0X04 연결리스트

상태	완료
담당자	㉮ 서연

연결리스트란?

> 💡 메모리 상에서 불연속적으로 할당되어지지만, 내 다음 위치의 노드의 주소값을 함께 저장하는 선형 자료구조

성질?

1. K번째 원소를 검색하기 위해 O(K)의 시간이 소요됨

→ 연결리스트는 반드시 가장 앞 노드에서 시작해서 해당 노드를 찾을 때까지 순회를 계속하기 때문.

- 최악의 경우 가장 마지막 노드를 찾게 되고, 가장 Best 경우는 노드의 처음 원소 값을 찾게 되는 것임.

2. 임의의 원소 위치에 추가 및 삭제에 O(1)의 시간이 소요됨.

→ 배열과 가장 큰 차이점을 가지는 부분이기도함.

- 앞에서 설명했듯이 메모리 상에서 불연속적으로 위치해있고 주소값으로 내 다음 노드를 저장하기 때문.

3. cache hit rate가 낮지만 할당이 쉬움.

→ 메모리에 연속적으로 할당 받는 것이 아니기 때문

배열과의 비교

	배열	연결 리스트
k번째 원소의 접근	O(1)	O(k)
임의 위치에 원소 추가/제거	O(N)	O(1)
메모리 상의 배치	연속	불연속
추가적으로 필요한 공간 (Overhead)	-	O(N)

종류?

단일 연결 리스트, Singly Linked List

💡 내 다음 노드의 주소 값 만을 가지고 연결되어 있는 리스트

이중 연결 리스트, Doubly Linked List

💡 나의 이전 주소 값과 다음 주소 값을 저장하여 연결되어있는 리스트

1. 서로 양방향으로 연결되어있음.

원형 연결리스트, Cirular Linked List

💡 리스트의 가장 처음 노드와 가장 뒤의 원소가 연결되어있는 리스트

연결리스트의 연산 (여기서 양방향 리스트라고 가정함)

