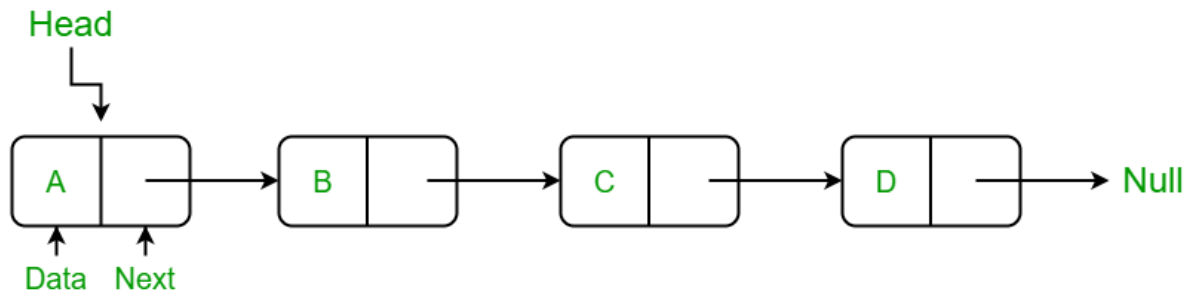


연결리스트(Linked List)

리스트(List)

데이터에 순서를 매겨 늘어놓은 자료구조로 기본구조는 크게 **선형리스트(Linear List)** 또는 **연결리스트(Linked List)**가 있다.

연결리스트란?



: 연결리스트의 기본 구조

: A부터 D까지 데이터가 순서대로 나열되고 각 데이터가 화살표로 연결되어 있다

: 이러한 구조에서는 중간에 건너 뛰거나 뒤에서 역방향으로 오는 것은 불가

: 각각의 원소(element)를 노드(node)라고 하며 모든 노드들은 데이터와 포인터(pointer)로 구성됨

이때, 포인터는 뒤쪽 노드를 가리키는(참조하는) 역할을 한다.

*또는 노드를 연결 리스트에서 하나의 원소에 필요한 데이터를 갖고 있는 자료 단위라고 설명하기도 함. 이때 데이터 필드는 원소의 값을 저장하는 자료구조, 링크 필드는 다음 노드의 주소를 저장하는 자료구조

: 레퍼런스는 메모리주소를 의미하며 이미지에서는 화살표를 지칭하는 것이라고 생각하면 된다

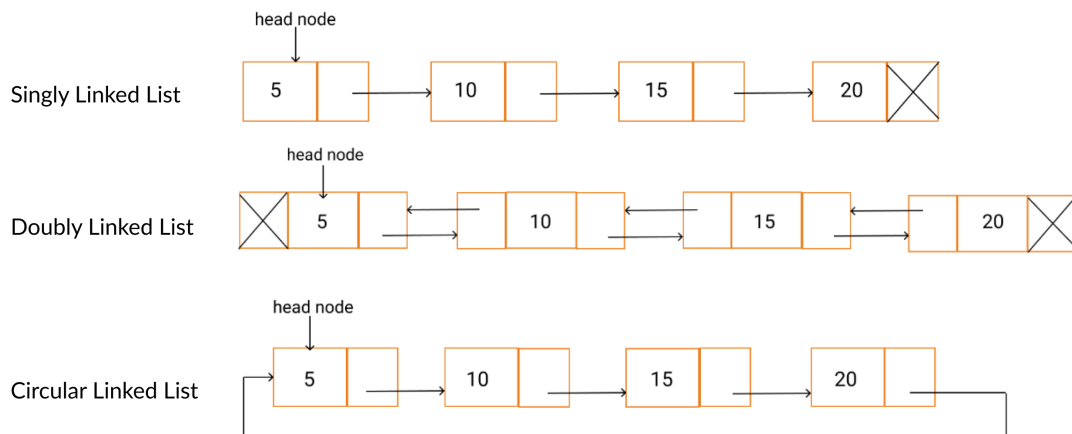
: 연결리스트에서 맨 앞에 있는 노드를 머리노드(head node), 맨 끝에 있는 노드를 꼬리 노드(tail node). 또한 각 노드에서 바로 앞에 있는 노드를 앞쪽 노드(predecessor node), 바로 뒤에 있는 노드를 뒤쪽 노드 (successor node)

: 주의 사항!

