스택, 큐, 우선순위 큐, 데크

1. 자료구조와 자료형 구분하기

- 컴퓨터과학에서 자료구조란 데이터에 효율적으로 접근하고 조작하기 위한 데이터의 조직, 관리, 저장 구조를 말한다.
- 보통 자료형과 자료구조를 많이 혼동하는데, **자료형**이란 컴파일러 또는 인터프리터에게 프로그래머가 데이터를 어떻게 사용하는지를 알려주는 일종의 데이터 속성이다. 자료형은 자료구조에 비해 훨씬 더 구체적이며, 특정 언어에서 자료형이라 함은 정수, 실수, 문자열 등 해당 언어에서 지원하는 원시 자료형까지 포함하는 모든 자료의 유형을 말한다.
- **자료구조**는 일반적으로 원시 자료형을 기반으로 하는 배열, 연결 리스트, 객체 등을 말하며, 자료형의 관점에서 보자면 여러 원시 자료형을 조합한 자료구조는 복합 자료형이 된다.

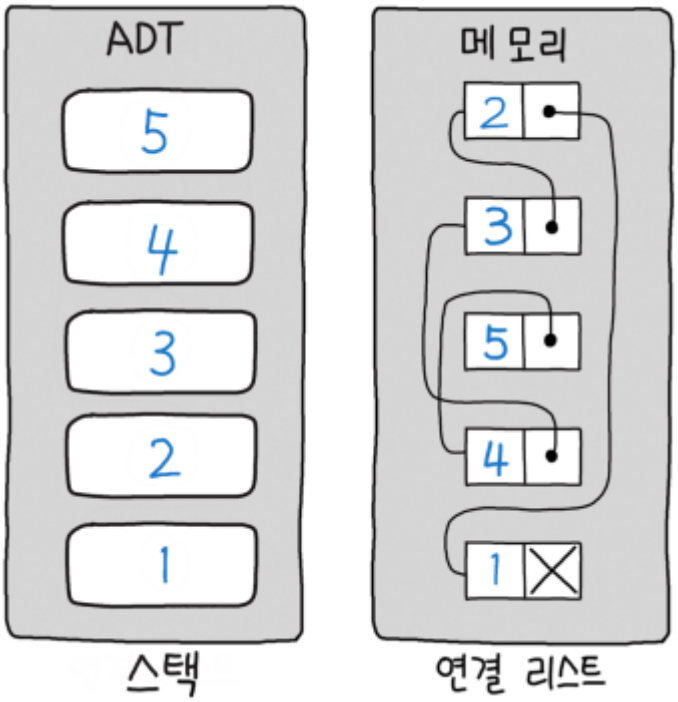
2. 스택과 큐

- 스택과 큐는 프로그래밍이라는 개념을 탄생할 때부터 사용된 가장 고전적인 자료구조 중 하나로, 그 중에서도 스택은 모든 애플리케이션을 만들 때 사용되는 자료구조다.
  - 스택 : LIFO(Last-In-First-Out) / 후입선출
  - 큐 : FIFO (First-In-First-Out) / 선입선출
- 파이썬은 스택과 큐 자료형을 별도로 제공하지는 않지만, 리스트가 사실상 모든 연산을 지원한다.

3. 스택

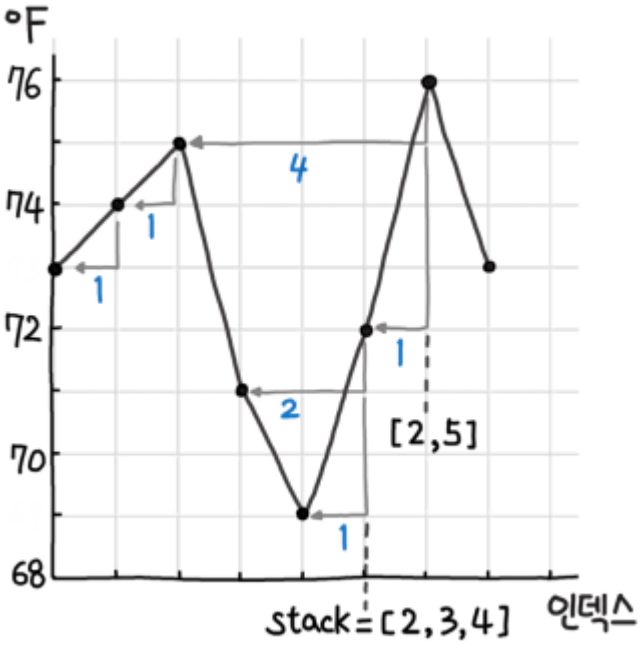
- 스택은 다음과 같은 2가지 주요 연산을 지원하는 요소의 컬렉션으로 사용되는 추상 자료형이다.
  - push(): 요소를 컬렉션에 추가한다.
  - pop(): 아직 제거되지 않은 가장 최근에 삽입된 요소를 제거한다.

