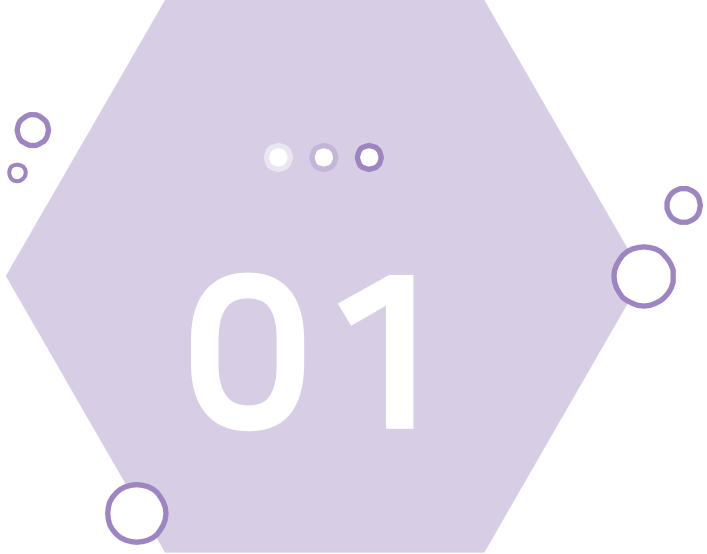
10강. 가상 메모리 II

방송대 컴퓨터과학과
김진욱 교수

목차

01 다양한 페이지 교체기법

02 프로세스별 페이지 집합 관리

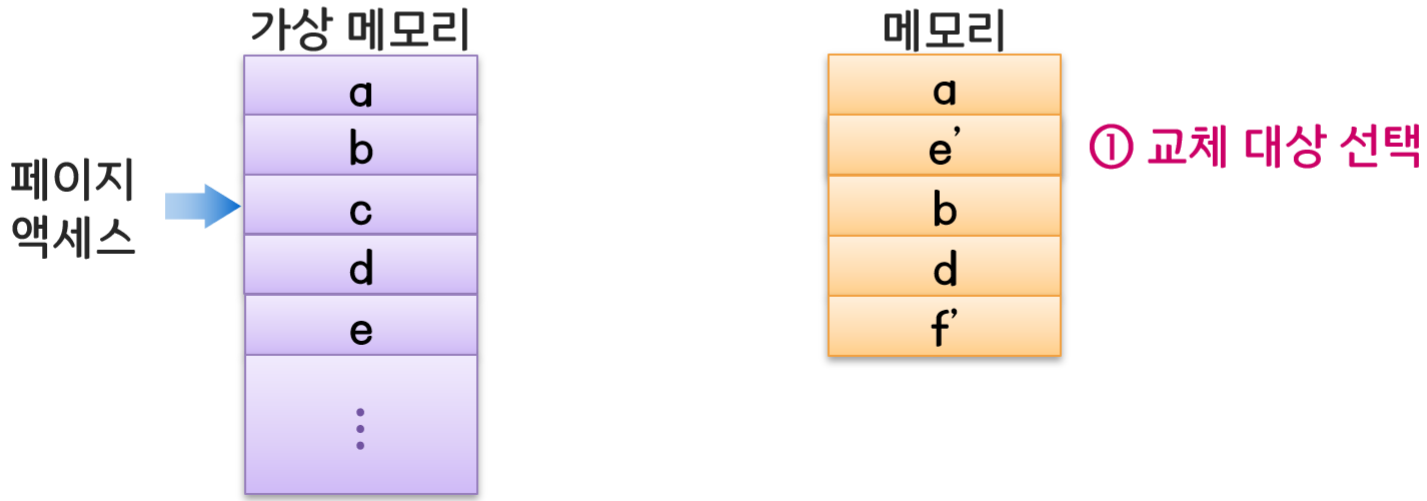


다양한 페이지 교체기법

페이지 교체기법

페이지 교체기법

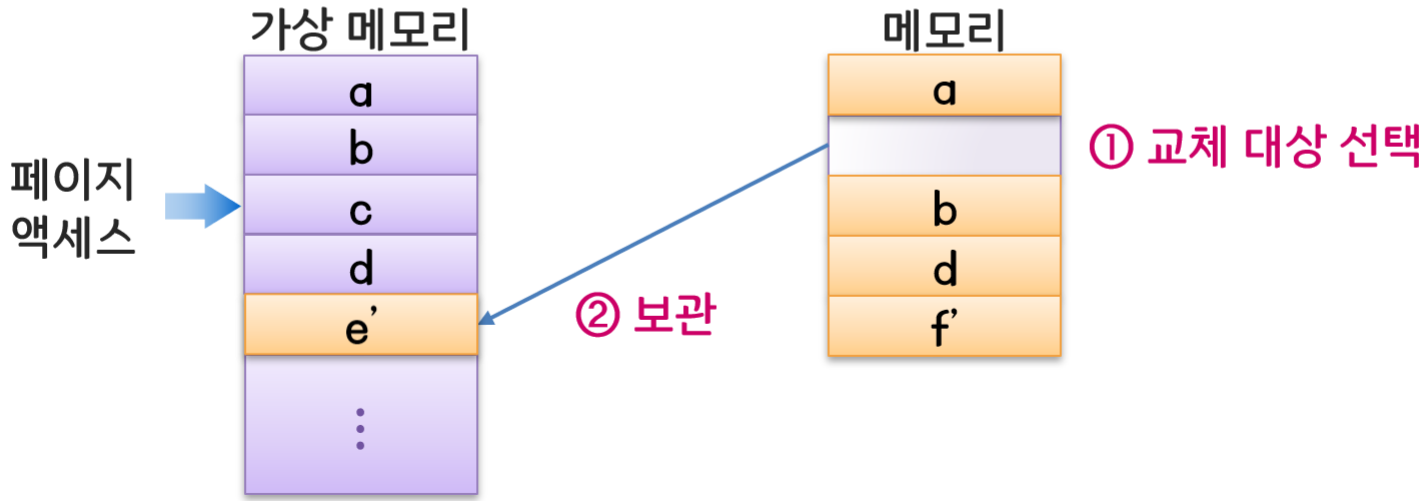
- 모든 페이지 프레임이 사용되고 있을 때 새로 적재되어야 할 페이지를 위하여 어느 페이지를 교체할 것인가를 결정
- 교체 대상 선택 → 보조기억장치에 보관 → 새로운 페이지를 적재



