

선수 지식 - 자료구조

연결 리스트

연결 리스트 | 다양한 알고리즘의 기본이 되는 자료구조 이해하기

강사 나동빈

선수 지식 - 자료구조

연결 리스트

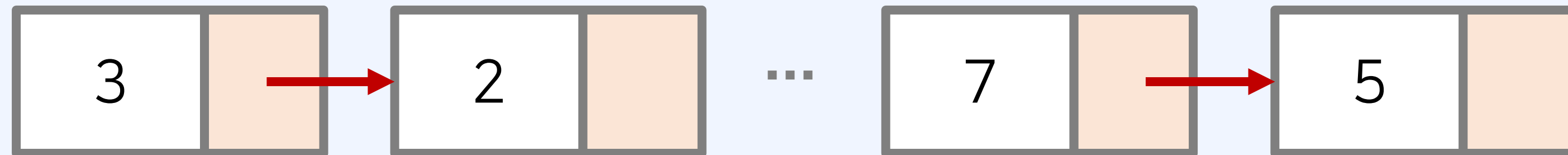
연결 리스트(Linked List)

- 연결 리스트는 각 노드가 한 줄로 연결되어 있는 자료 구조다.
- 각 노드는 (데이터, 포인터) 형태를 가진다.
- 포인터: 다음 노드의 메모리 주소를 가리키는 목적으로 사용된다.



연결 리스트(Linked List)

- 연결 리스트는 각 노드가 한 줄로 연결되어 있는 자료 구조다.
- 각 노드는 (데이터, 포인터) 형태를 가진다.
- 연결성: 각 노드의 포인터는 다음 혹은 이전 노드를 가리킨다.



연결 리스트(Linked List)

- 연결 리스트를 이용하면 다양한 자료구조를 구현할 수 있다.
- 예시) 스택, 큐 등
- Python은 연결 리스트를 활용하는 자료구조를 제공한다.
- 그래서 연결 리스트를 실제 구현해야 하는 경우는 적지만, 그 원리에 대해서 이해해 보자.

연결 리스트(Linked List) vs. 배열(Array)

- 연결 리스트와 배열(array)을 비교하여 장단점을 이해할 필요가 있다.
- 특정 위치의 데이터를 삭제할 때, 일반적인 배열에서는 $O(N)$ 만큼의 시간이 소요된다.
- 하지만, 연결 리스트를 이용하면 단순히 연결만 끊어주면 된다.
- 따라서 삭제할 위치를 정확히 알고 있는 경우 $O(1)$ 의 시간이 소요된다.

- 배열에 새로운 원소를 삽입할 때, 최악의 경우 시간 복잡도를 계산하여라.
- 예시) 아래 배열에서 인덱스 3에 원소 "59"를 삽입할 경우

삽입할 위치



인덱스	0	1	2	3	4	5	6	7	8
값	25	75	83	24	72	55	17	42	

- 배열에 새로운 원소를 삽입할 때, 최악의 경우 시간 복잡도를 계산하여라.
- 예시) 아래 배열에서 인덱스 3에 원소 "59"를 삽입할 경우

한 칸씩 밀기

인덱스	0	1	2	3	4	5	6	7	8
값	25	75	83		24	72	55	17	42

- 배열에 새로운 원소를 삽입할 때, 최악의 경우 시간 복잡도를 계산하여라.
- 예시) 아래 배열에서 인덱스 3에 원소 "59"를 삽입할 경우

삽입 완료



인덱스	0	1	2	3	4	5	6	7	8
값	25	75	83	59	24	72	55	17	42

- 배열에 존재하는 원소를 삭제할 때, 최악의 경우 시간 복잡도를 계산하여라.
- 예시) 아래 배열에서 인덱스 3에 해당하는 원소를 삭제하는 경우

삭제할 원소



인덱스	0	1	2	3	4	5	6	7	8
값	25	75	83	59	24	72	55	17	42

- 배열에 존재하는 원소를 삭제할 때, 최악의 경우 시간 복잡도를 계산하여라.
- 예시) 아래 배열에서 인덱스 3에 해당하는 원소를 삭제하는 경우

삭제 완료



인덱스	0	1	2	3	4	5	6	7	8
값	25	75	83		24	72	55	17	42

- 배열에 존재하는 원소를 삭제할 때, 최악의 경우 시간 복잡도를 계산하여라.
- 예시) 아래 배열에서 인덱스 3에 해당하는 원소를 삭제하는 경우


한 칸씩 당기기



인덱스	0	1	2	3	4	5	6	7	8
값	25	75	83	24	72	55	17	42	

- 배열에 존재하는 원소를 삭제할 때, 최악의 경우 시간 복잡도를 계산하여라.
- 예시) 아래 배열에서 인덱스 3에 해당하는 원소를 삭제하는 경우

