



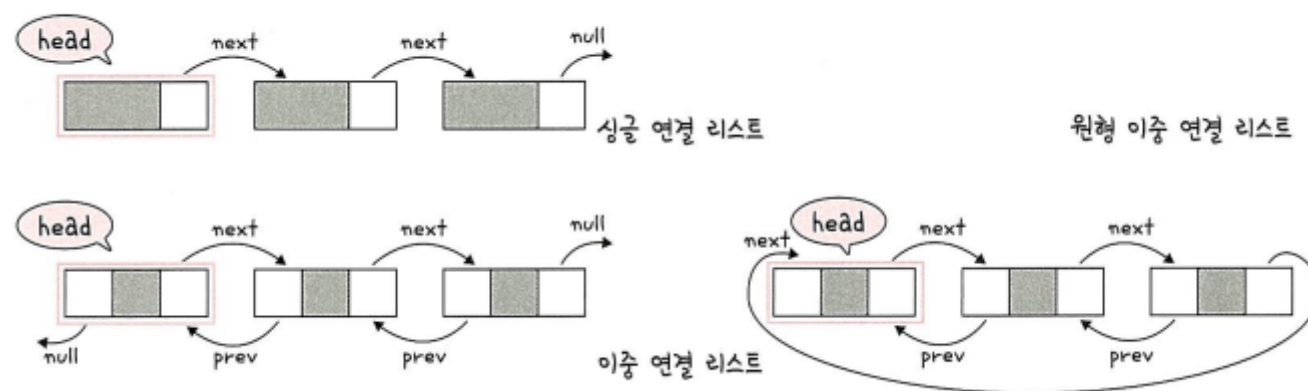
Section2. 선형 자료 구조

선형 자료구조란 요소가 일렬로 나열되어 있는 자료구조.

5.2.1 연결 리스트

- 데이터를 감싼 노드를 포인터로 연결해서 공간적인 효율성을 극대화한 자료구조.
- 삽입과 삭제가 $O(1)$.
- 탐색은 $O(n)$

▼ 그림 5-3 연결 리스트



연결리스트의 종류

- 싱글 연결리스트
 - next 포인터만 가짐
- 이중 연결리스트
 - next 포인터와 prev 포인터를 가짐
- 원형 이중 연결리스트
 - 이중 연결리스트와 같지만 마지막 노드의 next 포인터가 헤드 노드를 가리키는 것.

주요 함수(구현)

push_front() : 앞에서부터 요소를 넣음.

push_back() : 뒤에서부터 요소를 넣음.

insert() 중간에 요소 삽입.

pop_back() : 맨 뒤 요소를 제거 후 리턴

erase() : 특정 요소를 제거

find() : 요소 찾기

clear() : 배열 초기화

5.2.2 배열

- 같은 타입의 변수들로 이루어져있음
- 크기가 정해져있음.
- 인접한 메모리 위치에 있는 데이터를 모아놓은 집합.

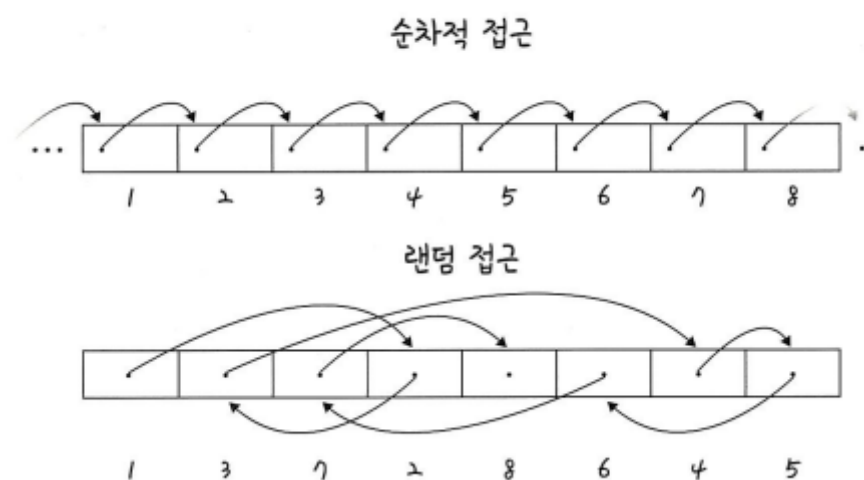
- 중복을 허용하고 순서가 있음.
- 탐색이 빠름 $O(1)$
 - 랜덤 접근 가능.

배열 vs 연결리스트

- 데이터 추가 삭제가 많이 일어난다면 연결리스트를 사용.
- 탐색을 많이한다면 배열을 사용.

랜덤 접근과 순차적 접근

▼ 그림 5-4 랜덤 접근과 순차적 접근




- 랜덤접근
 - 직접 접근이라고도 함.
 - 동일한 시간에 배열과 같은 순차적인 데이터가 있을 때, 임의의 인덱스에 해당하는 데이터에 접근할 수 있는 기능.
- 순차적 접근
 - 데이터를 저장된 순서대로 검색해야 함.

