1. 연결리스트

**각**[**노드**](https://ko.wikipedia.org/wiki/%EB%85%B8%EB%93%9C)**가**[**데이터**](https://ko.wikipedia.org/wiki/%EB%8D%B0%EC%9D%B4%ED%84%B0)**와**[**포인터**](https://ko.wikipedia.org/wiki/%ED%8F%AC%EC%9D%B8%ED%84%B0_(%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D))**를 가지고 한 줄로 연결되어 있는 방식으로 데이터를 저장하는**[**자료 구조**](https://ko.wikipedia.org/wiki/%EC%9E%90%EB%A3%8C_%EA%B5%AC%EC%A1%B0)

2. 장단점

탐색 : O(n) 순차탐색 (1열종대)

삽입 : O(1)+

삭제 : O(1)+

\* (+)는 이전 노드의 레퍼런스가 주어진 경우 또는 첫 노드

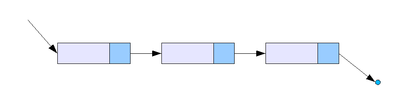
삽입 삭제 시 항목 이동 필요 X

레퍼런스(화살표)만 바꿔주면 된다.

(응용) 스택, 큐, 해싱의 체이닝(chaining), 트리

3. 종류

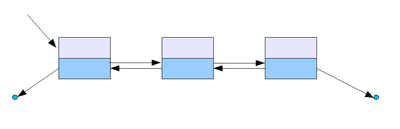
가. 단순연결리스트 : 동적 메모리할당을 이용해 노드들을 한 방향으로 리스트 연결



단점) 1. 삽입, 삭제 시 반드시 이전노드를 가리키는 레퍼런스를 추가로 알아야 함

2. 역방향 노드 탐색 불가

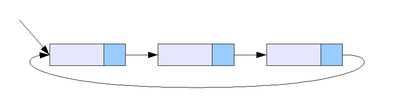
나. 이중연결리스트 : 각 노드가 두 개의 레퍼런스를 가지고 각각 이전, 다음 노드 가리킴



장점 : 단순연결리스트 단점 해결

단점 : 각 노드마다 이전, 이후 레퍼런스 저장해야함

다. 원형연결리스트 : 마지막 노드가 첫 노드와 연결된 단순연결리스트



장점) 1. 마지막 노드와 첫 노드방문 O(1)

2. none조건 검사 불필요 (empty만 none)

단점) 1. 반대방향 노드 방문 어려움

2. 무한루프 발생 유의

(응용) 턴제 게임, CPU시간분할, 이항힙, 피보나치합 (우선순위 큐)

4. 어디에 쓸까?