한 칸씩 당기기

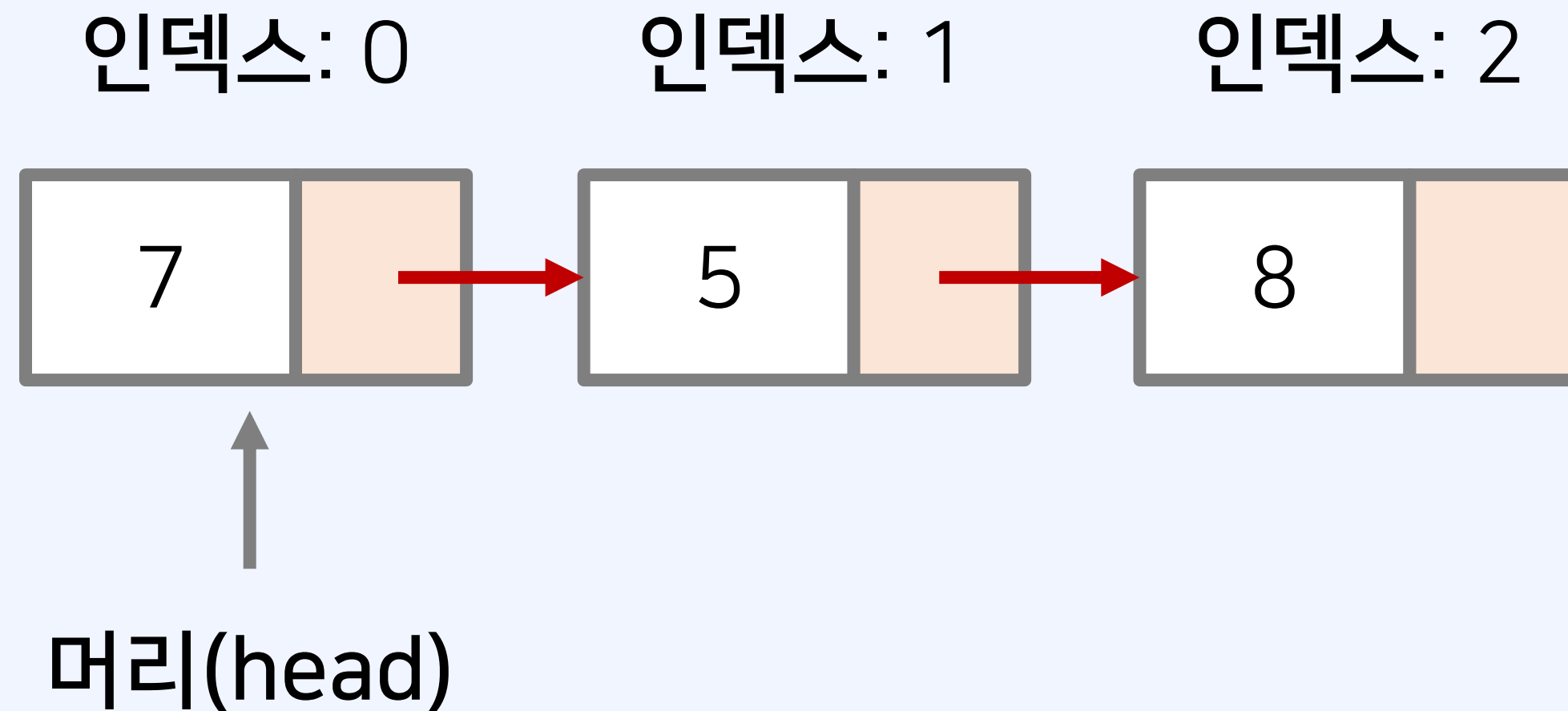


인덱스	0	1	2	3	4	5	6	7	8
값	25	75	83	24	72	55	17	42	

- 따라서, 최악의 경우 시간 복잡도는 $O(N)$ 이다.

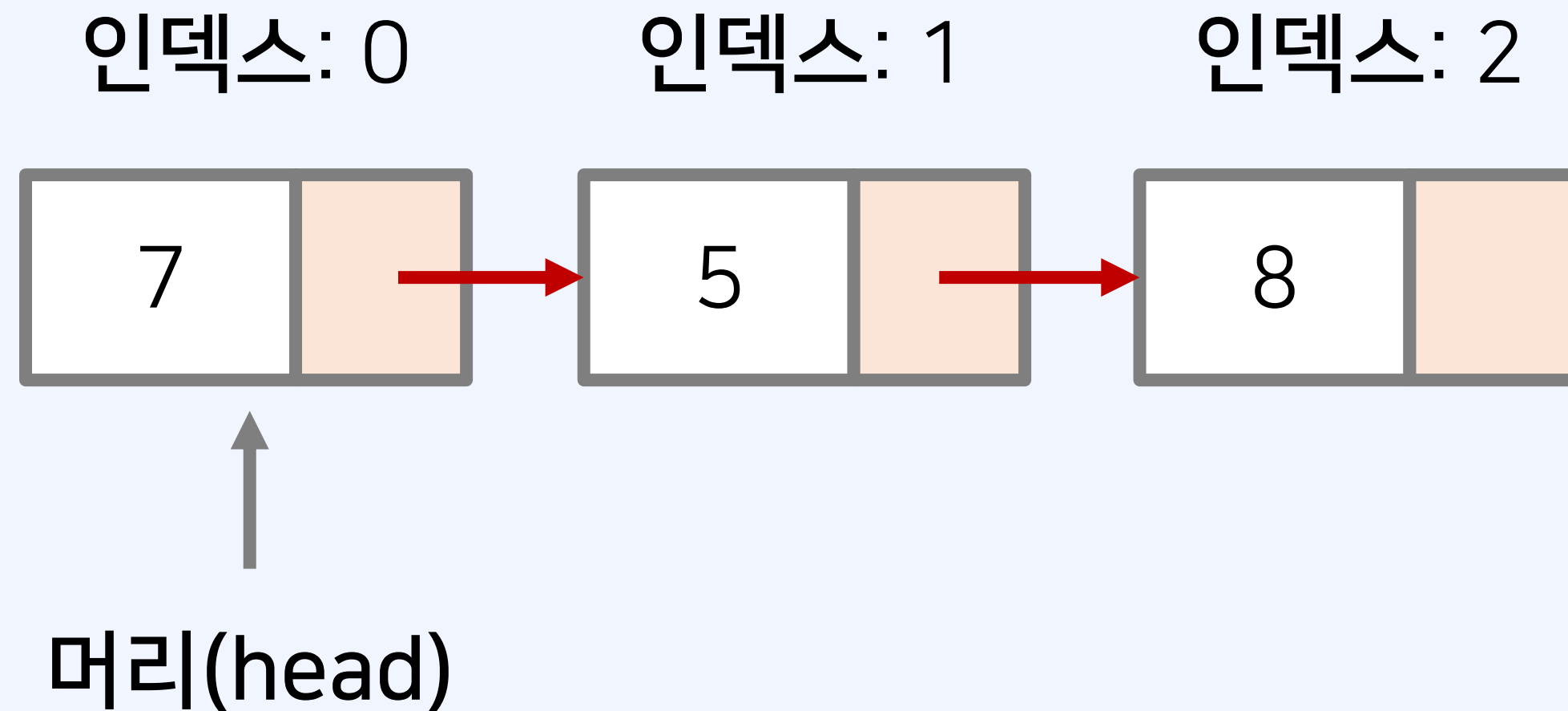
연결 리스트(Linked List): 삽입(Insert) 연산

- 삽입할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삽입할 수 있다.

