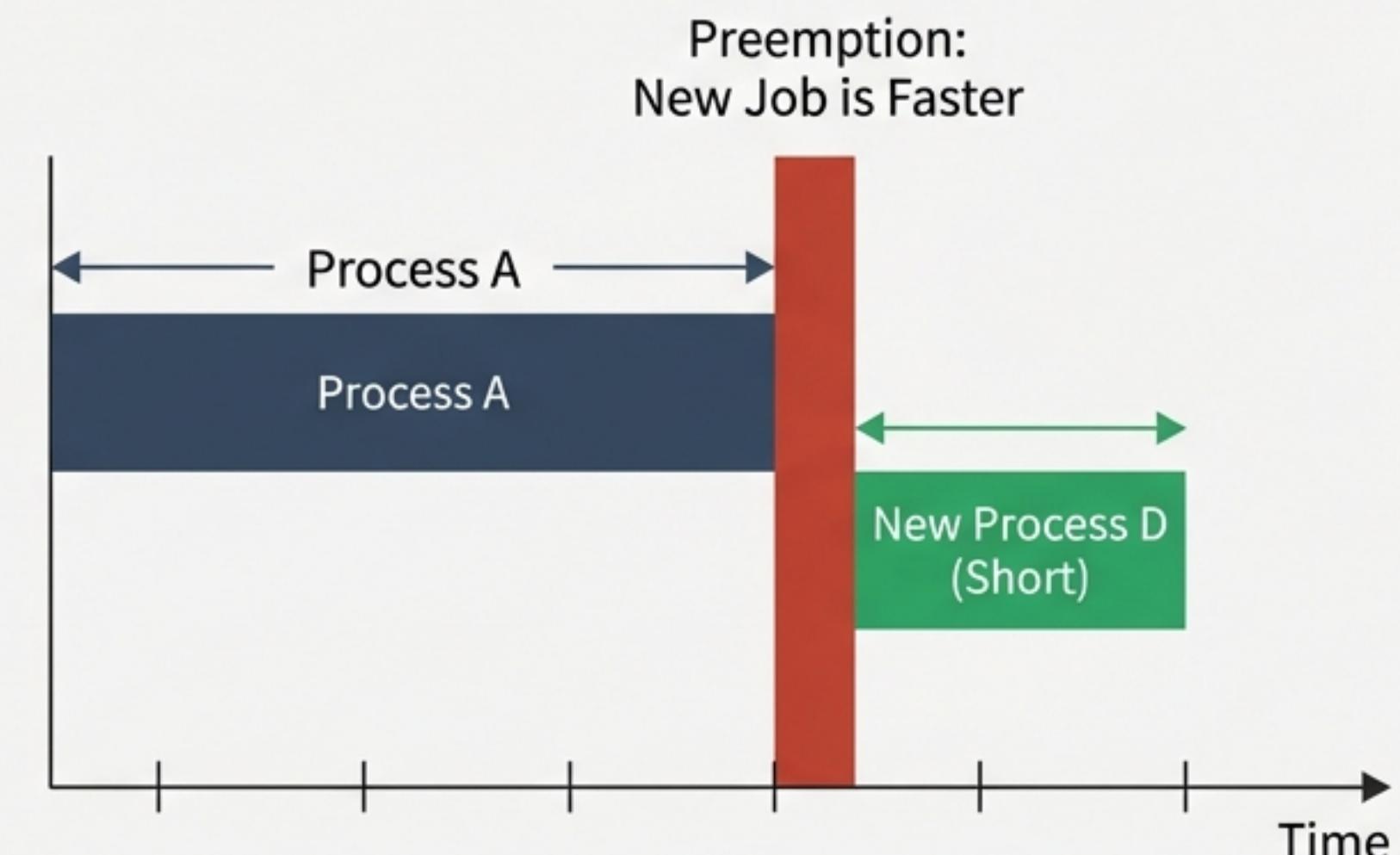
4. 최소 잔여 시간 우선 스케줄링 (SRT)

Shortest Remaining Time

SjF (최단 작업 우선) + Round Robin
(타임 슬라이스)

정해진 타임 슬라이스만큼 사용하되,
다음 프로세스는 남은 작업 시간이
가장 적은 것을 선택.

