연결리스트 → SLL 단순 stack
→ CLL 원형 큐
→ DLL 이중 스택, 큐, 덱
복잡함

6장 연결된 구조

연결된 구조

- Linked List, 연결리스트라고함
각 노드는 데이터와 포인터를 가지고 한줄로 연결되어있는 방식의 자료구조
연결된 구조는 떨어진 곳에 존재하는
본래 C 언어에서는 주로한 데이터군과 파이썬의 리스트 타입이 연결된 구조와 기능을 모두지원

특징 vs 배열

장점
1, 용량이고 정해지지않음
✓ 필요한것만 필요할때 사용 → 효율적
✓ 컴퓨터 메모리가 남아있다면 계속 자료를 넣을 수있음

2, 중간에 자료를 삽입 삭제 하는 것이 용이
✓ 리스트와 같은 배열구조에서는 원소의 항목이 필요
✓ 연결된 구조에서는 링크만수정하면되므로 시간복잡도 O(1)

단점 - 3, N번째 항목에 접근하는데 시간복잡도 O(n)
연결정보를 찾는 시간이 필요하므로 접근속도가 느림

노드

데이터 저장단위 (데이터값, 포인터구성)

- 시작 노드 헤드포인터
  - 각 노드안에서 다음이나 이전의 노드와의 연결정보를 가진공간

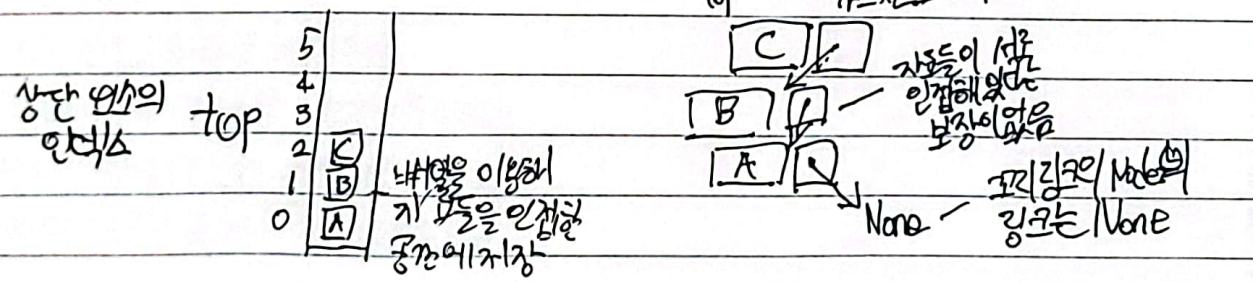
64까지

단순 연결리스트
V 하나의 방향으로만 연결된구조
V 링크는 하나이며, 이밴수는 다음 노드의 주소를 가지고있음
V 마지막노드

원형 연결리스트
V 단순 연결리스트와 동일한 노드구조
V 마지막 링크값이 첫 번째 노드를 가리킴

이중 연결리스트 + 원형
V 하나의 노드는 이전노드 다음 노드를 모두 알고있음
V 두개의 링크는 선행노드, 후속노드를 가르킴

일반스택 VS 연결된 스택



· 배열구조의 스택 연결된구조의 스택

연결된 스택
· 삽입연산
V 연결리스트에 삽입데이터를 직접 삽입할수 없음
V 데이터를 넣을 노드를 추가 해야함

삭제연산
상단항목을 꺼내서 반환하는연산 top이 가리키는 노드를 꺼내고 데이터를 반환
반환은 노드가 아닌 스택 반환
메모리해제 신경쓸필요 없음

?말 후위표기 식낸다

6.3 단순 연결리스트의 응용 연결된 리스트

연결리스트
- 스택과 달리 리스트는 항목의 삽입 삭제가 어느위치에서나 가능
- 스택보다 복잡한 구조

getNode()
리스트가 공백이면 None을 반환
head가 node에 가리키게함
pos가 0보다 크고 Null과 None이 아닐때까지
노드가 가리키는 다음부분을 노드라하고 pos1씩감소
노드가 while문을 만족못하면 반환

getEntry()
pos번째 노드를 node라고함
찾는 node가 None이라면 None반환
노드를 찾았다면 노드의 데이터필드 반환

삭제연산
1) before이 내가 삭제할노드의 다음노드를 가르키게함

삽입연산
노드N이 노드C를 가르키게 함
노드B가 노드N을 가르키게 함

6.4 연결큐

연결된 형태로 구현한 큐 연결된 큐
크기가 제한적이지 않을 뿐만 아니라 메모리를 효율적으로 사용
프로그램이 더 복잡해지지만 메모리 공간을 더 사용
마지막 노드의 링크가 첫노드를 가르킴

6.5 이중연결리스트의 응용: 연결된 덱

노드마다 2개의 링크를 관리 복잡
선형 자료구조의 연산들을 가장 효율적으로 구현가능