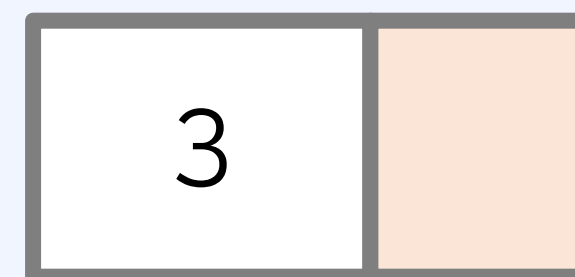


연결 리스트(Linked List): 삽입(Insert) 연산

- 삽입할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삽입할 수 있다.

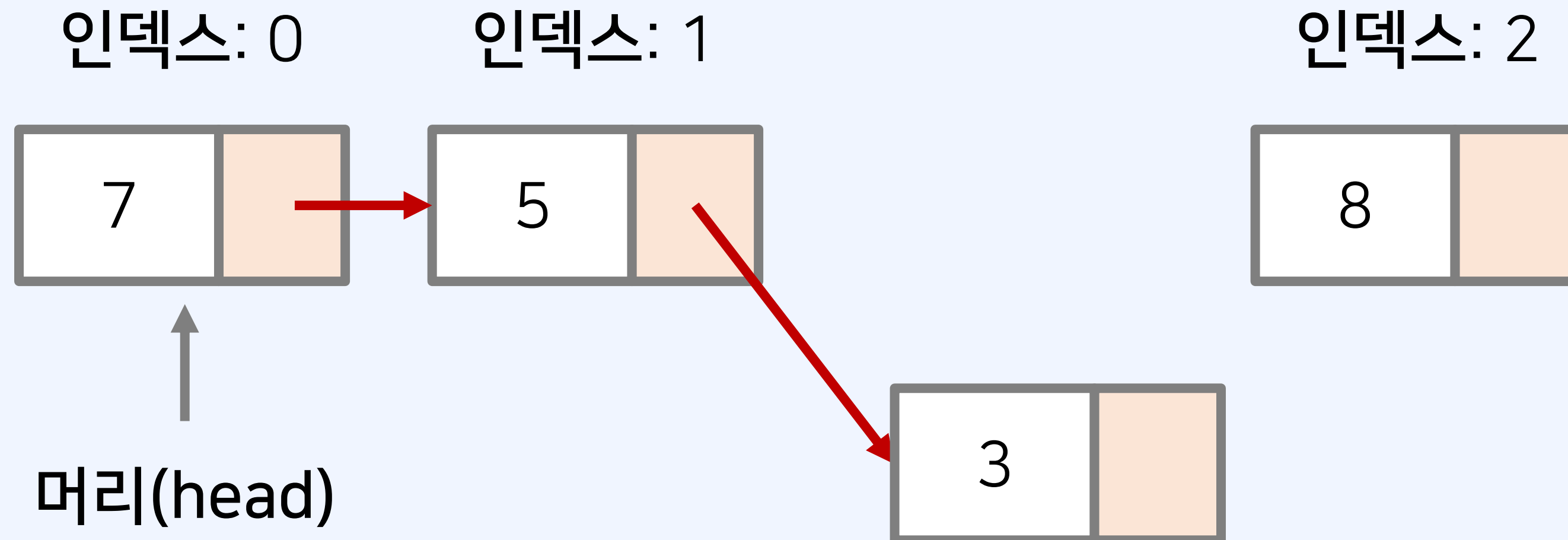


- 인덱스 2의 위치에 원소를 삽입



연결 리스트(Linked List): 삽입(Insert) 연산

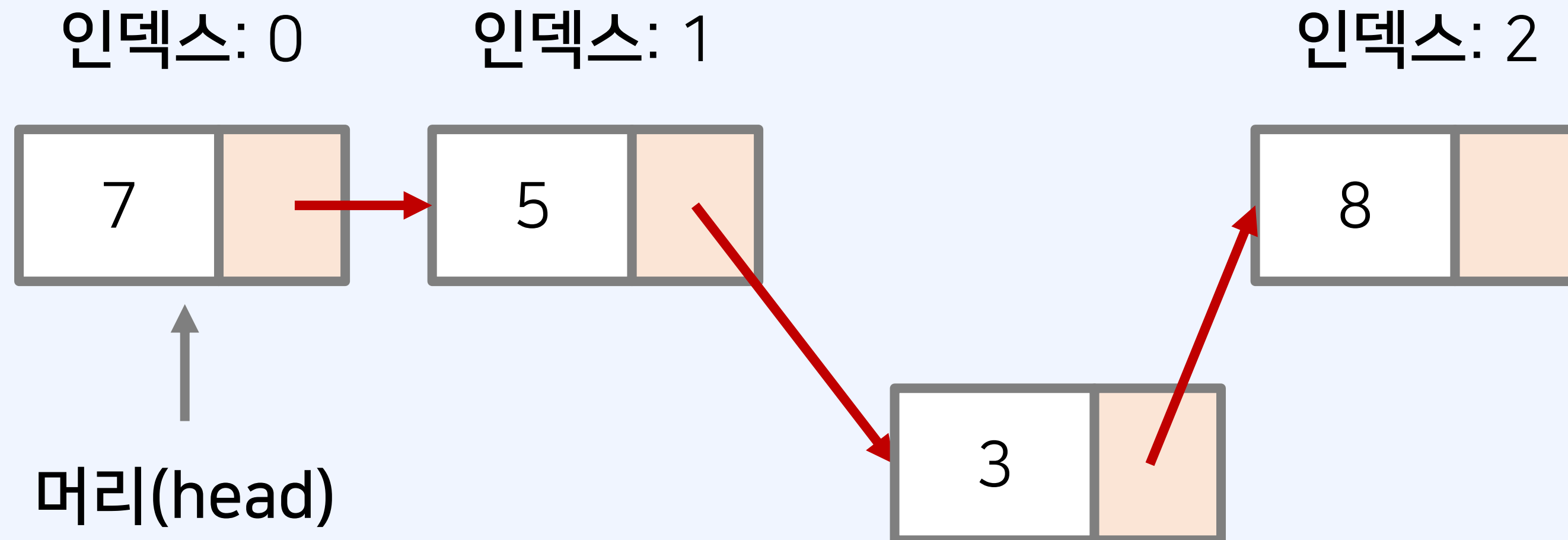
- 삽입할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삽입할 수 있다.