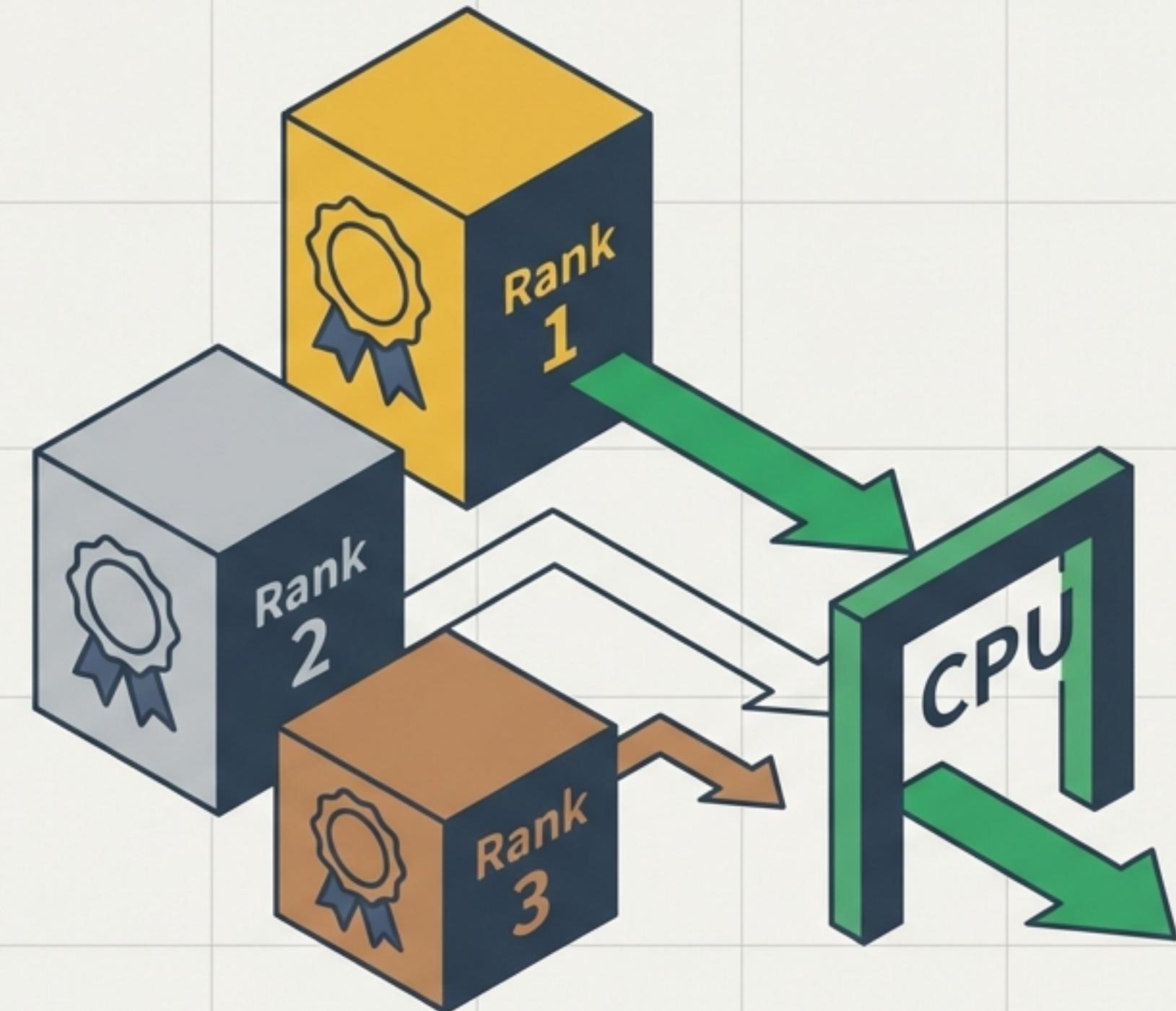
새 프로세스의 시간이 현재 남은 시간보다
짧으면 CPU를 즉시 뺏어옴 (Preemptive).



5. 우선순위 스케줄링

Priority Scheduling

- 프로세스마다 우선순위(Priority)를 부여하고, 가장 높은 순서대로 실행.
- SJF/SRT도 넓은 의미의 우선순위 스케줄링임 (짧은 시간 = 높은 우선순위).
- 우선순위가 같으면 FCFS로 처리.