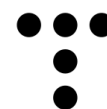
5.2.3 벡터

- 동적으로 요소를 할당할 수 있는 동적 배열
- 컴파일 시점에 요소의 개수를 모른다면 벡터를 써야 함.
- 중복을 허용
- 순서가 있고 랜덤접근 가능.
- 탐색과 맨 뒤의 요소를 삭제하거나 삽입하는데 $O(1)$ 걸림.
- 맨 뒤나 맨 앞이 아닌 요소 삽입 삭제시에는 $O(n)$
- `push_back()`는 $O(1)$
 - 벡터의 크기가 증가되는 시간 복잡도가 amortized 복잡도.
 - 즉, 상수 시간 복잡도 $O(1)$ 과 유사한 시간복잡도 가지기 때문.

할부 시간 복잡도(amortized time complexity)

컴퓨터 프로그램의 시간 복잡도를 계산할 때, 가장 많이 사용되는 것은 최악 시간 복잡도(worst-case time complexity)다. 어떤 입력이 들어왔을 때, 가장 오래 걸리는 시간을 추정하는 것이다. 그 다음으로 많이 사용되는 것은 평균 시간 복잡도(average-case time complexity)다. 시간을 가장 많이 끄는 입력이 천 번에 한 번꼴로만 발생한다면

 <https://woogyun.tistory.com/245>

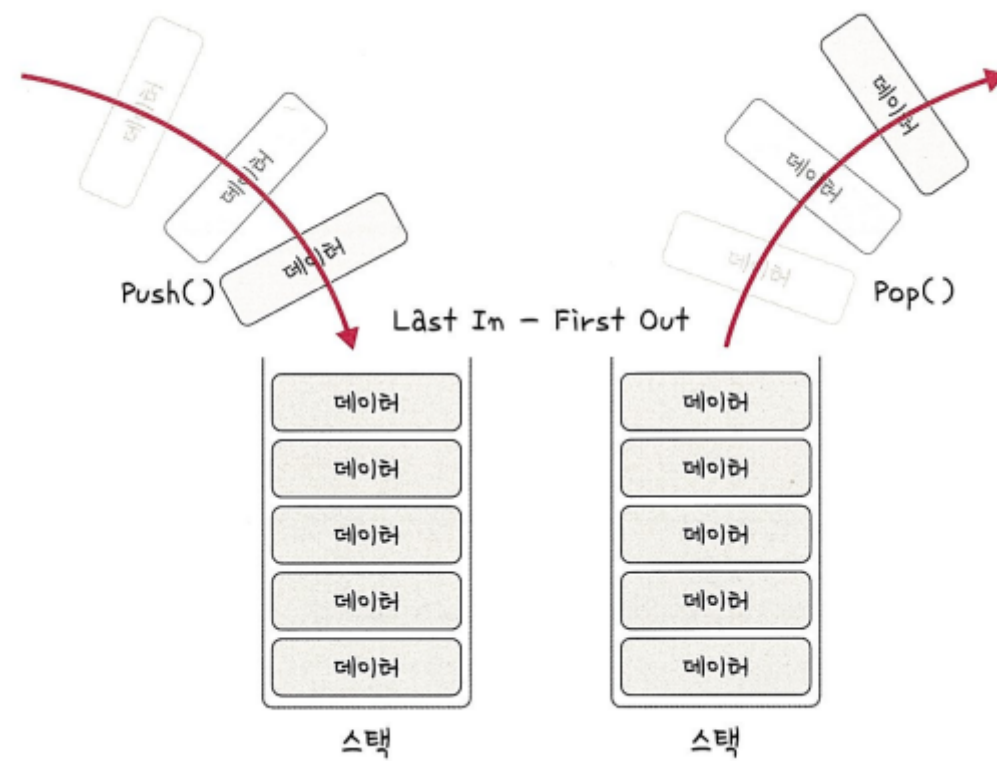


5.2.4 스택

스택이란?

- 가장 마지막으로 들어간 데이터가 가장 첫번째로 나오는 성질. (LIFO)
- 재귀적인 함수, 알고리즘에 사용되며 웹 브라우저 방문 기록 등에 쓰임.
- 삽입과 삭제 $O(1)$
- 탐색 $O(n)$

▼ 그림 5-7 스택



큐

- 먼저 집어넣은 데이터가 먼저 나오는 성질 (FIFO)
- 나중에 집어넣은 데이터가 먼저 나오는 스택과는 반대되는 개념.
- 삽입 및 삭제에 $O(1)$
- 탐색 $O(n)$
- CPU 작업을 기다리는 프로세스, 스레드 행렬 또는 네트워크 접속을 기다리는 행렬, 너비우선 탐색, 캐시 등에 사용.

▼ 그림 5-8 큐