- 인덱스 2의 위치에 원소를 삽입

연결 리스트(Linked List): 삽입(Insert) 연산

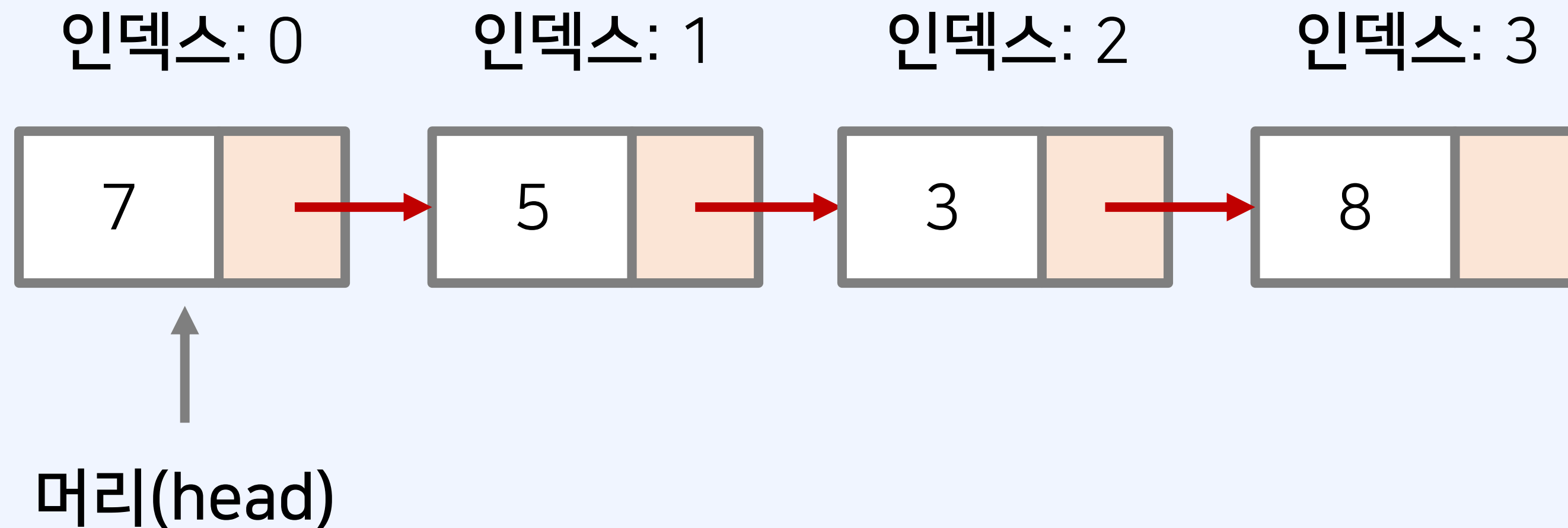
- 삽입할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삽입할 수 있다.



- 인덱스 2의 위치에 원소를 삽입

연결 리스트(Linked List): 삽입(Insert) 연산

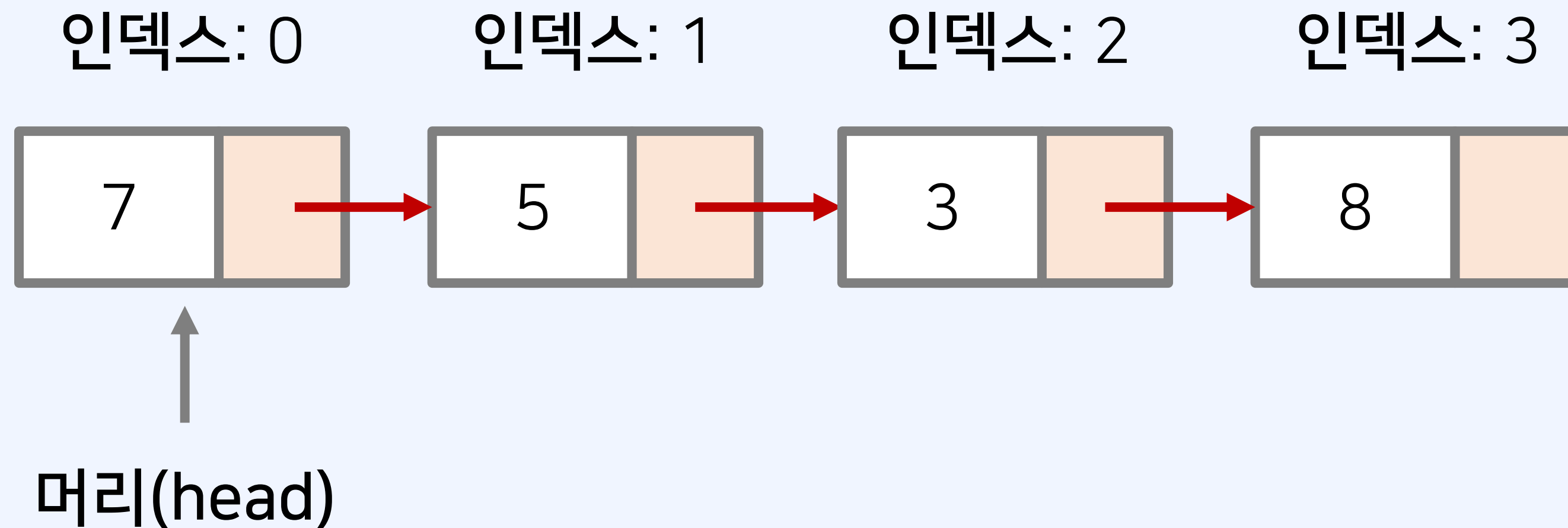
- 삽입할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삽입할 수 있다.