헤드노드 \neq 헤드 (포인터)

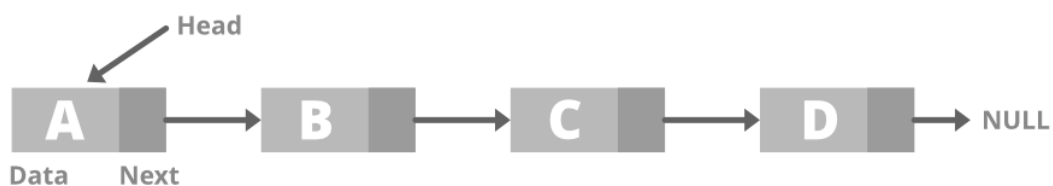
헤드 (포인터)는 리스트의 처음을 가리키는 포인터 부분만 가지고 있기 때문에 헤드 (포인터) 자체에는 데이터가 저장되어 있지 않다

: 연결리스트의 기본적인 형태 중 대표적인 단순 연결리스트, 이중 연결리스트, 원형 연결리스트



- 단순연결리스트(Singly Linked List)

Singly Linked List



: 노드가 하나의 포인터에 의해 다음 노드와 연결되는 구조를 가짐

= 동적 메모리 할당을 이용해 노드들을 한 방향으로 연결하여 리스트를 구현하는 자료구조

*이때 동적 메모리 할당을 이용한다는 것은 동적 메모리 할당을 받아 노드를 저장하기 때문

: 헤드가 가장 앞의 노드를 가리키고, 각 노드의 포인터가 연속적으로 다음 노드를 가리킴

: 최종적으로 None을 가리키는 노드가 리스트의 가장 마지막 노드

: 삽입이나 삭제 시 항목들의 이동은 필요없지만 항목을 탐색하려면 항상 첫 노드부터 원하는 노드를 찾을 때까지 순차탐색(sequential search)를 해야 한다.

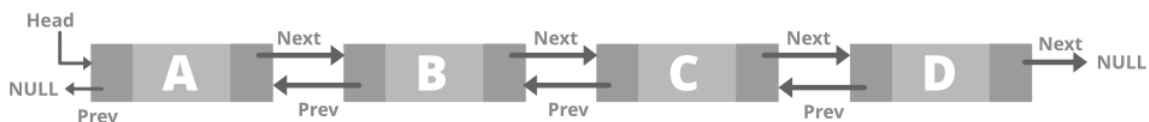
*underflow 발생 : 연결리스트가 empty인 상태에서 삭제 메소드가 호출되었을 때

: 단점. 역방향으로 노드들을 탐색할 수 없다

: 탐색의 경우 연결리스트의 노드들을 첫 노드부터 순차적으로 방문해야 하므로 노드의 수를 N 개라고 했을 때, $O(N)$ 의 시간이 소요된다. 삽입이나 삭제 연산은 각각 $O(1)$ 개의 레퍼런스만을 갱신하므로 $O(1)$ 의 시간이 소요됨. 그러나 특정노드의 레퍼런스가 주어지지 않았을 경우 탐색의 과정이 이루어져야 하기 때문에 최대 $O(N)$ 의 시간이 소요될 수 있다.

- 이중연결리스트(Doubly Linked List)