페이지 교체기법

페이지 교체기법

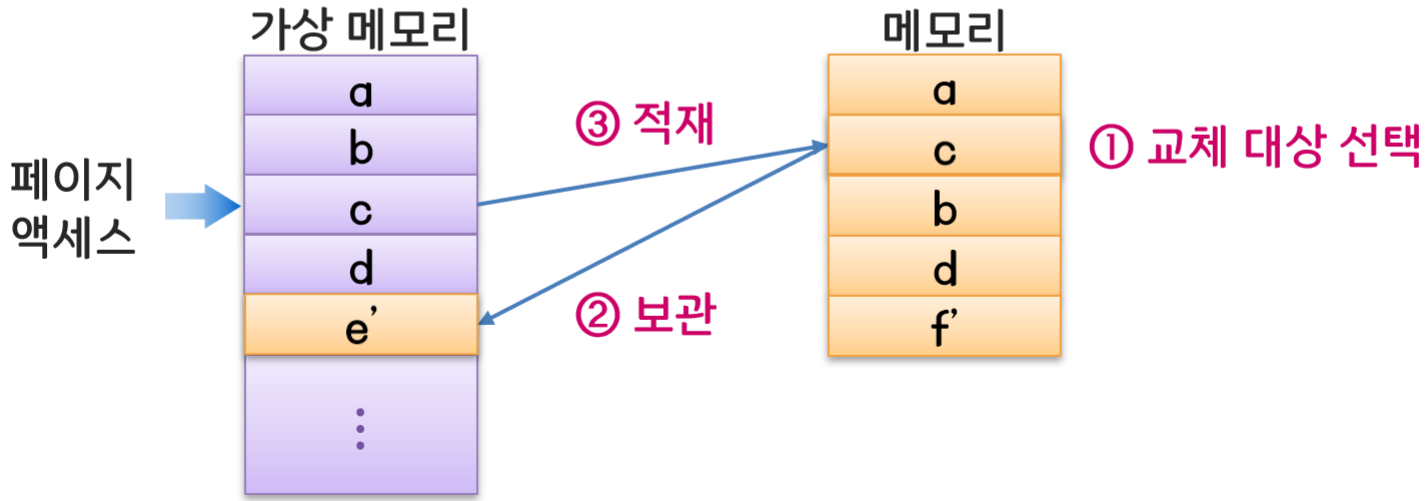
- 모든 페이지 프레임이 사용되고 있을 때 새로 적재되어야 할 페이지를 위하여 어느 페이지를 교체할 것인가를 결정
- 교체 대상 선택 → 보조기억장치에 보관 → 새로운 페이지를 적재



페이지 교체기법

페이지 교체기법

- 모든 페이지 프레임이 사용되고 있을 때 새로 적재되어야 할 페이지를 위하여 어느 페이지를 교체할 것인가를 결정
- 교체 대상 선택 → 보조기억장치에 보관 → 새로운 페이지를 적재



교체 대상 선택

최적화의 원칙

- 앞으로 가장 오랫동안 사용되지 않을 페이지를 교체 대상으로 선택
- 이론적으로는 최적이나 미래를 예측할 수 없어 실현 불가능

선택을 위한 기본 정책

- 대체로 좋은 결론을 내리면서 시간 및 공간의 오버헤드가 적은 방법

교체 제외 페이지

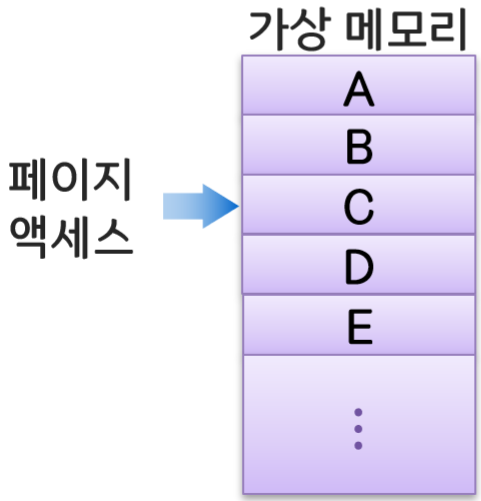
- 페이징을 위한 슈퍼바이저 코드 영역, 보조기억장치 드라이버 영역, 입출력장치를 위한 데이터 버퍼 영역 등

페이지 교체 알고리즘

- FIFO 페이지 교체기법
- LRU 페이지 교체기법
- LFU 페이지 교체기법
- NUR 페이지 교체기법
- 2차 기회 페이지 교체기법
- 클럭 페이지 교체기법
- 워킹세트 알고리즘, PFF 알고리즘