- 인덱스 2의 위치에 원소를 삽입

연결 리스트(Linked List): 삭제(Remove) 연산

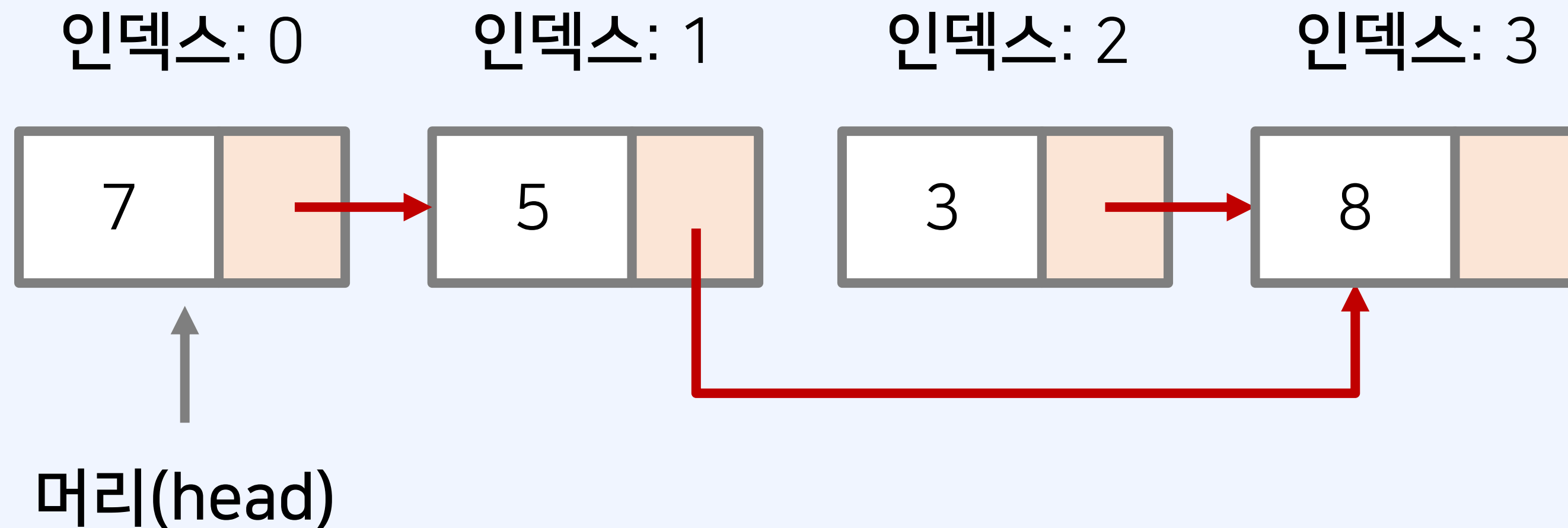
- 삭제할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삭제할 수 있다.



- 인덱스 2의 위치의 원소를 삭제

연결 리스트(Linked List): 삭제(Remove) 연산

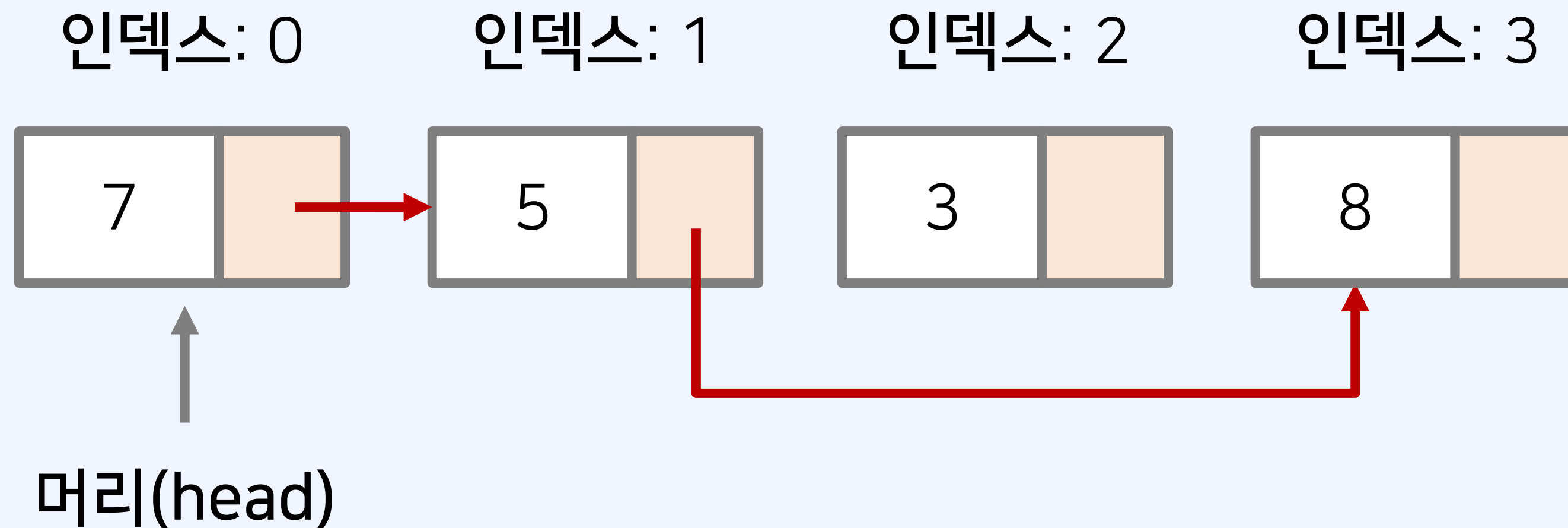
- 삭제할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삭제할 수 있다.