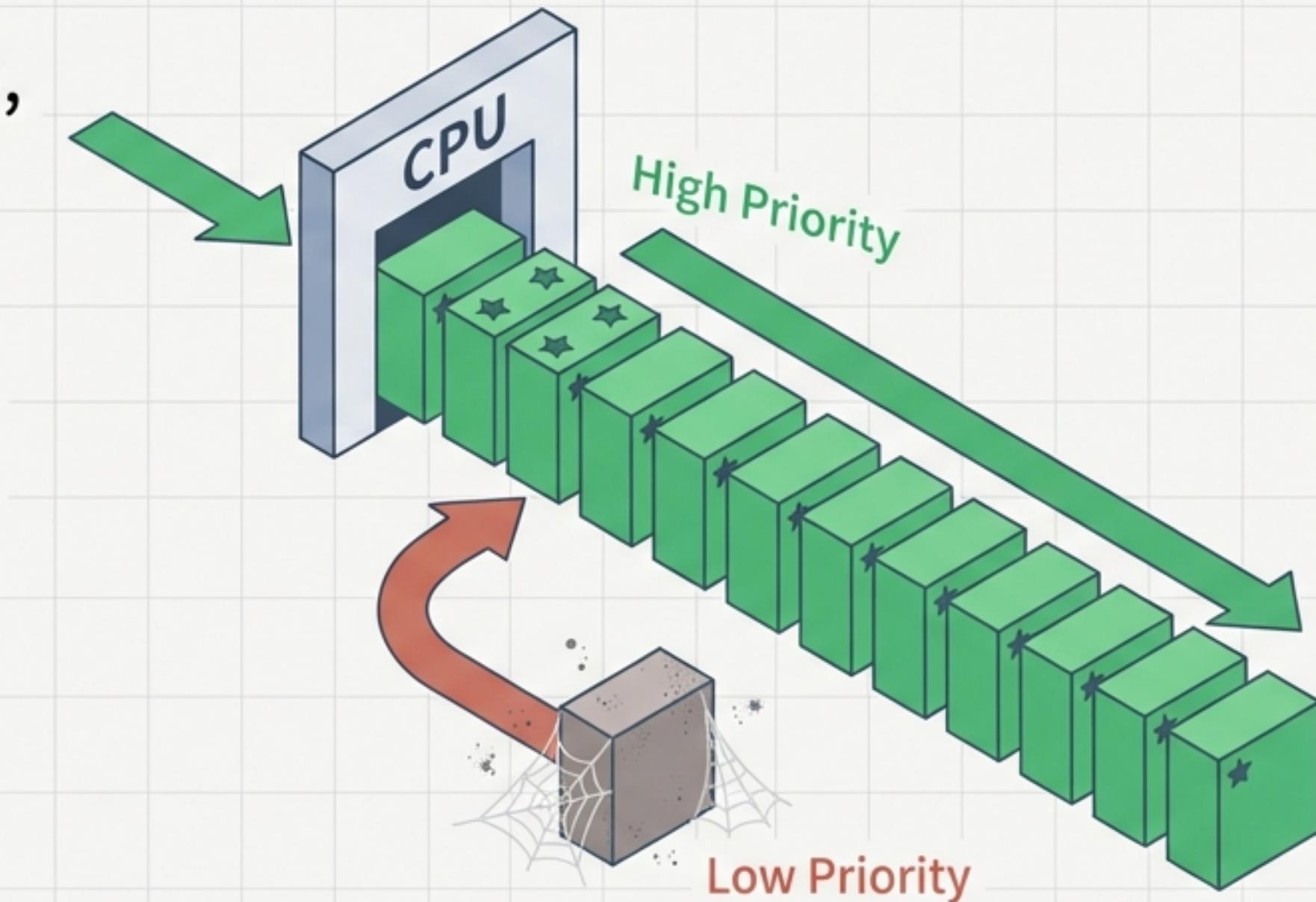


우선순위 방식의 치명적 단점: 기아 현상

Starvation

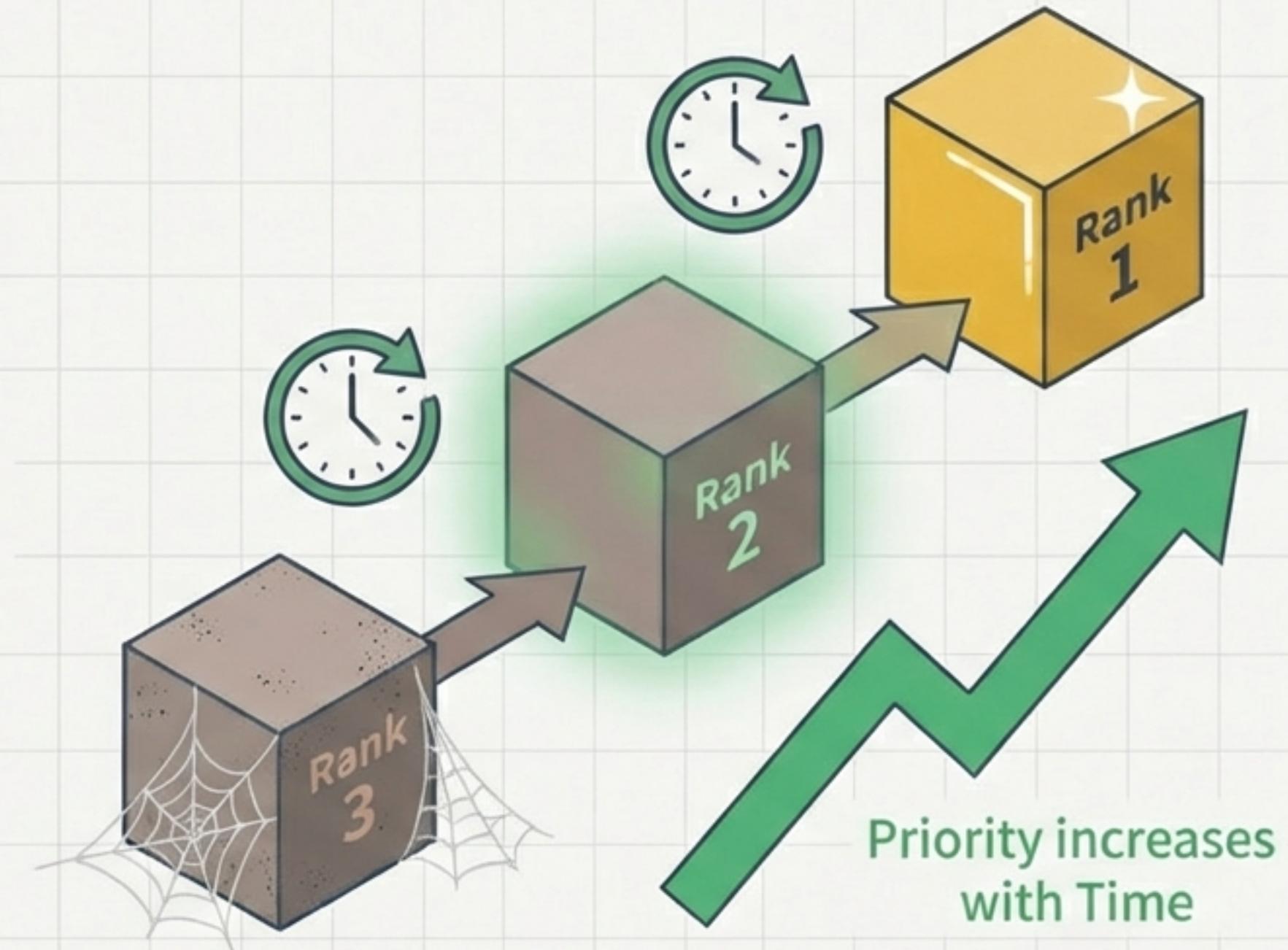
우선순위가 높은 프로세스가 계속 들어오면,
낮은 프로세스는 영원히 실행되지 못함.

기아(Starvation) / 아사 현상: 준비 큐에
먼저 들어왔음에도 실행이 무한정 연기됨.



기아 현상의 해결책: 에이징(Aging)

- 오랫동안 대기한 프로세스의 우선순위를 점차 높여주자.
- 마치 나이를 먹듯이, 대기 시간이 길어질수록 우선순위를 1단계씩 승격.
- 결국 언젠가는 가장 높은 우선순위가 되어 실행됨을 보장.



Priority increases
with Time

6. 다단계 큐 스케줄링 (Multilevel Queue)