- 스택 추상 자료형(Abstract Data Tpye)(이하 ADT)을 연결 리스트로 구현하면 다음 그림과 같다.
    - 아래 그림에서 ADT는 스택의 연산을 지원하기 위해 1부터 5까지 각 요소가 접시 쌓듯 차곡차곡 놓이겠지만, 실제로 연결 리스트로 구현하게 된다면 물리 메모리 상에는 순서와 관계 없이 여기저기에 무작위로 배치되고 포인터로 가리키게 될 것이다.



3-1) 일일 온도 (스택 관련 문제)

- 매일 화씨 온도 리스트 T를 입력받아서, 더 따뜻한 날씨를 위해서는 며칠을 더 기다려야 하는지를 출력하라.
- 입력: T = [73, 74, 75, 71, 69, 72, 76, 73]
- 출력: [1, 1, 4, 2, 1, 1, 0, 0]
- 주의) 입력이 ebcabc라면, 출력은 eabc



접근 방식: 현재의 인덱스를 계속 스택에 쌓아두다가, 이전보다 상승하는 지점에서 현재 온도와 스택에 쌓아둔 인덱스 지점의 온도 차이를 비교해서, 더 높다면 스택의 값을 pop()으로 꺼내고 현재 인덱스와 스택에 쌓아둔 인덱스의 차이를 정답으로 처리한다.

```
def dailyTemp(T):
    # 처음 디폴트 값을 0으로 설정
    answer = [0] * len(T)
    # 스택을 쌓을 리스트 생성
    stack = []
                                #  i = [0,  1,  2,  3,  4,  5,  6,  7]
    for i, cur in enumerate(T):  # T = [73, 74, 75, 71, 69, 72, 76, 73]
        # 현재 온도가 스택 값보다 높다면 정답 처리
        while stack and (cur > T[stack[-1]]):
            last = stack.pop()
            answer[last] = i - last
        stack.append(i)
    return answer

print(dailyTemp([73, 74, 75, 71, 69, 72, 76, 73]))
# [1, 1, 4, 2, 1, 1, 0, 0]

# for문 끝에 print(stack)하면
# [0], [1], [2], [2, 3], [2, 3, 4], [2, 5], [6], [6, 7]
```

4. 큐

- FIFO(선입선출)로 처리되며, 줄을 서는 것에 비유할 수 있다. 큐는 너비 우선 탐색(BFS)이나 캐시 등을 구현할 때도 널리 사용된다.

4-1) 백준 10845 (큐 관련 문제)

정수를 저장하는 큐를 구현한 다음, 입력으로 주어지는 명령을 처리하는 프로그램을 작성하시오.

명령은 총 여섯 가지이다.

- push X: 정수 X를 큐에 넣는 연산이다.
- pop: 큐에서 가장 앞에 있는 정수를 빼고, 그 수를 출력한다. 만약 큐에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- size: 큐에 들어있는 정수의 개수를 출력한다.
- empty: 큐가 비어있으면 1, 아니면 0을 출력한다.
- front: 큐의 가장 앞에 있는 정수를 출력한다. 만약 큐에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- back: 큐의 가장 뒤에 있는 정수를 출력한다. 만약 큐에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- 입력(예시):

  5

  push 1

  push 2

  pop

  size

  empty

  pop

```
from sys import stdin
N = int(stdin.readline())
Que = []
for i in range(N) :
    A = stdin.readline().split()

    if A[0] == 'push' : Que.append(A[1])
```

```
    elif A[0] == 'pop' :
        if Que : print(Que.pop(0))
        else : print(-1)

    elif A[0] == 'size' : print(len(Que))

    elif A[0] == 'empty' :
        if len(Que) == 0 : print(1)
        else : print(0)

    elif A[0] == 'front' :
        if len(Que) == 0 : print(-1)
        else : print(Que[0])

    elif A[0] == 'back' :
        if len(Que) == 0 : print(-1)
        else : print(Que[-1])
```