- 인덱스 2의 위치의 원소를 삭제

연결 리스트(Linked List): 삭제(Remove) 연산

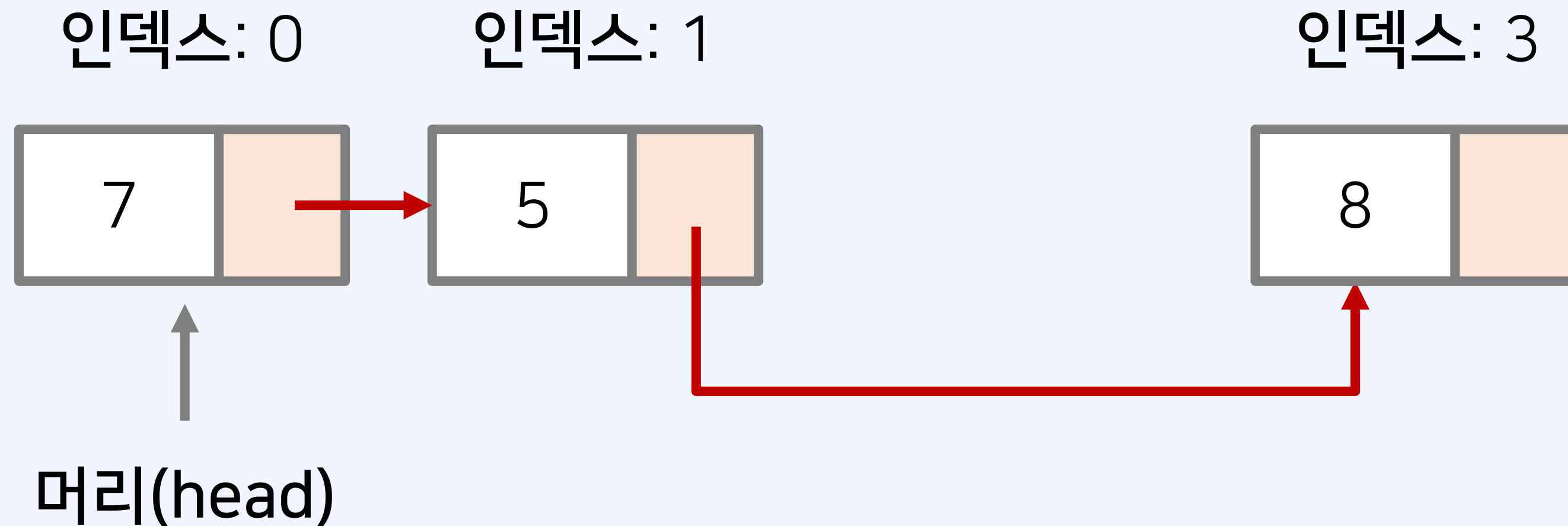
- 삭제할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삭제할 수 있다.



- 인덱스 2의 위치의 원소를 삭제

연결 리스트(Linked List): 삭제(Remove) 연산

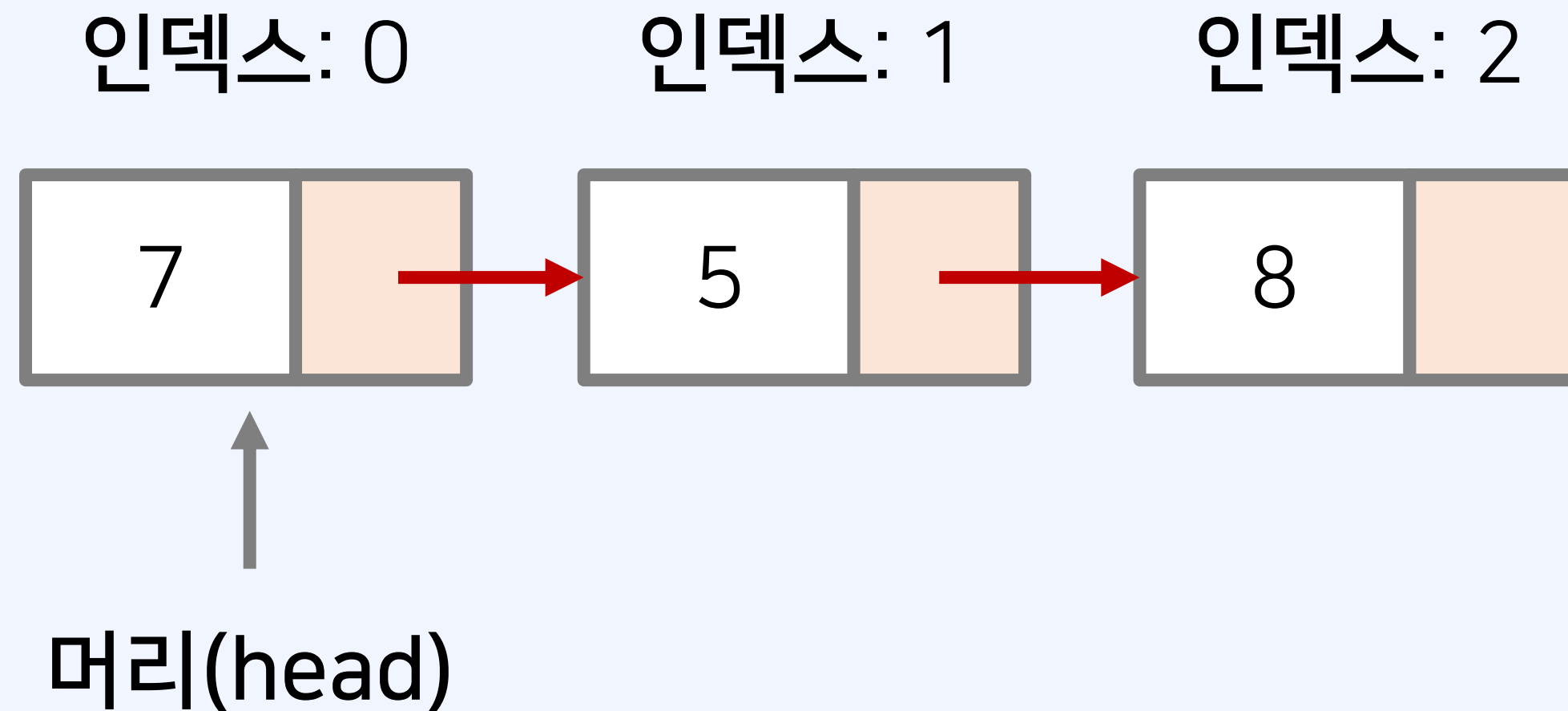
- 삭제할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삭제할 수 있다.