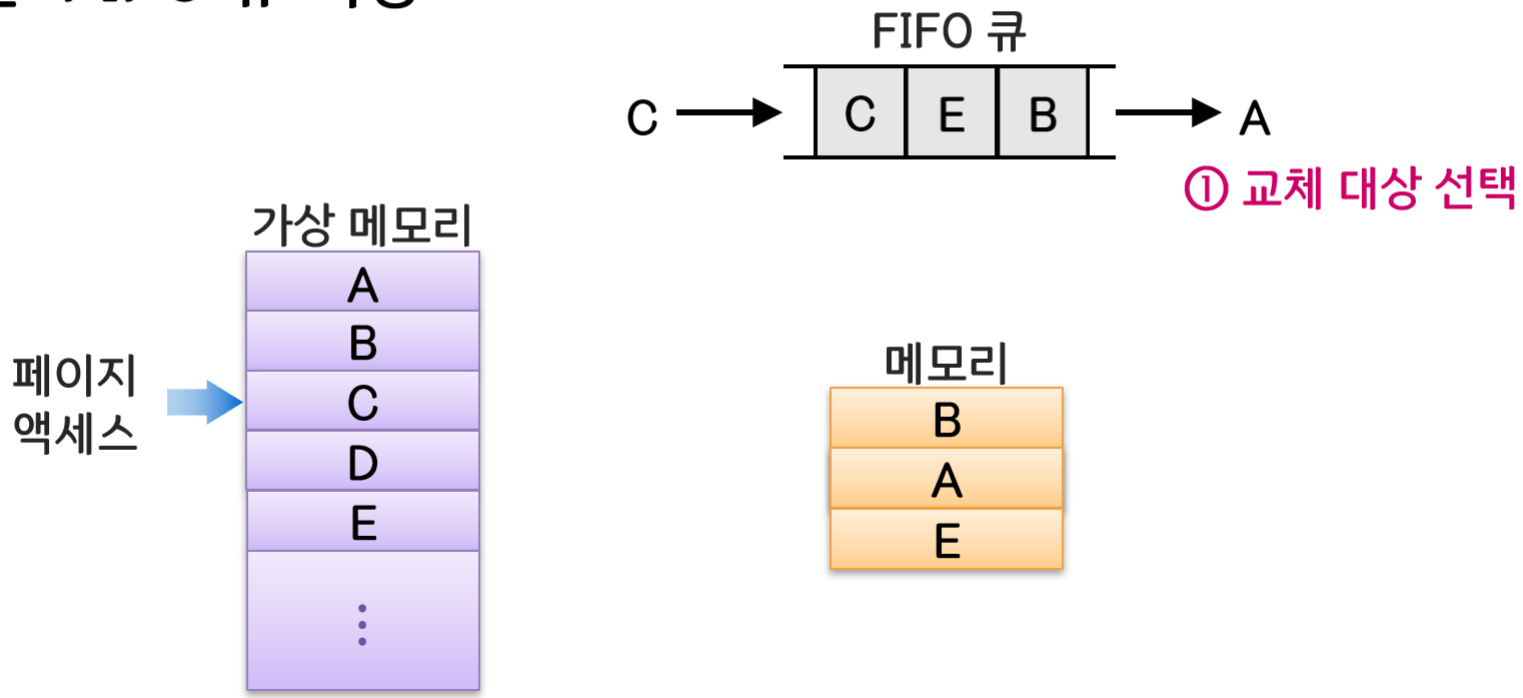
FIFO (First-In First-Out) 페이지 교체기법

- 메모리 내에 가장 오래있었던 페이지를 교체
- 구현: FIFO 큐 이용



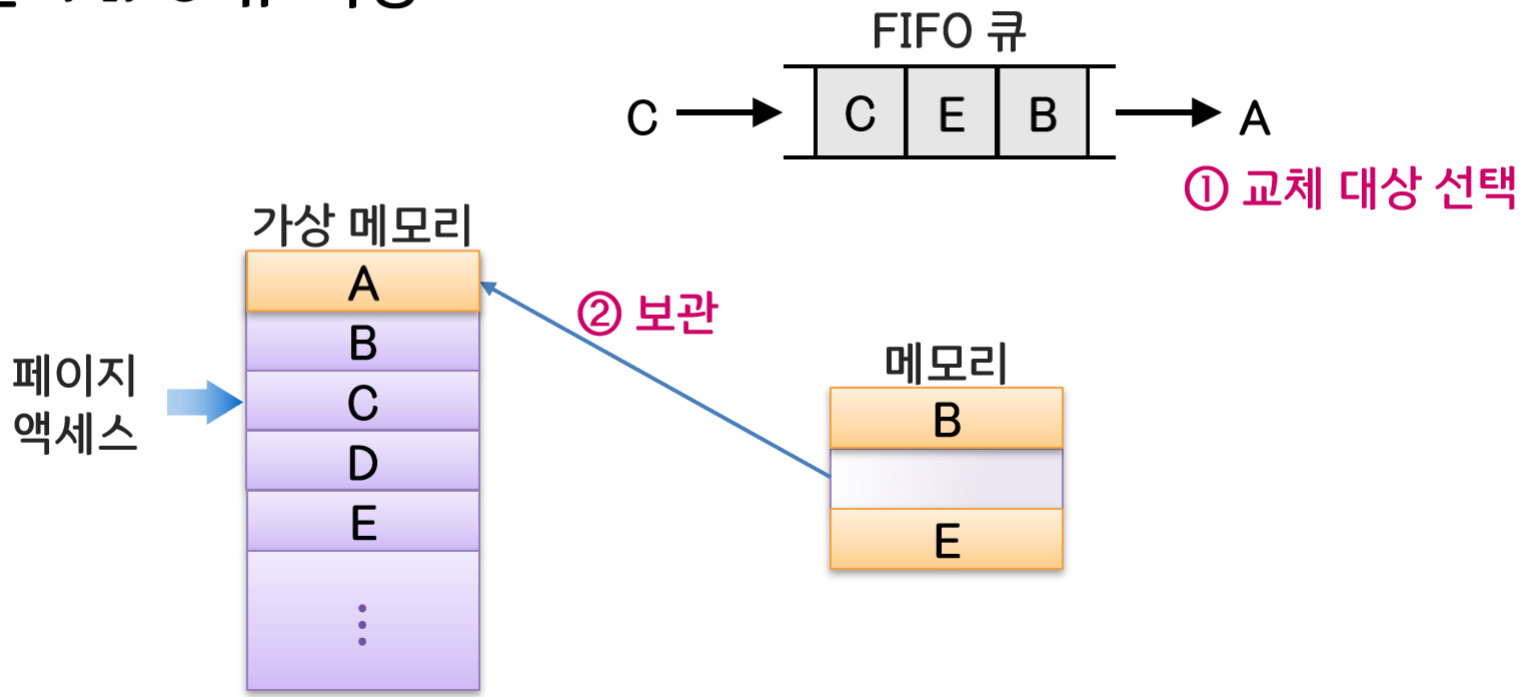
FIFO (First-In First-Out) 페이지 교체기법

- 메모리 내에 가장 오래있었던 페이지를 교체
- 구현: FIFO 큐 이용



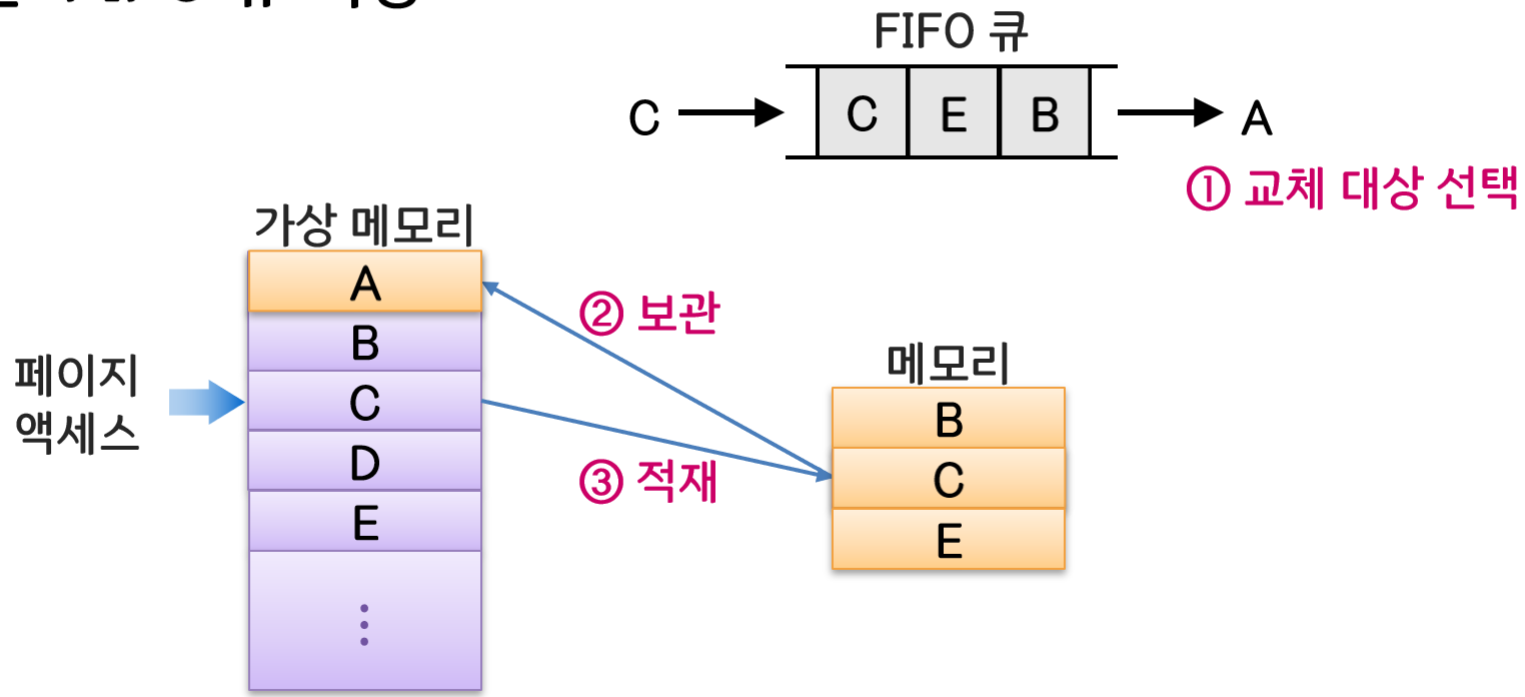
FIFO (First-In First-Out) 페이지 교체기법

- 메모리 내에 가장 오래있었던 페이지를 교체
- 구현: FIFO 큐 이용



FIFO (First-In First-Out) 페이지 교체기법

- 메모리 내에 가장 오래있었던 페이지를 교체
- 구현: FIFO 큐 이용



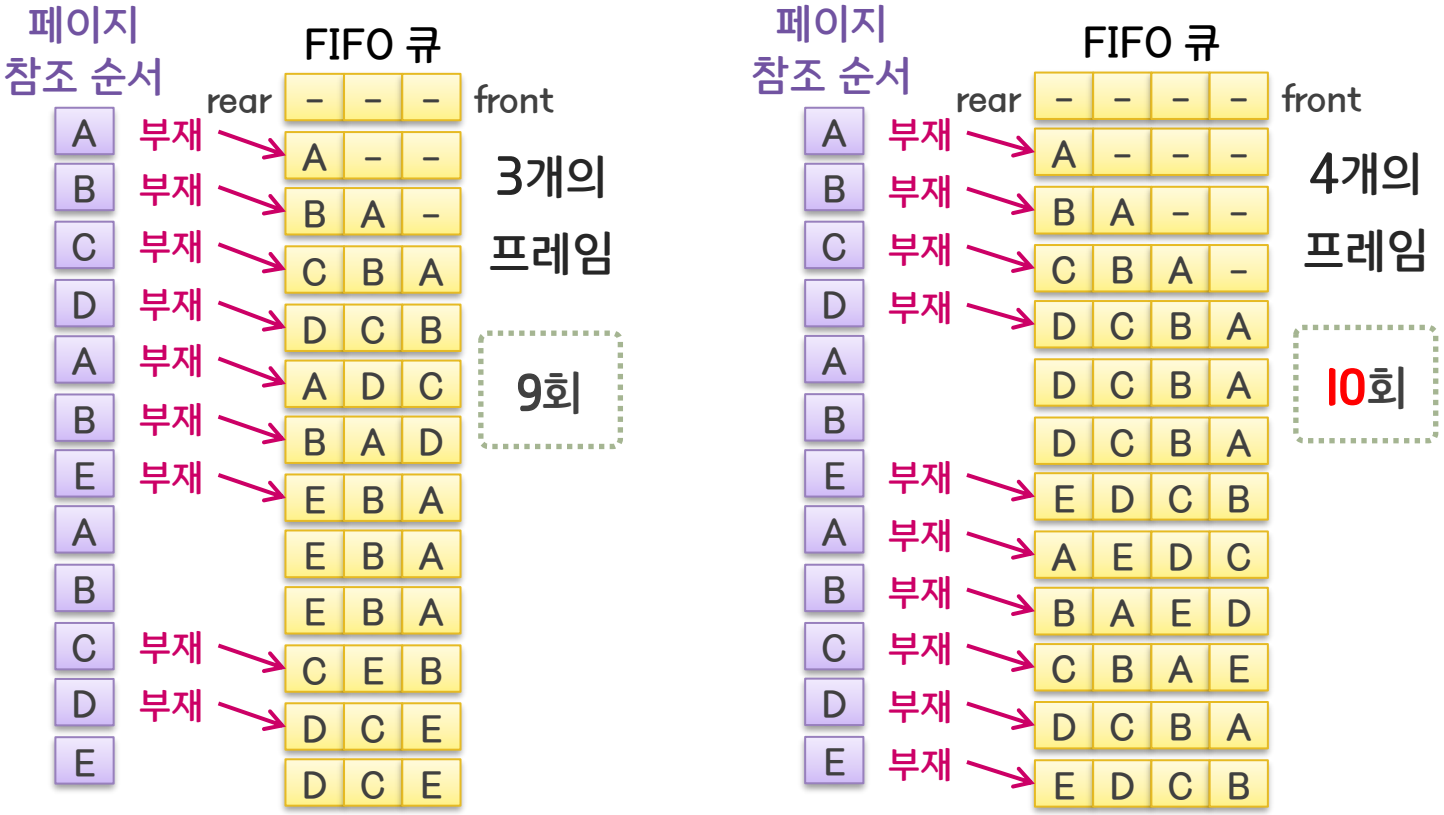
FIFO (First-In First-Out) 페이지 교체기법

- 메모리 내에 가장 오래있었던 페이지를 교체