Doubly Linked List



: 양쪽 방향으로 순회할 수 있도록 노드를 연결한 리스트

= 각 노드가 두 개의 레퍼런스를 가지고 각각 이전노드와 다음 노드를 가리키는 연결리스트

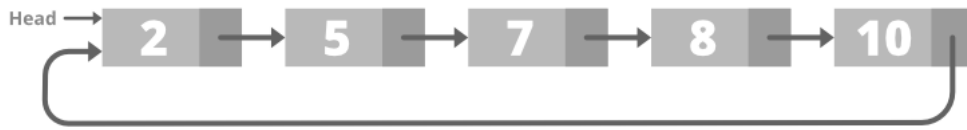
: 두 개의 포인터와 한 개의 데이터로 구성됨

: 단순연결리스트의 단점을 보완하였으나 각 노드마다 1개의 레퍼런스를 추가로 저장해야 한다는 단점을 가짐

: 삽입이나 삭제 연산 시 각각 $O(1)$ 개의 레퍼런스만을 갱신하므로 $O(1)$ 의 시간이 소요된다. 탐색 연산은 단순연결리스트와 마찬가지로 시작 또는 끝부터 노드들을 순차적으로 탐색해야 하므로 실제 항목들을 저장하고 있는 노드의 수를 N 이라고 했을 때, $O(N)$ 의 시간이 소요된다

- 원형연결리스트(Circular Linked List)

Circular Linked List



: 마지막 노드가 첫 노드와 연결된 단순연결리스트

= 마지막 노드를 참조하는 포인터가 단순연결리스트의 head 역할을 한다.

: 장점. 이러한 이유로 마지막 노드와 첫 노드를 $O(1)$ 시간에 방문할 수 있다

: 장점. 리스트가 empty가 아니면 어떤 노드도 None을 가지고 있지 않기 때문에 None 조건을 검사하지 않아도 된다

*연결 리스트의 앞이나 뒤에 노드를 추가할 때는 HEAD와 TAIL이 NULL이면 연결된 노드가 없는 빈 상태를 (Empty)라고 한다

: 단점. 반대 방향으로 노드들을 방문하기 쉽지 않으며, 무한 루프가 발생할 수 있다

: 삽입이나 삭제 연산 각각 $O(1)$ 개의 레퍼런스를 갱신하므로 $O(1)$ 시간이 수행된다. 탐색 연산의 경우 단순연결리스트의 탐색과 같이 tail로부터 노드들을 순차적으로 탐색해야 하므로 $O(N)$ 시간이 소요된다

연결리스트 관련 면접 질문

*검색 결과 연결리스트는 크게 CS 중 Data Structure에 해당하며 개념에 대해 설명하시오 또는 다른 리스트와 대조하면 설명하시오 라는 유형이 가장 일반적이었고 직접 코드를 작성해서 구현해보라는 질문들도 몇몇 확인할 수 있었다.

Q. Linked List에 대해 설명 해보세요

A. Linked List는 데이터와 다음 데이터를 가리키는 정보를 저장하고 있는 자료구조입니다. 데이터를 추가 하기에 좋지만 데이터를 찾을(탐색) 때 순차적으로 찾아야 하기 때문에 검색하는 것은 느립니다. 그래서 데이터의 입출력이 많을 때 Linked List를 사용 하는 것이 좋습니다

Q. 연결리스트(linked list) 에 대해서 설명하시오

A.

단순 연결 리스트 (singly linked list) : 단순 연결 리스트는 노드에 다음 노드의 주소를 가리키는 정보만 추가되어있는 가장 단순한 형태의 연결 리스트입니다. 가장 마지막 원소를 찾으려면 처음부터 리스트의 끝까지 탐색해야하기 때문에 시간복잡도가 $O(n)$ 입니다. 이 자료구조는 가장 첫번째 노드의 참조 주소를 잃어버릴 경우 데이터 전체를 못 쓰게되는 단점이 있

습니다. 또한 다음 노드를 참조하는 주소 중 하나가 잘못되는 경우에도 리스트가 끊어져 뒤쪽 자료들을 유실할 수 있는 불안정적인 자료구조입니다.

이중 연결 리스트 (doubly linked list) : 다음 노드의 참조뿐만 아니라 이전 노드의 참조도 같이 가리키게한 리스트 자료구조입니다. 단순 연결 리스트와 다르게 뒤에서부터 탐색하는 것이 빠르며 특정 노드를 삭제하는데 훨씬 간단하게 구현할 수 있습니다. 또한 첫 노드와 마지막 노드 중 하나를 가지고 있다면 전체 리스트를 순회할 수 있기 때문에 끊어진 리스트를 복구하는게 가능합니다. 그러나 관리해야할 참조 주소가 2개로 삽입이나 정렬의 경우 작업량이 더 많고 소모되는 메모리의 양도 많습니다.