- 인덱스 2의 위치의 원소를 삭제

연결 리스트(Linked List): 삭제(Remove) 연산

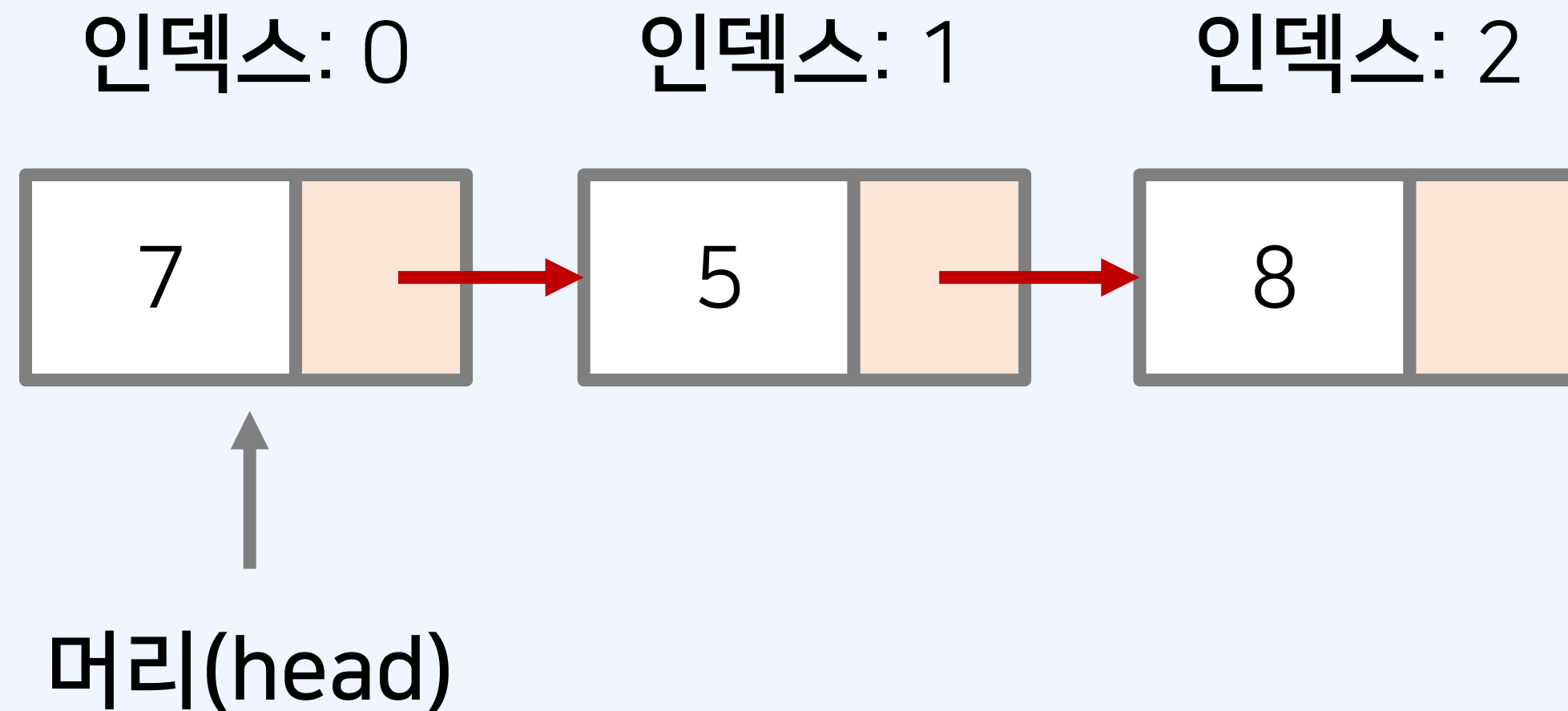
- 삭제할 위치를 알고 있다면, 물리적인 위치를 한 칸씩 옮기지 않아도 삭제할 수 있다.



- 인덱스 2의 위치의 원소를 삭제

연결 리스트(Linked List): 뒤에 붙이기(Append) 연산

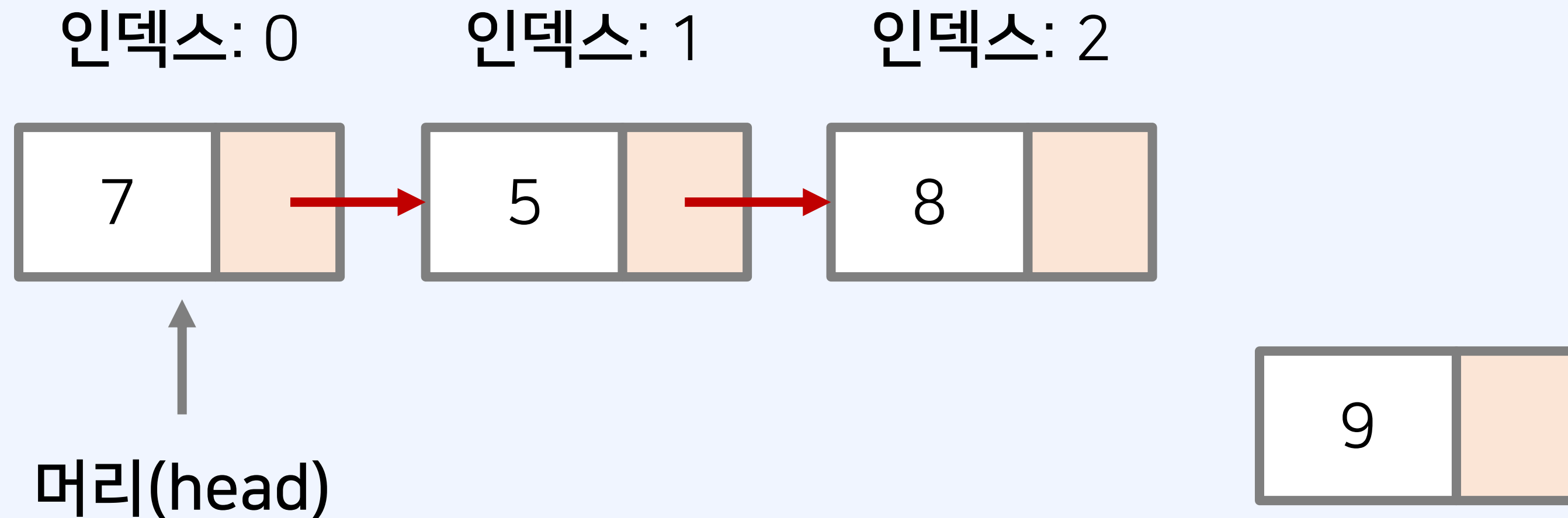
- 뒤에 붙일 때는 마지막 노드의 다음 위치에 원소를 넣으면 된다.