- 구현: FIFO 큐 이용

- 단점
  - 오래 전에 적재되어 반복적으로 사용되는 페이지가 교체될 수 있음
  - Belady의 이상현상 발생
    - ↳ 프로세스에 더 많은 수의 페이지 프레임을 할당할 경우 오히려 페이지 부재가 더 많이 발생할 수 있는 현상

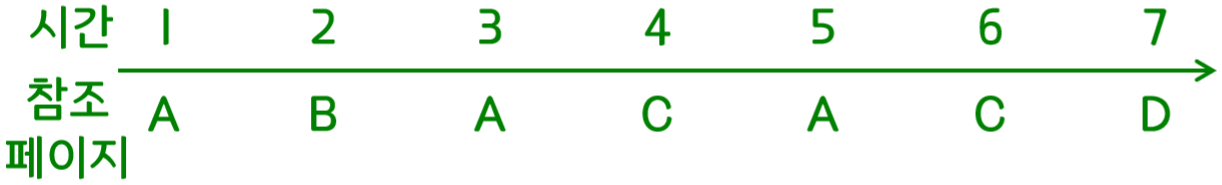
FIFO (First-In First-Out) 페이지 교체기법

- Belady의 이상현상 예



LRU (Least Recently Used) 페이지 교체기법

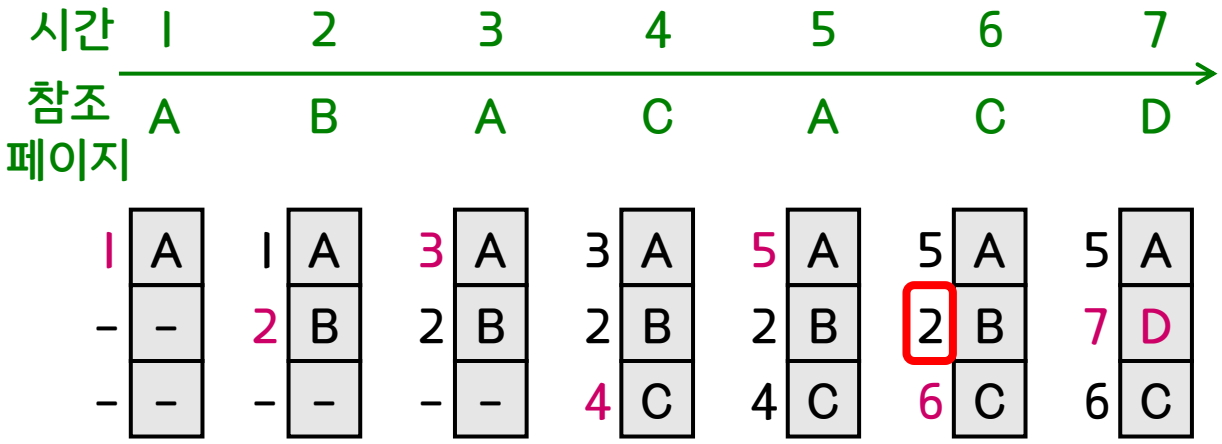
- 가장 오랫동안 사용되지 않은 페이지를 교체
- 구현: 참조시간 이용 또는 리스트 이용



- 페이지가 참조될 때마다 그때의 시간을 기록
- 참조시간이 가장 오래된 페이지를 교체

LRU (Least Recently Used) 페이지 교체기법

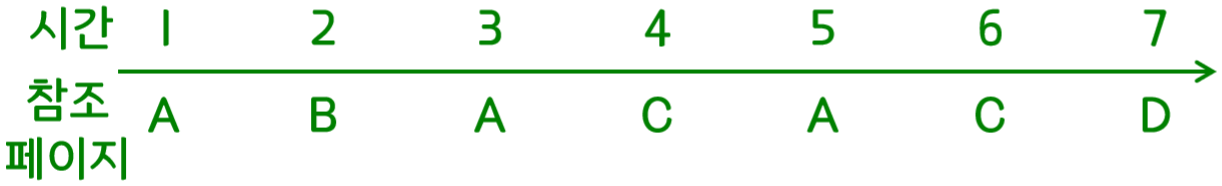
- 가장 오랫동안 사용되지 않은 페이지를 교체
- 구현: 참조시간 이용 또는 리스트 이용



