빅오

- O(): 상한
  - 입력값이 무한대로 향할때 함수의 상한을 설명하는 수학적 표기 방법
  - 점근적 실행 시간을 표기할때 널리쓰임
  - 시간복잡도의 사전적 정의는 어떤 알고리즘을 수행하는 데 설명하는 계산 복잡도
    - O(1): 상수 시간
    - O(logn): ex:이진 검색
    - O(n): 선형 시간
    - O(nlogn): 모든 수를 비교해야 하는 정렬 알고리즘의 최소 시간, 효율 좋은 알고리즘이 이에 해당함 ex: 머지, 퀵, 병합
    - O(n^2): ex: 버블
    - O(2^n): ex: 피보나치
    - O(n!): ex: TSP
- Ω(): 하한
- Θ(): 평균

  -> 상한 != 최악, 상한은 상한일뿐 최악의 경우는 아니다.
- 평균 상환 분석

자료형

- bool
- int
- float
- str
  - **str.**count(string): string의 갯수 반환
  - **str.**find(string): string의 시작 인덱스 반환
    - 존재하지 않는다면 -1 반환
  - **str.**index(string): string의 맨처음 나온 인덱스 반환
    - 존재하지 않는다면 오류
  - **str.**join(string): str을 string 사이에 삽입
  - **str.**upper(): 대문자로 변환
  - **str.**lower(): 소문자로 변환
  - **str.**split(string): string 기준으로 str을 나눈 리스트 반환
    - string의 기본값은 공백
  - **str.**strip(): 공백 지움
  - **str**.lstrip(): 왼쪽 공백 지움
  - **str**.rstrip(): 오른쪽 공백 지움
  - **str**.replace(string1, string2): str안에 있는 string1을 string2로 바꿈
- list: 가변(mutable) 리스트
  - len(**list**) - O(1): 요소의 갯수 반환
  - **list**[l] - O(1): i번째 요소 반환
  - **list**[i:j] - O(k): i부터 j-1번째 요소 반환
  - elem in **list** - O(n): elem이 list안에 있는지 true/false 반환
  - **list**.count(elem) - O(n): list안에 elem의 개수를 반환
  - **list**.index(elem) - O(n): list안에 elem의 인덱스를 반환
  - **list**.append(elem) - O(1): list끝에 elem을 추가함(elem을 풀지 않고 삽입)
    - 더블링 시(최악)에 O(n)이지만 평균 상환 분석

    - **list**.extend(elem) - O(1): list끝에 elem을 추가함(
    - **list**.pop() - O(1): list의 마지막 요소를 반환후 삭제
    - **list**.pop(0) - O(n): list의 첫 요소를 반환후 삭제
        - 복사가 일어나므로 O(n), 큐의 연산을 주로 사용할때는 데크를 사용
    - del **list**[i]- O(n): i번째 요소 삭제
    - **list**.remove(elem) - O(n): list의 elem요소 삭제
    - **list**.sort() - O(n log n): 팀소트
        - key를 넘겨서 정렬 방법, 순위를 정해줄 수 있음
    - min(**list**) / max(**list**) - O(n): 최솟값/최댓값을 반환
    - **list**.reverse() - O(n): list의 요소를 뒤집음
- tuple: 불변(immutable) 리스트
- dict: 딕셔너리(해시 테이블)
    - len(**dict**) - O(1): 요소의 갯수 반환
    - **dict**[key] - O(1): 키에 해당하는 값 반환
    - key in **dict** - O(1): 딕셔너리에 키가 있는지 true/false 반환
        - python3.6 이전: 입력순서가 유지되지 않음 -> collections.OrderedDict() 별도 자료형 제공
        - python3.6 이후: 입력순서가 유지됨
- set

내장 함수
- map(func, iterable): func를 iterable의 각 요소에 시행한 리스트 반환
- filter(func, iterable): func가 true인 요소로 된 리스트 반환
- zip(*params): params의 같은 인덱스끼리의 요소를 합친 제러레이터 반환
    - 제너레이터는 list(generator)로 쉽게 변환 가능
- int(char, order): char(문자열로 된 숫자)를 숫자로 order진법으로 해석한 10진수를 반환
    - 숫자가 아닌 문자열일 경우 에러
- bin(number): number를 2진수로 변환해서 문자열로 돌려줌
    - ex: '0b0011'
- zfill(n): 문자열을 n만큼 0을로 앞부분에 채워서 늘려준 문자열 반환
- rjust(n, char): 문자열을 n만큼 char로 앞부분에 채워서 늘려준 문자열 반환

라이브러리
- from typing import *: 타입 힌트를 위한 라이브러리
    - *은 List, Dict, Tuple, Set을 포함한다.
    - 위와 같은 자료형의 타입 힌트를 사용할 수 있게 한다.
- import math
    - math.sqrt(exe): exp의 루트값을 반환해줌
- import collections
    - **collections**.defaultdict(type): 존재하지 않는 키를 조회할 경우 type 자료형으로 초기화 해서 생성
        - int: 0으로 생성해줌
            - 2번째 인자를 주어서 기본값 정해줄수 있음. ex: defaultdict(int, 2)
    - **collections.**Counter(**dict**): dict의 각 값들을 키로, 그 갯수를 값으로 가지는 딕셔너리 반환
        - Counter.most_common(num): 가장 빈도수가 높은 순으로 num개까지 담은 리스트 반환
            - **dict**가 아닌 iterable도 가능하다. ex: list
    - **collections.**OrderedDict(**dict**): dict의 순서를 유지하는 딕셔너리 생성
- import bisect: 이분 검색 라이브러리
    - bisect.bisect_left(arr, n): arr에서 n의 왼쪽 인덱스를 반환