- 마지막 위치에 새로운 원소를 추가

연결 리스트(Linked List): 뒤에 붙이기(Append) 연산

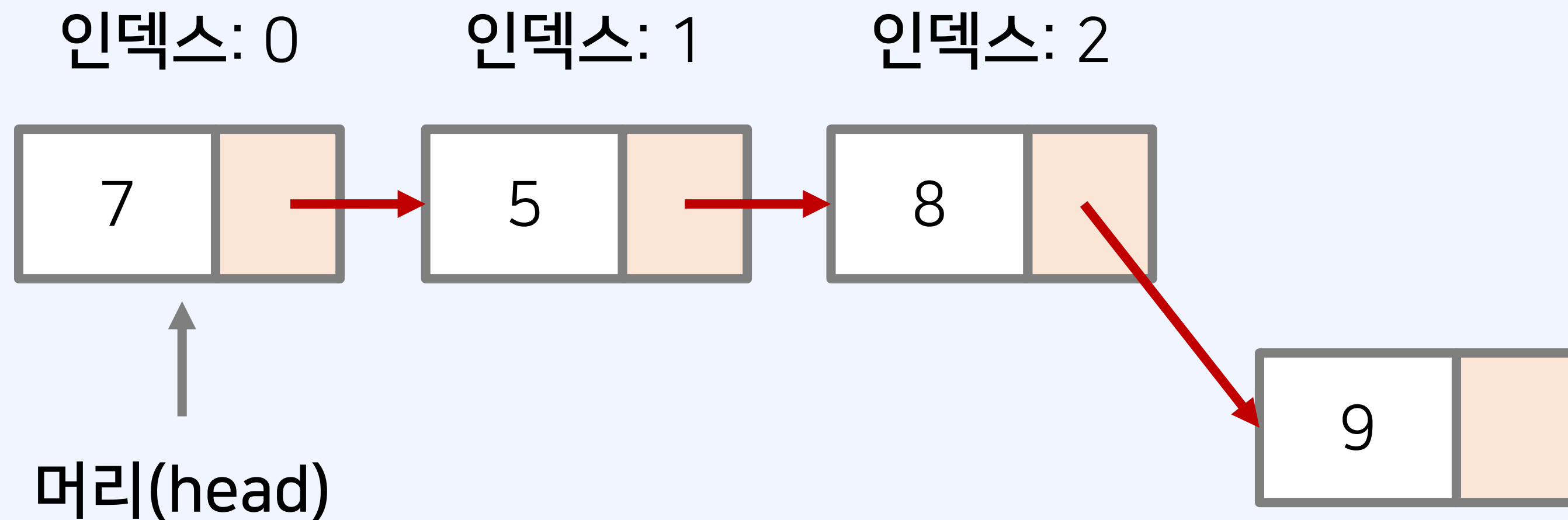
- 뒤에 붙일 때는 마지막 노드의 다음 위치에 원소를 넣으면 된다.



- 마지막 위치에 새로운 원소를 추가

연결 리스트(Linked List): 뒤에 붙이기(Append) 연산

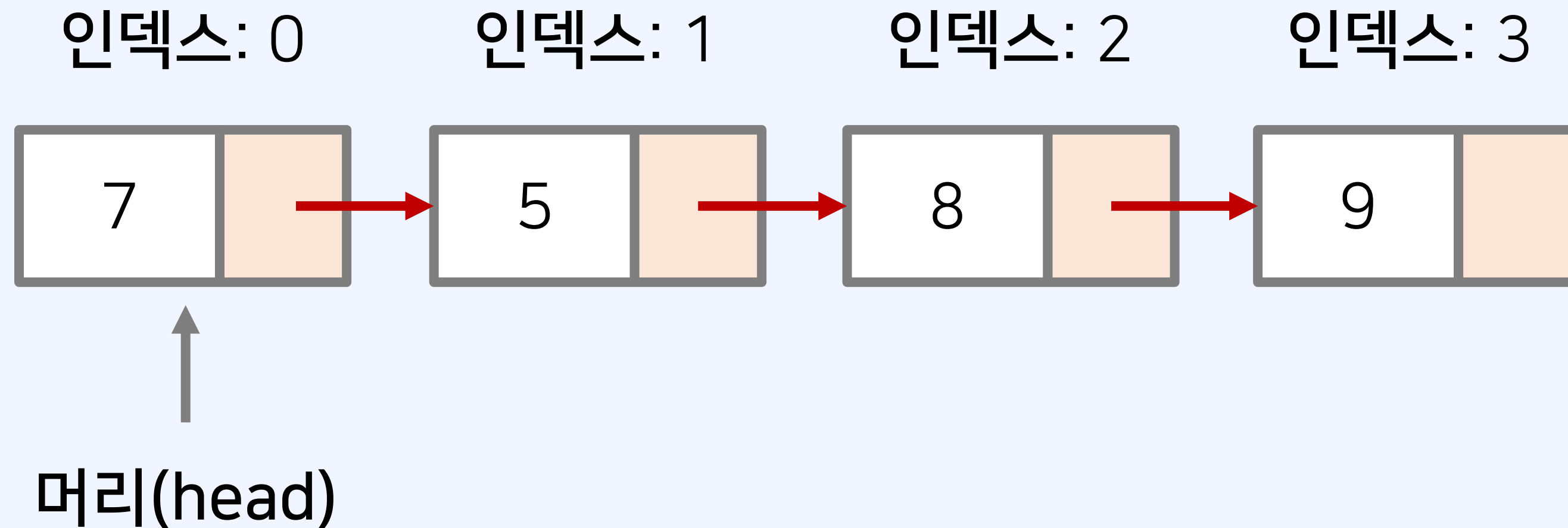
- 뒤에 붙일 때는 마지막 노드의 다음 위치에 원소를 넣으면 된다.



- 마지막 위치에 새로운 원소를 추가

연결 리스트(Linked List): 뒤에 붙이기(Append) 연산

- 뒤에 붙일 때는 마지막 노드의 다음 위치에 원소를 넣으면 된다.



- 마지막 위치에 새로운 원소를 추가