- 페이지가 참조될 때마다 그때의 시간을 기록
- 참조시간이 가장 오래된 페이지를 교체

LRU (Least Recently Used) 페이지 교체기법

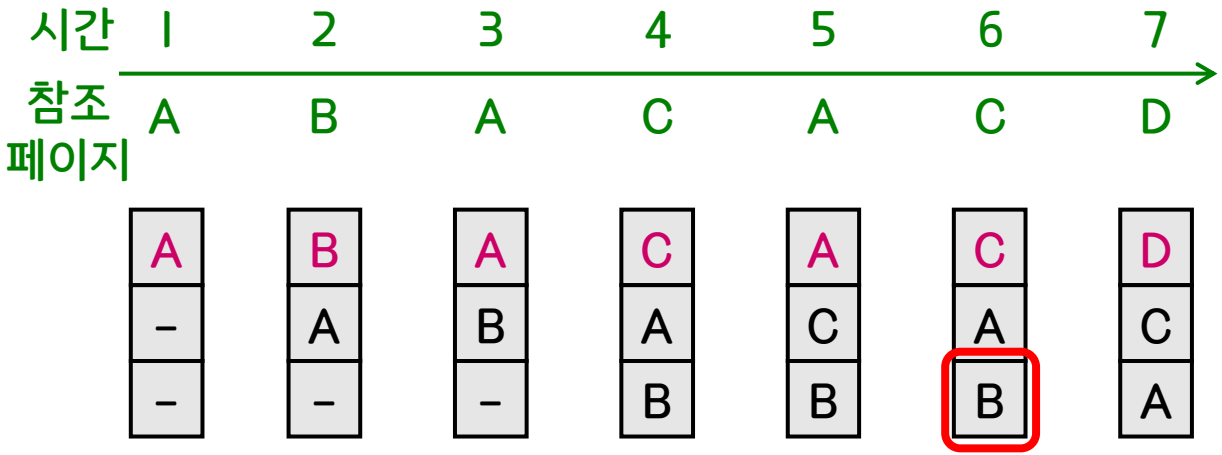
- 가장 오랫동안 사용되지 않은 페이지를 교체
- 구현: 참조시간 이용 또는 리스트 이용



- 페이지가 참조되면 리스트의 선두로 옮김
- 리스트의 끝에 있는 페이지를 교체

LRU (Least Recently Used) 페이지 교체기법

- 가장 오랫동안 사용되지 않은 페이지를 교체
- 구현: 참조시간 이용 또는 리스트 이용



- 페이지가 참조되면 리스트의 선두로 옮김
- 리스트의 끝에 있는 페이지를 교체

LRU (Least Recently Used) 페이지 교체기법

- 가장 오랫동안 사용되지 않은 페이지를 교체
- 구현: 참조시간 이용 또는 리스트 이용

- 특징
  - Belady의 이상현상 발생하지 않음
  - 많은 경우 최적화 원칙에 근사한 선택을 함
  - 국부성(locality)에 기반
    - ↳ 어느 한순간에 특정 부분을 집중적으로 참조

시간 국부성

현재 참조된 기억장소는
가까운 미래에도 계속
참조될 가능성이 높음

공간 국부성

하나의 기억장소가 참조되면
근처의 기억장소가 계속
참조될 가능성이 높음

LRU (Least Recently Used) 페이지 교체기법

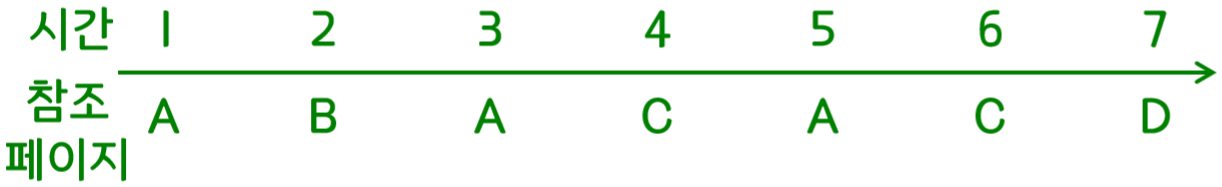
- 가장 오랫동안 사용되지 않은 페이지를 교체
- 구현: 참조시간 이용 또는 리스트 이용

- 특징
  - Belady의 이상현상 발생하지 않음
  - 많은 경우 최적화 원칙에 근사한 선택을 함
  - 국부성(locality)에 기반
    - ↪ 어느 한순간에 특정 부분을 집중적으로 참조

- 단점
  - 경험적 판단이 맞지 않는 상황도 존재
  - 막대한 오버헤드

LFU (Least Frequently Used) 페이지 교체기법

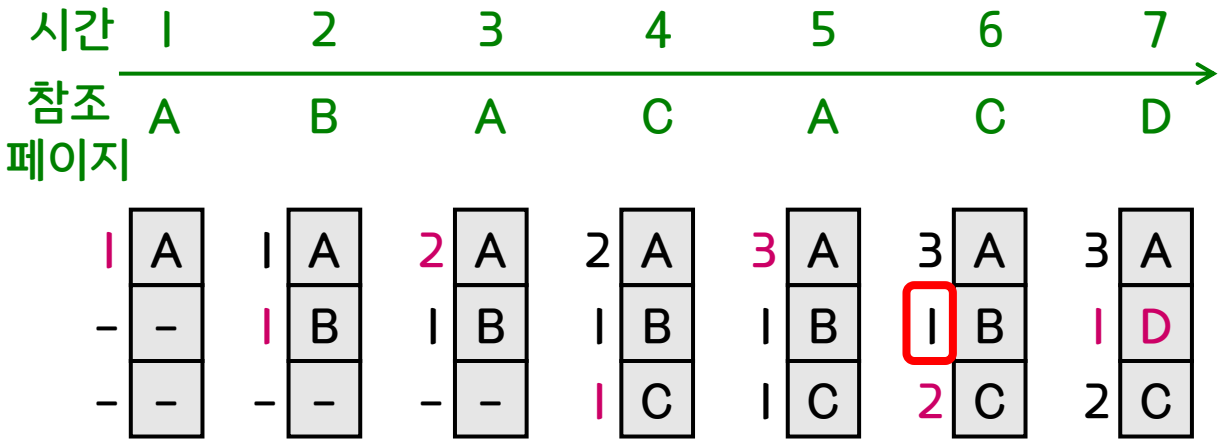
- 참조된 횟수가 가장 적은 페이지를 교체
- 구현: 참조횟수 이용



- 페이지가 참조될 때마다 증가된 횟수 기록
- 참조횟수가 가장 적은 페이지를 교체

LFU (Least Frequently Used) 페이지 교체기법

- 참조된 횟수가 가장 적은 페이지를 교체
- 구현: 참조횟수 이용



- 페이지가 참조될 때마다 증가된 횟수 기록
- 참조횟수가 가장 적은 페이지를 교체

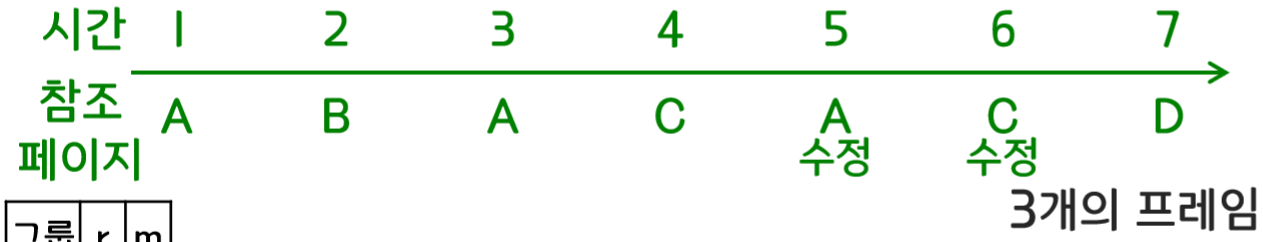
LFU (Least Frequently Used) 페이지 교체기법

- 참조된 횟수가 가장 적은 페이지를 교체
- 구현: 참조횟수 이용

- 단점
  - 가장 최근에 메모리로 옮겨진 페이지가 교체될 가능성 높음
  - 초기에 매우 많이 사용된 후 더 이상 사용되지 않는 페이지는 교체가능성 낮음
  - 막대한 오버헤드