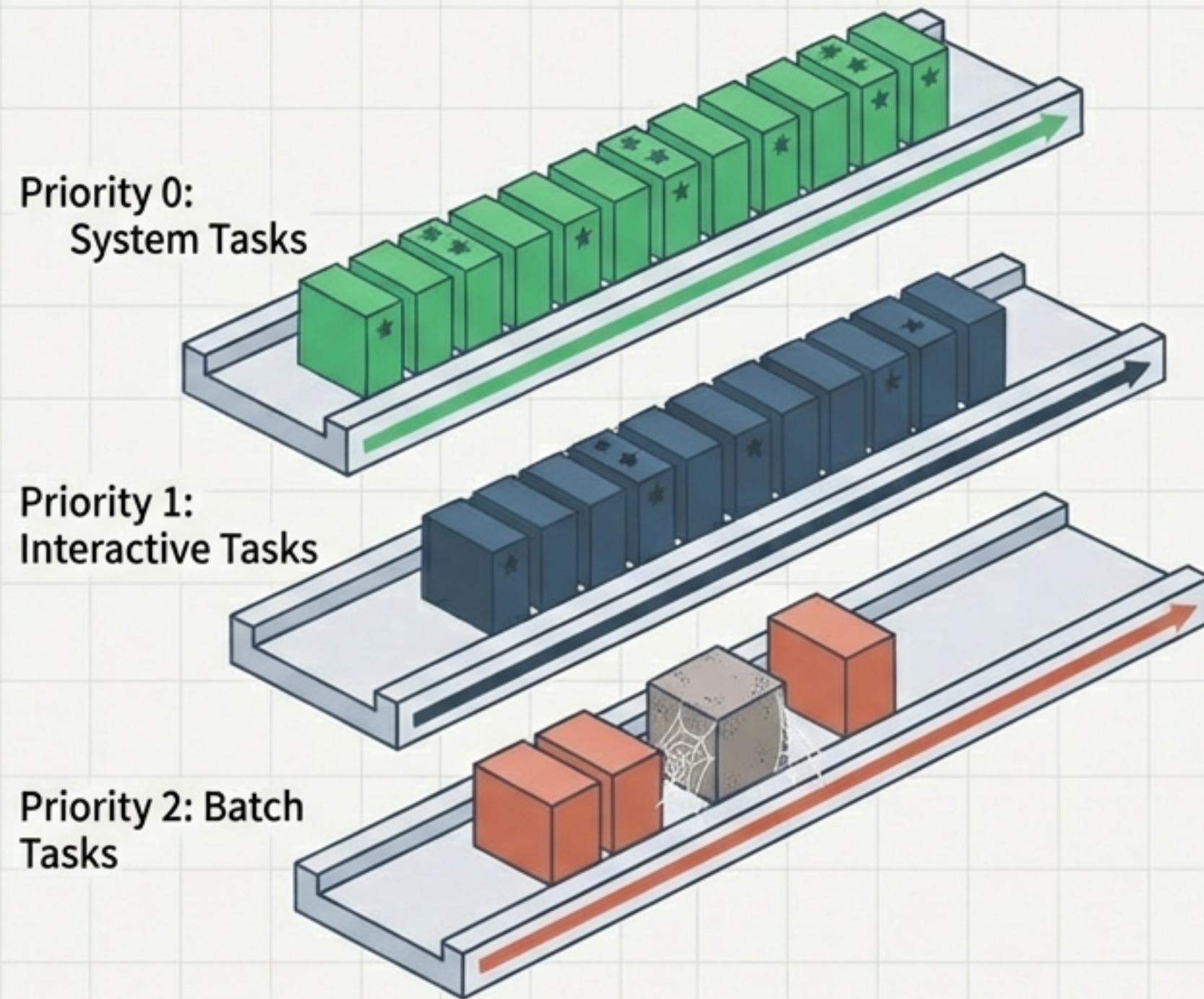
우선순위별로 별도의 큐(Queue)를

여러 개 생성.

큐마다 다른 규칙 적용 가능
(e.g., RR, FCFS).