원형 연결 리스트(circular linked list) : 단순 연결 리스트에서 마지막 원소를 null이 나닌 첫 노드의 주소를 가리키게하면 원형 연결 리스트 자료구조가 됩니다. 이 방법은 이중 연결 리스트에도 마찬가지로 적용됩니다.

Q. Array와 LinkedList의 차이가 무엇인가요?

A.

1) 접근

Array : Random Access를 지원한다. 요소들을 인덱스를 통해 직접 접근할 수 있다. 따라서 특정 요소에 접근하는 시간복잡도는 $O(1)$ 이다. Linkedlist는 Sequential Access를 지원한다. 어떤 요소를 접근할 때 순차적으로 검색하며 찾아야 한다. 따라서 특정 요소에 접근할 때 시간복잡도는 $O(N)$ 이다. 저장방식도 배열에서 요소들은 인접한 메모리 위치에 연이어 저장된다. 반면 Linkedlist에서는 새로운 요소에 할당된 메모리 위치 주소가 linkedlist의 이전 요소에 저장된다.

2) 삽입과 삭제

저장방식도 배열에서 요소들은 인접한 메모리 위치에 연이어 저장된다. Linkedlist에서는 새로운 요소에 할당된 메모리 위치 주소가 linkedlist의 이전 요소에 저장된다. 배열에서 삽입과 삭제는 $O(N)$ 이 소요되지만, Linkedlist에서 삽입과 삭제는 $O(1)$ 이 소요된다.


3) 메모리 할당

배열에서 메모리는 선언 시 컴파일 타임에 할당이 된다.(정적 메모리 할당) 반면 Linkedlist에서는 새로운 요소가 추가될 때 런타임에 메모리를 할당한다.(동적 메모리 할당) 배열은 Stack 섹션에 메모리 할당이 이루어 진다. 반면 Linkedlist는 Heap 섹션에 메모리 할당이 이루어진다.

참고 자료 링크

Types of Linked List - GeeksforGeeks

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers. In simple words, a linked list

 <https://www.geeksforgeeks.org/types-of-linked-list/>



연결 목록 복제

연결 목록 복제: 단일 연결 목록을 사용하여 해당 목록의 전체 복사본을 반환하는 함수를 작성하십시오.

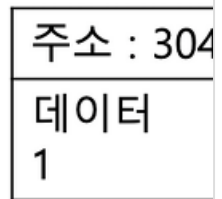
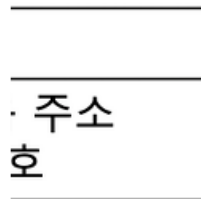
 <https://www.techiedelight.com/ko/clone-given-linked-list/>



코딩 면접 질문들 정리 - 제5편 자료구조 Stack, Hash


1. 알고 있는 자료구조의 종류에 대해 이야기 해보세요. List(리스트), Linked List(링크드 리스트), Array(배열), Stack(스택), Queue(큐), Dequeue(디큐), Tree(트리), Heap(힙), Graph(그래프) 2.Linked

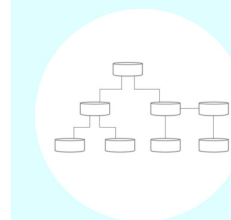
 <https://krksap.tistory.com/1706>



[자료구조] 면접질문 모음

자료구조와 알고리즘 자료구조와 알고리즘에 대해 설명해주세요. 자료구조는 데이터를 원하는 규칙 또는 목적에 맞게 저장하기 위한 구조이고, 알고리즘이란 자료구조에 쌓인 데이터를 활용해 어떠한 문제를

 <https://velog.io/@humblechoi/%EC%9E%90%EB%A3%8C%EA%B5%AC%EC%A1%B0-%EB%A9%B4%EC%A0%91%EC%A7%88%EB%AC%B8-%EB%AA%A8%EC%9D%8C>



Data Structure

<https://github.com/devetude/Data-Structure-QnA>