NUR (Not Used Recently) 페이지 교체기법

- 참조 여부와 수정 여부에 따른 우선순위에 따라 적합한 페이지를 교체
- 구현: 페이지마다 참조 비트 r과 수정 비트 m 이용



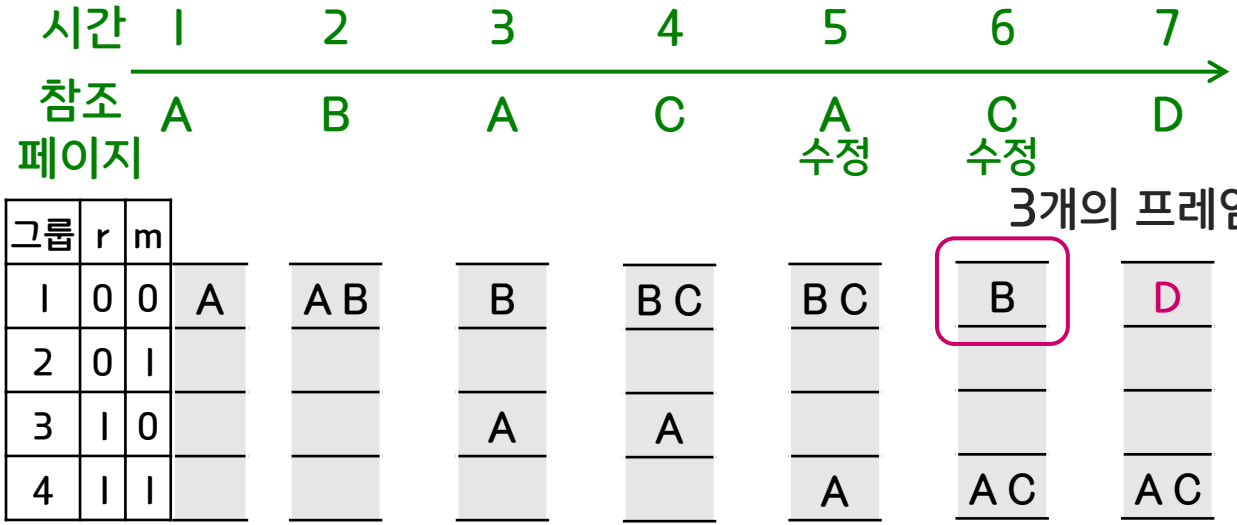
그룹	r	m
1	0	0
2	0	1
3	1	0
4	1	1

- 페이지 프레임에 적재: r=0, m=0
- 페이지를 참조: r=1
- 페이지를 수정: m=1

- 교체 우선순위:
  ① r=0, m=0
  ② r=0, m=1
  ③ r=1, m=0
  ④ r=1, m=1

NUR (Not Used Recently) 페이지 교체기법

- 참조 여부와 수정 여부에 따른 우선순위에 따라 적합한 페이지를 교체
- 구현: 페이지마다 참조 비트 r과 수정 비트 m 이용



그룹	r	m						
1	0	0	A	A B	B	B C	B C	B	D
2	0	1							
3	1	0			A	A			
4	1	1					A	A C	A C

- 페이지 프레임에 적재: r=0, m=0
- 페이지를 참조: r=1
- 페이지를 수정: m=1

- 교체 우선순위:
  - ① r=0, m=0
  - ② r=0, m=1
  - ③ r=1, m=0
  - ④ r=1, m=1

NUR (Not Used Recently) 페이지 교체기법

- 참조 여부와 수정 여부에 따른 우선순위에 따라 적합한 페이지를 교체
- 구현: 페이지마다 참조 비트 r과 수정 비트 m 이용

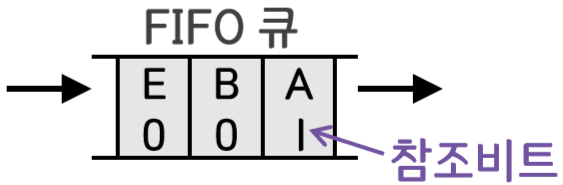
- 특징
  - 적은 오버헤드로 적절한 성능을 낼 수 있음
  - LRU와 유사하면서도 실제로 자주 쓰임
  - 동일 그룹 내에서의 선택은 무작위
  - 모든 참조 비트 r을 주기적으로 0으로 변경

2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

✪ 페이지 프레임에 적재:
참조 비트는 0

✪ 페이지 참조:
참조 비트는 1



2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

✪ 페이지 프레임에 적재:
참조 비트는 0

✪ 페이지 참조:
참조 비트는 1

페이지 B 참조

2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

✪ 페이지 프레임에 적재:
참조 비트는 0

✪ 페이지 참조:
참조 비트는 1

페이지 C 참조

2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음

- 페이지 프레임에 적재: 참조 비트는 0
- 페이지 참조: 참조 비트는 1

페이지 C 참조

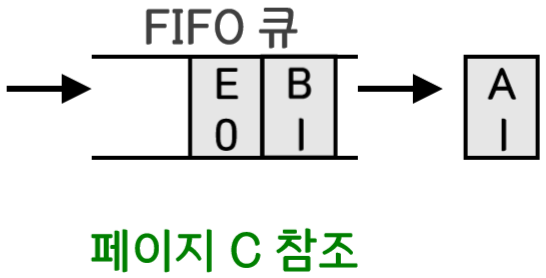
2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

✪ 페이지 프레임에 적재:
참조 비트는 0

✪ 페이지 참조:
참조 비트는 1

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음



2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음

- 페이지 프레임에 적재: 참조 비트는 0
- 페이지 참조: 참조 비트는 1

페이지 C 참조

2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음

- 페이지 프레임에 적재: 참조 비트는 0
- 페이지 참조: 참조 비트는 1

페이지 C 참조

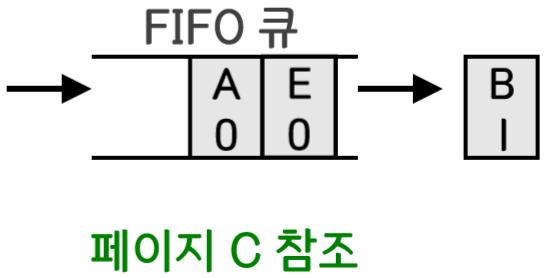
2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

✪ 페이지 프레임에 적재:
참조 비트는 0

✪ 페이지 참조:
참조 비트는 1

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음



2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

✪ 페이지 프레임에 적재:
참조 비트는 0

✪ 페이지 참조:
참조 비트는 1

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음

2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음

- 페이지 프레임에 적재: 참조 비트는 0
- 페이지 참조: 참조 비트는 1

페이지 C 참조

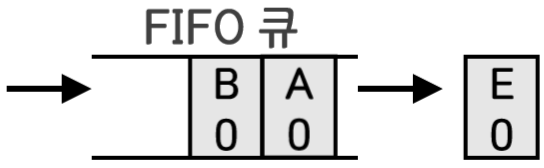
2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음