단점: 큐 간의 이동이 불가능함

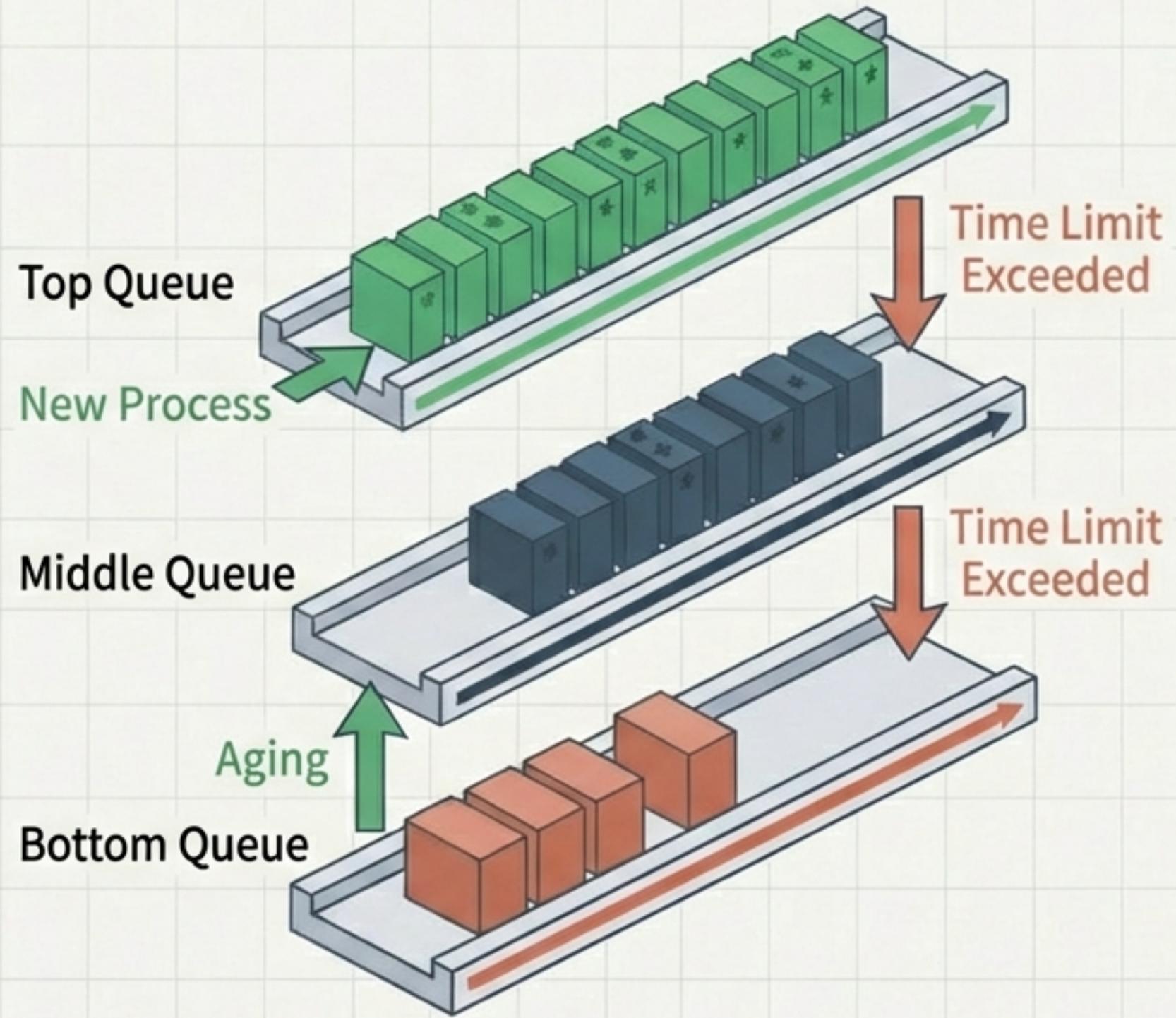
-> 기아 현상 가능성 여전함.



7. 다단계 피드백 큐 스케줄링 (MFQ)