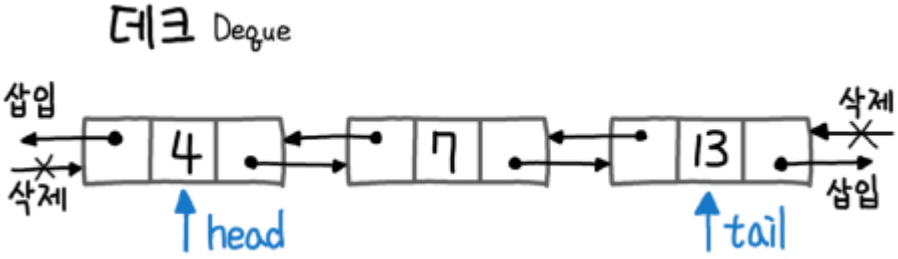
5. 우선순위 큐

- 우선순위 큐는 어떠한 특정 조건에 따라 우선순위가 가장 높은 요소가 추출되는 자료형이다.
- 대표적으로 최댓값 추출을 들 수 있는데, 예를 들어 큐에 [1, 4, 5, 3, 2]가 들어 있고 최댓값을 추출하는 우선순위 큐가 있다고 가정하자. 이 경우 항상 남아 있는 요소들의 최댓값이 우선순위에 따라 추출되어 5, 4, 3, 2, 1 순으로 추출된다.

6. 데크

- 스택과 큐의 특징을 모두 갖고 있는 복합 자료형

- 데크는 Double-Ended Queue의 줄임말로, 글자 그대로 양쪽 끝을 모두 추출할 수 있는, 큐를 일반화한 형태의 추상 자료형이다.



- 위의 이중 연결 리스트처럼 양쪽에서 head와 tail이라는 이름의 두 포인터를 가지고 있다가 새로운 아이템이 추가될 때마다 앞쪽 또는 뒤쪽으로 연결시켜주기만 하면 된다. 당연히 연결 후에는 포인터를 이동하면 된다.

- 파이썬에서 데크 자료형을 사용하는 방법

```
import collections
d = collections.deque()
print(type(d))  # <class 'collections.deque'>
```

- 데크 메서드
  - `deque.append(item)`: item을 데크의 오른쪽 끝에 삽입한다.
  - `deque.appendleft(item)`: item을 데크의 왼쪽 끝에 삽입한다.
  - `deque.pop()`: 데크의 오른쪽 끝 엘리먼트를 가져오는 동시에 데크에서 삭제한다.
  - `deque.popleft()`: 데크의 왼쪽 끝 엘리먼트를 가져오는 동시에 데크에서 삭제한다.
  - `deque.extend(array)`: 주어진 배열(array)을 순환하면서 데크의 오른쪽에 추가한다.
  - `deque.extendleft(array)`: 주어진 배열(array)을 순환하면서 데크의 왼쪽에 추가한다.
  - `deque.remove(item)`: item을 데크에서 찾아 삭제한다.
  - `deque.rotate(num)`: 데크를 num만큼 회전한다(양수면 오른쪽, 음수면 왼쪽).

- 데크(deque), 언제, 왜 써야 하는가?
  - 데크(deque)는 스택처럼 사용할 수도 있고, 큐 처럼 사용할 수도 있다.
  - 시작점의 값을 넣고 빼거나, 끝 점의 값을 넣고 빼는 데 최적화된 연산 속도를 제공한다.
  - 대부분의 경우, 데크(deque)는 리스트(list)보다 월등한 옵션이다.
  - 데크는 특히 push/pop 연산이 빈번한 알고리즘에서 리스트보다 월등한 속도를 자랑한다.

6-1) 유효한 팰린드롬 (데크 관련 문제) (리스트로 변환)

- 주어진 문자열이 팰린드롬인지 확인하라. 대소문자를 구분하지 않으며, 영문자와 숫자만을 대상으로 한다.
- 슬라이싱 사용 못한다고 가정.
- 입력: “A man, a plan, a canal: Panama”
- 출력: True

```
def isPalindrome(s):
    strs = []
    for char in s:
        # isalnum()은 영문자, 숫자 여부를 판별하는 함수
        if char.isalnum():
            strs.append(char.lower())

    while len(strs) > 1:
        # 첫번째 글자와 마지막 글자 확인
        if strs.pop(0) != strs.pop():
            return False

    return True
```

6-2) 유효한 팰린드롬 (데크 관련 문제) (데크 활용)

```
import collections

def isPalindrome(s):
    # 자료형 데크로 선언
    strs = collections.deque()

    for char in s:
        if char.isalnum():
            strs.append(char.lower())

    while len(strs) > 1:
        # 첫번째 글자와 마지막 글자 확인
        if strs.popleft() != strs.pop():
            return False

    return True
```

풀이	방식	실행 시간
1	리스트로 변환	304밀리초
2	데크 자료형을 이용한 최적화	64밀리초
3	슬라이싱 사용	36밀리초
4	C 구현	4밀리초

예제문제

백준

1715 카드 정렬하기

1966 프린터 큐

1158 요세푸스 문제

1764 듣보잡

https://www.acmicpc.net/problemset?sort=ac_desc&algo=175