✪ 페이지 프레임에 적재:
참조 비트는 0

✪ 페이지 참조:
참조 비트는 1



페이지 C 참조

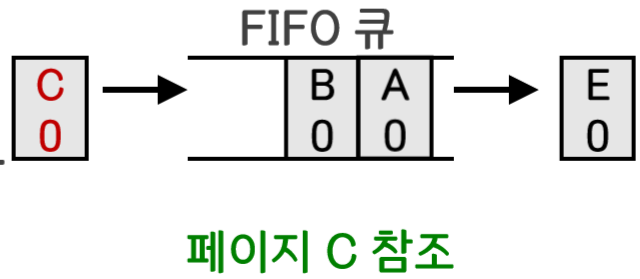
2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

✪ 페이지 프레임에 적재:
참조 비트는 0

✪ 페이지 참조:
참조 비트는 1

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음

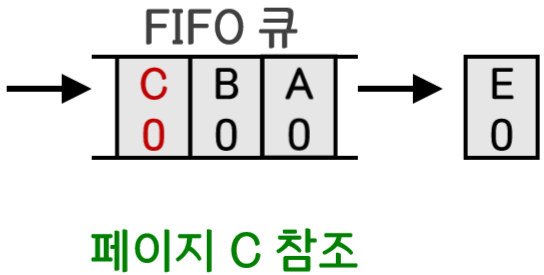


2차 기회 (second chance) 페이지 교체기법

- FIFO 페이지 교체기법과 참조 여부에 따른 우선순위를 고려하여 적합한 페이지를 교체
- 구현: FIFO 큐와 참조 비트 이용

- 교체 대상 선택 방법
  1) 큐의 선두를 꺼내 참조 비트 조사
  2) 참조 비트가 0이면 교체 대상으로 선택
  3) 참조 비트가 1이면 0으로 바꿔 큐의 뒤에 넣음

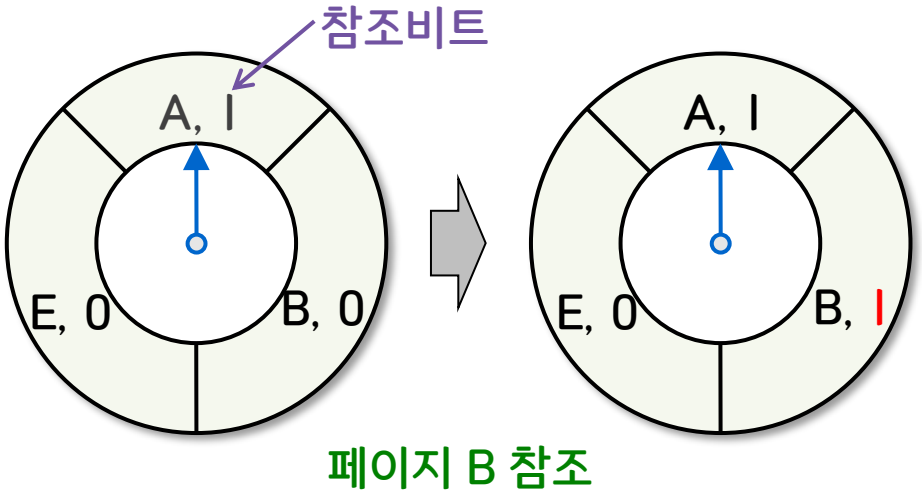
- 페이지 프레임에 적재: 참조 비트는 0
- 페이지 참조: 참조 비트는 1



페이지 C 참조

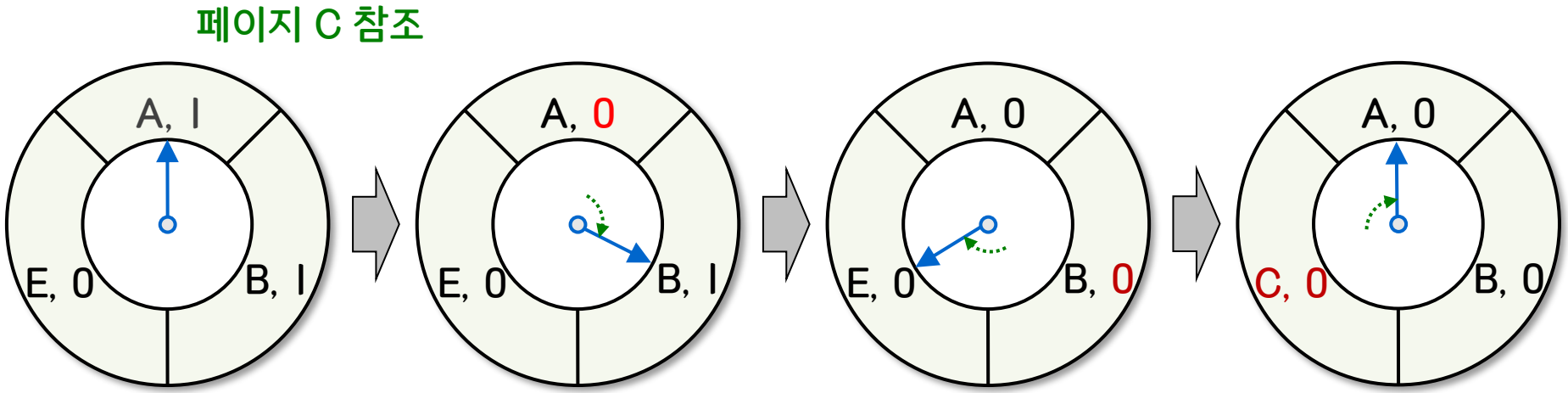
클럭 (clock) 페이지 교체기법

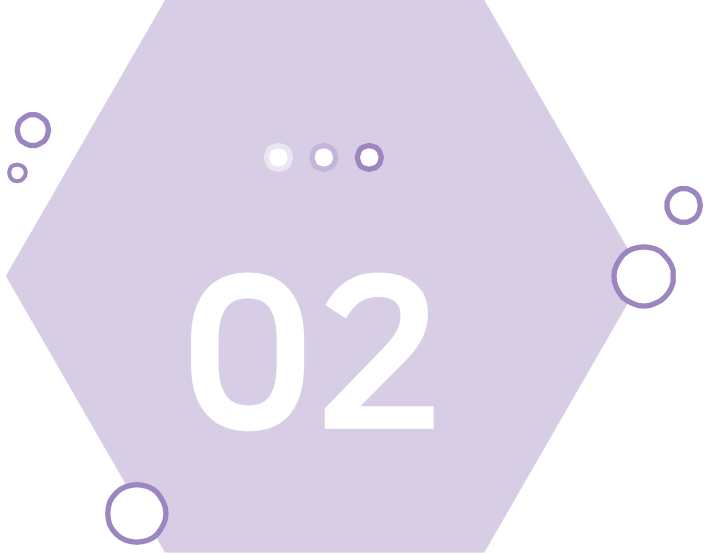
- 2차 기회 페이지 교체를 원형 큐를 이용하여 구현한 것
- 교체가 필요한 경우 큐에서의 삭제 및 삽입 대신 포인터 이동으로 간단히 구현