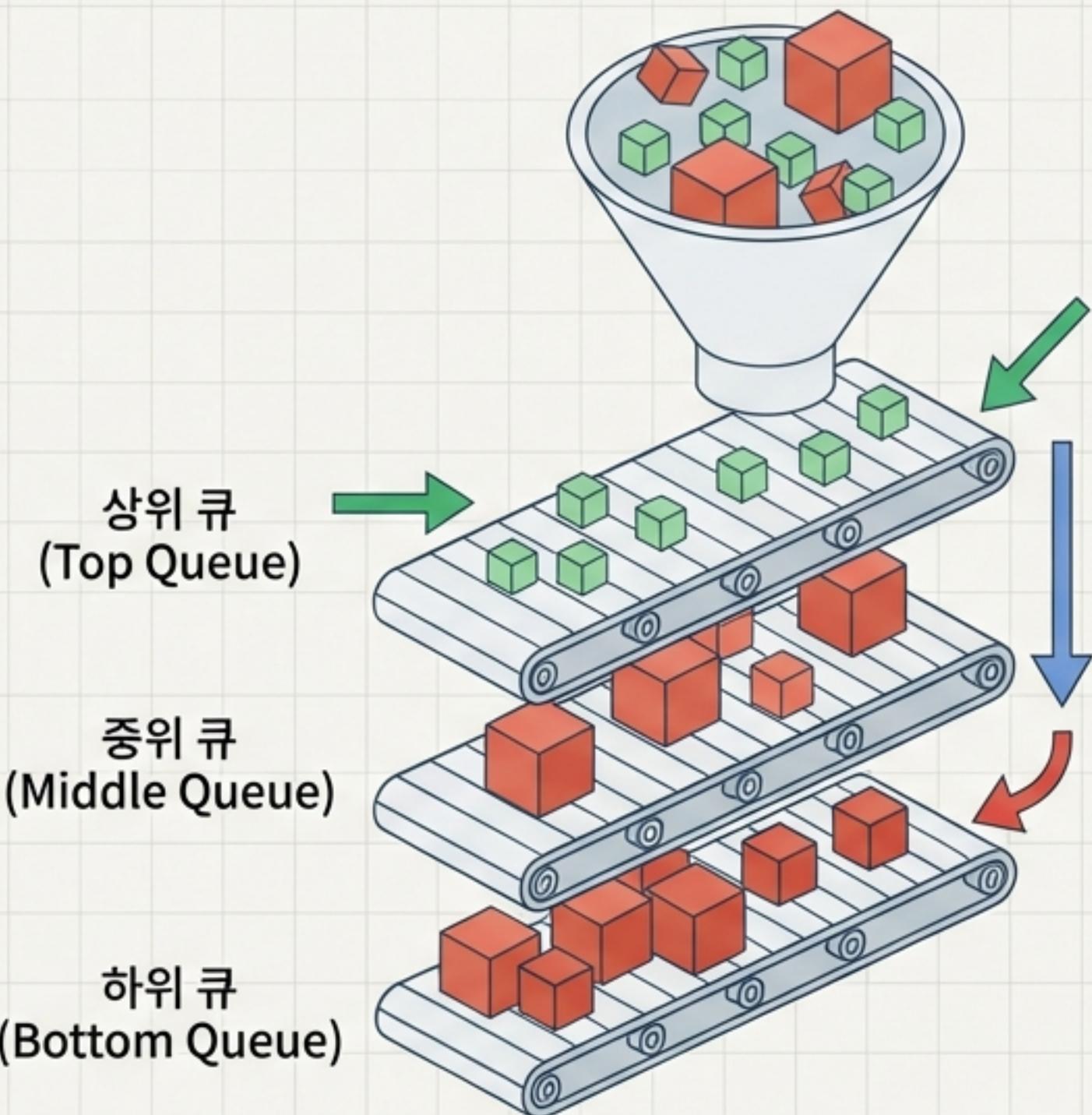
Multilevel Feedback Queue

- 다단계 큐 + 큐 간의 이동 허용.
- 새 프로세스는 최상위 큐 진입.
- 시간 초과시 -> 하위 큐로 강등.
- 대기 시간 초과시(Aging) -> 상위 큐로 승격.

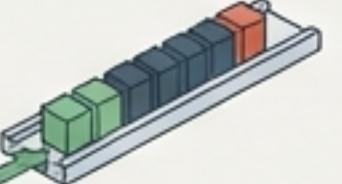
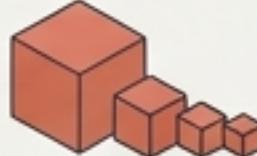
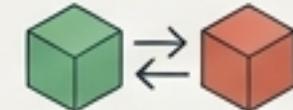
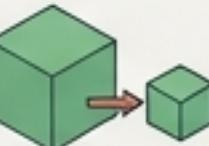
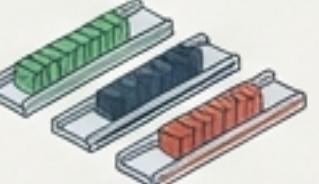
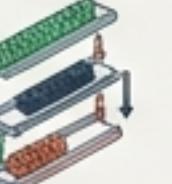


8. 왜 다단계 피드백 큐(MFQ)가 강력한가?

- CPU 집중 프로세스 (CPU-bound):
무겁고 오래 걸림 -> 자연스럽게 하위 큐로
침전.
- 입출력 집중 프로세스 (I/O-bound):
가볍고 짧음 -> 상위 큐 유지.
- 가장 일반적이고 현대적인 스케줄링 형태.