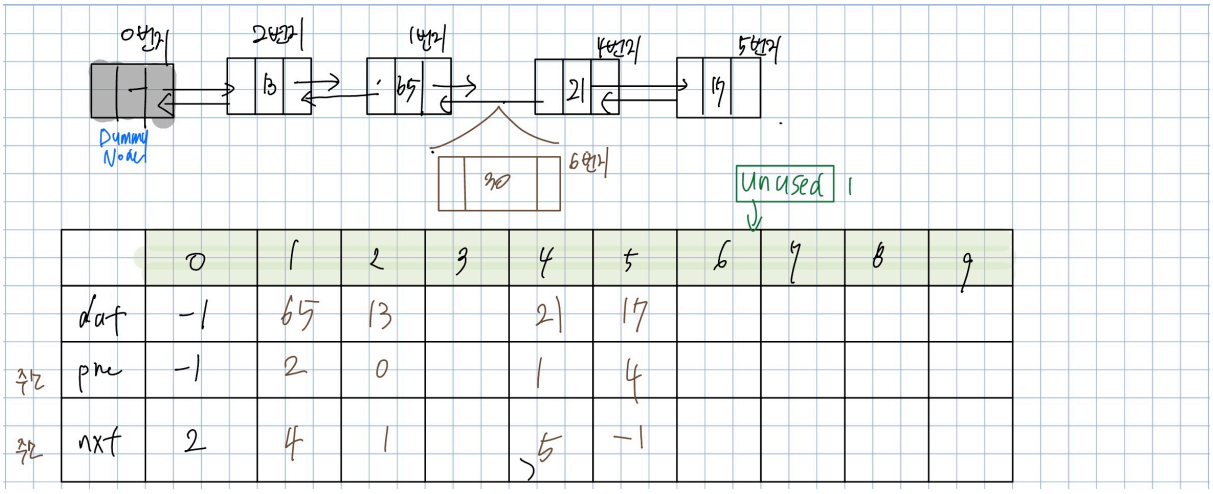
연결리스트의 삽입

1. 받아온 위치에 data를 가진 노드를 삽입하는 메소드

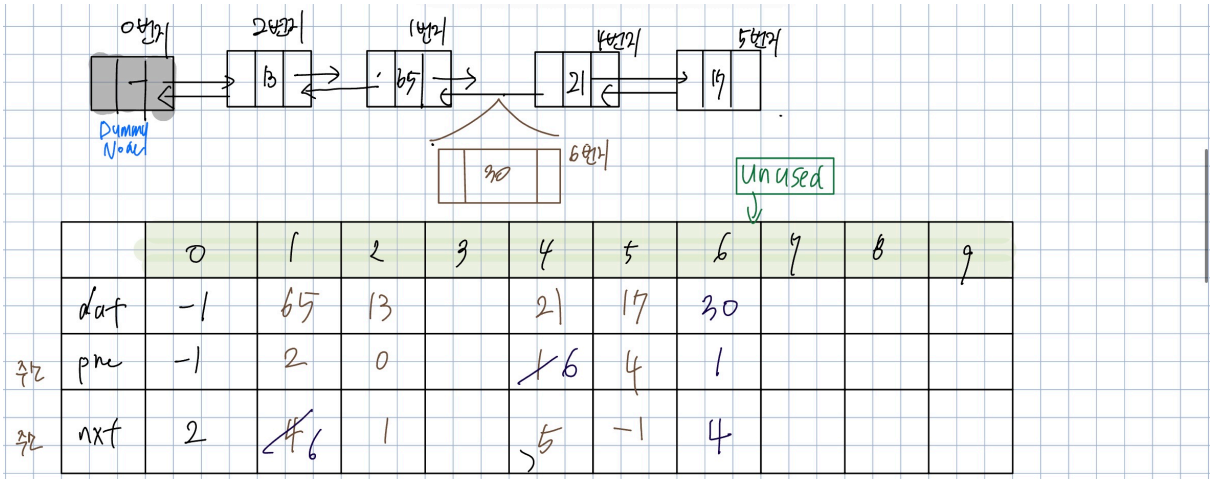
2. 삽입 연산 알고리즘

1. data를 가진 새 노드를 생성함.
2. addr 위치에 삽입하기 위해 원래 해당 자리에 있던 노드를 이전 노드로 만듦.
3. 원래 addr 위치가 가리키고 있던 next를 새로운 노드의 next로 갖게 함.
4. 삽입할 위치의 next 삽입할 위치의 다음원소 pre를 새원소를 설정함.
5. unused를 1로 증가함.

> 단 여기서 말하는 unused값은 새 원소가 들어갈 자리임.
> 밑에 그림에서는 6번지의 내용을 채우지는 않았고, 잘 삭제가 되면 아래와 같은 논리적 결과를 갖게 됨.



30 삽입 전



30 삽입 후

```
public static void insert(int addr, int num) {
        // 1. 새 노드에 대한 정보 저장
        dat[unused] = num;
        pre[unused] = addr;
        nxt[unused] = nxt[addr];

        // 2. 원래 있던 노드들의 연결 정보 수정
        // addr의 다음 노드의 pre를 새 노드로 변경
        if (nxt[addr] != -1) {
            pre[nxt[addr]] = unused;
        }
        // addr의 nxt를 새 노드로 변경
        nxt[addr] = unused;

        // 3. 다음 사용 가능한 주소 증가
        unused++;
    }
```

연결리스트의 삭제

1. 삭제할 노드의 주소 값을 받아서 삭제하는 메소드
2. 삭제 연산 알고리즘

1. 삭제할 노드와 연결된 pre node의 next를 삭제할 노드의 next에 연결된 next node를 가리키도록 함.
2. 반대로 삭제할 노드의 next에 연결된 node의 pre에 삭제할 노드의 이전에 연결된 노드로 바꿔줌.

- 논리적으로 연결을 끊어서 리스트 상에 포함되지 않게 하여 삭제 처리함.

```
// addr은 이 위치의 노드를 '삭제'하라는 의미입니다.
    public static void erase(int addr) {
        // 삭제할 노드의 이전 노드와 다음 노드를 서로 연결
        nxt[pre[addr]] = nxt[addr];
        if (nxt[addr] != -1) {
```

```
        pre[nxt[addr]] = pre[addr];
    }
}
```