클럭 (clock) 페이지 교체기법

- 2차 기회 페이지 교체를 원형 큐를 이용하여 구현한 것
- 교체가 필요한 경우 큐에서의 삭제 및 삽입 대신 포인터 이동으로 간단히 구현





프로세스별 페이지 집합 관리

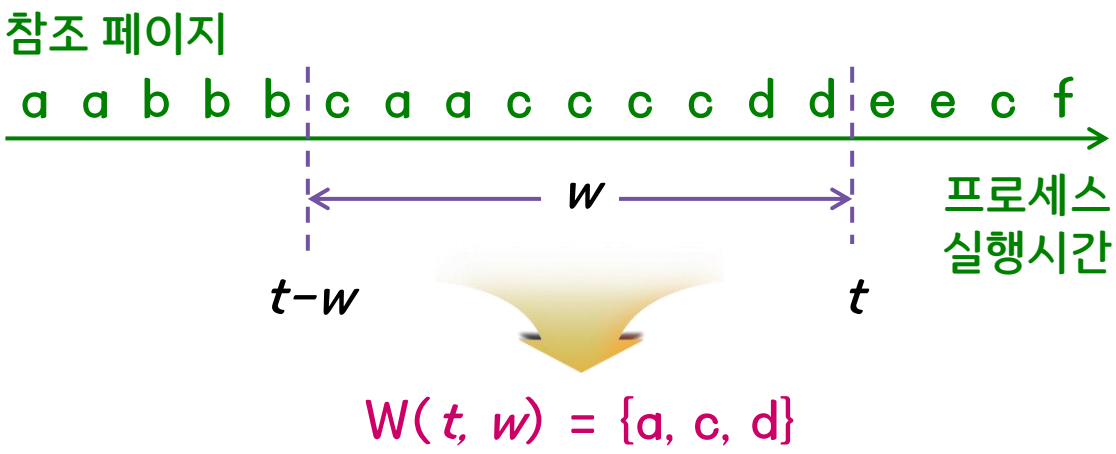
프로세스별 페이지 집합 관리

- **프로세스별 페이지 집합**
  - 프로세스마다 페이지 프레임에 적재된 페이지들의 집합
- **프로세스별 페이지 집합의 크기가 적은 경우**
  - 메모리에 적재할 수 있는 프로세스의 수 많아짐 → 시스템 처리량 증대
  - 각 프로세스별 페이지 부재 많아짐 → 성능 저하
- **페이지 집합 관리 알고리즘**
  - 워킹세트 알고리즘
  - PFF 알고리즘

워킹세트 알고리즘

워킹세트(working set)

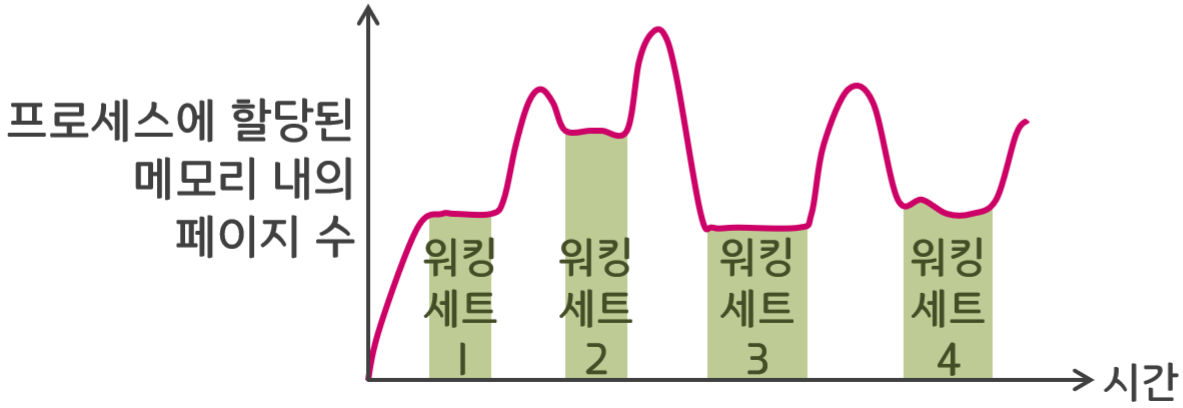
- 하나의 프로세스가 자주 참조하는 페이지의 집합
- 워킹세트 W(t, w): 시간 $t-w$로부터 시간 t까지의 프로세스 시간 간격 동안 참조된 페이지의 집합



- 프로세스 시간: 그 프로세스가 CPU를 점유하고 있는 시간
- t: 현재 시간
- w: 워킹세트 윈도 크기

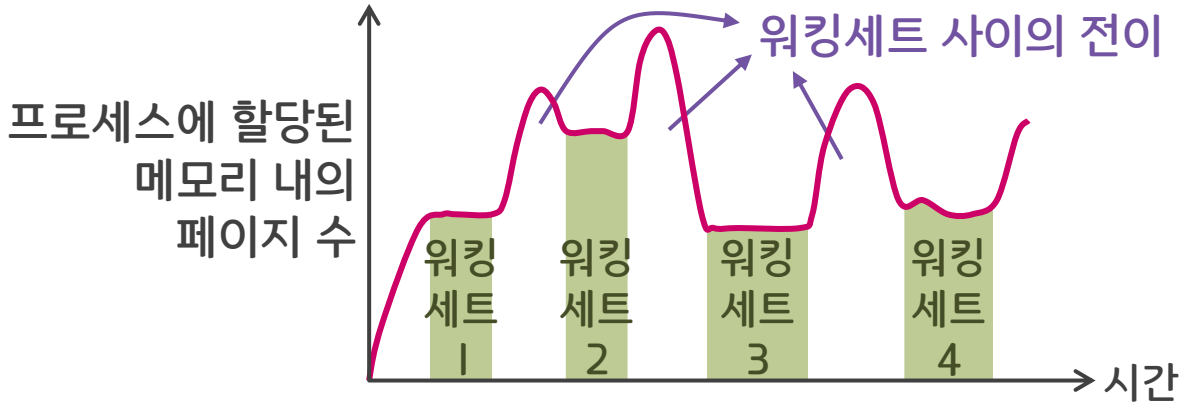
워킹세트 알고리즘

- 데닝(Denning)이 제안
- 페이지 부재 비율을 감소시키기 위한 방법
- 원칙: 실행 중인 프로그램의 워킹세트를 메모리에 유지(국부성)
- 프로세스가 실행됨에 따라 워킹세트의 크기는 변함



워킹세트 알고리즘

- 데닝(Denning)이 제안
- 페이지 부재 비율을 감소시키기 위한 방법
- 원칙: 실행 중인 프로그램의 워킹세트를 메모리에 유지(국부성)
- 프로세스가 실행됨에 따라 워킹세트의 크기는 변함



워킹세트 알고리즘

- 운영체제는 충분한 여분의 프레임이 존재하면
  새로운 프로세스를 실행시킴
- 반대로 프레임이 부족해지면
  가장 우선순위가 낮은 프로세스를 일시적으로 중지시킴
- 워킹세트가 메모리에 유지되지 못하는 경우
  - 쓰래싱 유발 가능

> **쓰래싱(thrashing)**
>
> 페이지 부재가 비정상적으로 많이 발생하여
> 프로세스 처리보다 페이지 교체에 너무 많은 시간을
> 소비하여 시스템 처리량이 급감하는 현상

워킹세트 알고리즘

- 문제점
  - 과거를 통해 미래를 예측하는 것이 정확하지 않음
  - 워킹세트를 정확히 알아내고 이를 계속적으로 업데이트하는 것이 현실적으로 어려움
  - 워킹세트 윈도의 크기 w의 최적 값을 알기 어려우며 이 역시 변화할 수 있음

PFF (Page Fault Frequency) 알고리즘

- 프로세스의 상주 페이지 세트를 변경하며 관리
- 페이지 부재가 발생할 때 빈도를 계산하여 상한과 하한을 벗어나는 경우에만 변경
- 페이지 부재 빈도
  - 두 페이지 부재가 일어난 사이의 시간의 역수
- 운영체제는 프레임의 여분 상황에 따라 새로운 프로세스 추가 및 중지 결정

✪ 상주 페이지 세트

프로세스가 페이지 부재 때문에 멈추게 되는 빈도에 기초한 페이지 세트

강의를 마쳤습니다.

다음시간에는

11강. 장치 관리