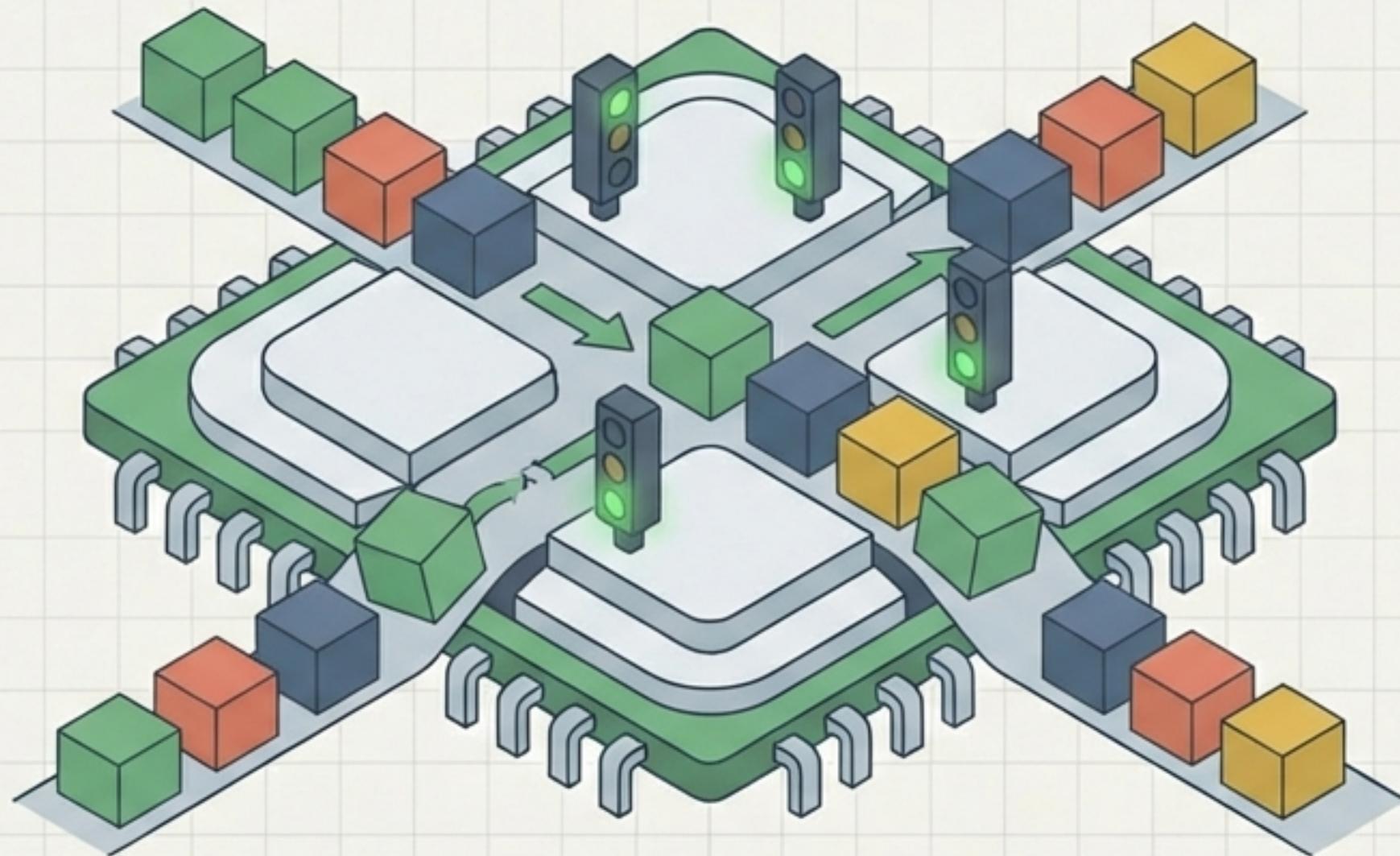


한눈에 보는 7가지 알고리즘

알고리즘 (Algorithm)	특징 (Feature)	단점/비고 (Drawback/Note)
FCFS	도착 순서대로 처리	 호위 효과 (Convoy Effect) 
SJF	짧은 작업 우선	 긴 작업 기아 현상 (Starvation) 
RR	타임 슬라이스 (Time Slice)	 문맥 교환 오버헤드 
SRT	SJF + 선점형	 기아 현상 
Priority	우선순위 순서	 기아 현상 (에이징으로 해결) 
Multilevel Q	큐별 독립 관리	 유연성 부족 
MFQ	큐 간 이동 가능	 가장 일반적/복잡함 

스케줄링의 본질은 ‘자원 분배의 최적화’입니다

이름을 외우기보다, 각 알고리즘이 어떤 문제를 해결하기 위해 등장했는지 그 흐름을 기억하세요.



본 콘텐츠는 [한빛미디어] ‘혼자 공부하는 컴퓨터 구조+운영체제’ (강사 강민철) 강의를 바탕으로 재구성되었습니다.