    - bisect.bisect_right(arr, n): arr에서 n의 오른쪽 인덱스를 반환
- import heapq: 리스트를 힙처럼 다룰수 있게 도와주는 라이브러리
    - heapq.heappush(list, elem): list에 elem을 최소 힙 구조로 넣어줌
        - O(logn)
        - 첫번째 인덱스도 활용함
        - 최대 힙은 heapq.heappush(list, (priority, value))에서 priority를 -value로 넣어줌
            - value를 가져올때는 heapq.heappop(heap)[1]로 priority 말고 value를 읽어줌
    - heapq.heappop(list): list에서 가장 작은 원소를 삭제, 반환함
        - 삭제를 원하지 않는 경우에는 list[0]와 같이 인덱스로 얻음
    - heapq.heapify(list): 기존의 list를 힙으로 변환
        - O(n)
    - heapq.nlargest(n, iterable, key=None): 가장 큰 n개의 원소를 가진 리스트 반환
    - heapq.nsmallest(n, iterable, key=None): 가장 작은 n개의 원소를 가진 리스트 반환
- import functools
    - functools.reduce(func(acc, val), iterable[, init]): init으로 초기화된 acc에 func를 누적해서 acc 반환
    - functools.cmp_to_key
        - sorted, sort 함수에 key에 cmp를 대입할 수 있음
            - a, b 중에 a가 작으면 음수, 같으면 0, 크면 양수 반환
            - 맞으면 음수, 아니면 양수 반환 하게 만들면 됨
- import itertools
    - itertools.permutations(itterable, num): itterable의 순열을 num개로 만들어서 반환
        - list(tuple...)로 반환
    - itertools.combinations(utterable, num): itterable의 조합을 num개로 만들어서 반환
        - lists(tuple...)로 반환
- from operator import *
    - operator.mul: 인자 2개를 받아서 곱을 반환
    - operator.add: 인자 2개를 받아서 합을 반환
- import re: 정규식
    - re.compile(reg): reg정규식 컴파일 객체(p) 반환
        - DOTALL - S: .문자가 \n도 포함하여 매치
        - IGNORECASE - I: 대/소문자 상관업이 매치
        - MULTILINE - M: 여러 줄과 매치
        - VERBOSE - X: verbose모드
    - p.match(str): str 문자열의 처음부터 매치되는지 조사
        - true면 match객체, false면 None 반환
    - p.search(str): str 문자열 전체를 검색하여 매치되는지 조사
        - true면 match객체, false면 None 반환
        - 맨 처음 매치된 문자열만 찾음
    - p.findall(str): str 문자열에서 매치되는 문자열 리스트 반환
    - p.finditer(str): p.findall()과 동일한 기능, iterable로 반환
    - p.sub(to, str): str에서 p에 의해 매치된 문자열을 to로 바꿈
        - 그룹을 참조할때는 to 대신 /g<num> g 인덱스 그룹
        - to 대신 func(match) 넣어서 변환 가능(func의 인자로 match객체 전달됨)
    - p.subn(to, str): p.sub와 동일, (바뀐 문자열, 바뀐 횟수)튜플 반환
    - match.group(): 매치된 문자열 반환
        - 그루핑 된경우 match.group(g) g그룹(인덱스나 그룹이름) 문자열 반환

    - match.start(): 매치된 문자열의 시작 인덱스 반환
    - match.end(): 매치된 문자열의 끝 인덱스 반환
    - match.span(): 매치된 문자열의 (시작, 끝) 튜플 반환
        - []: 문자 클래스
            - ^: not
            - \d: 숫자와 매치, [0-9]
            - \D: 숫자가 아닌 것과 매치, [^0-9]
            - \w: 문자+숫자와 매치, [a-zA-Z0-9_]
            - \W: 문자+숫자가 아닌것과 매치, [^a-zA-Z0-9_]
            - \s: whitespace와 매치, [\t\n\r\f\b], 공백도 매치
            - \S: whitespace가 아닌 것과 매치, [^ \t\n\r\f\b]
        - .: \n을 제외한 모든 문자와 매치
        - *: 문자가 0번 이상 반복되면 매치
        - +: 문자가 1번 이상 반복되면 매치
        - {m, n}: 최소m번 최대n번 반복되면 매치
            - m과 n은 생략가능
        - ?: 0번이나 1번 반복되면 매치
        - |: or
        - ^: 문자열의 맨 처음과 매치
            - MULTILINE의 경우 각 줄의 처음과 매치
        - $: 문자열의 맨 마지막과 매치
            - MULTILNE의 경우 각 줄의 마지막과 매치
        - \A: 문자열의 맨 처음과 매치
            - MULTILINE의 경우에도 맨 처음만 매치
        - \Z: 문자열의 맨 마지막과 매치
            - MULTILNE의 경우에도 맨 마지막과 매치
        - \b: 단어 구분자
        - \B: \b의 반대
    - 그루핑
        - (): 그룹을 만들어줌
            - (): 자동 인덱스 그룹
            - (num): num에 해당하는 인덱스 그룹
            - (?P<name>): name으로 그룹
            - \num: num 인덱스에 해당하는 그룹 재참조
            - (?=name): name그룹 재참조
    - 전방 탐색
        - (?=...): ...에 해당하는 정규식이 매치되어야 하고, 문자열이 소비되지 않음
        - (?!...): ...에 해당하는 정규식이 매치되지 않아야 하고, 문자열이 소비되지 않음

팁

- is 와 ==
    - is는 id( )값을 비교, ==은 값을 비교
- f문자열 포매팅
    - f' '로 선언하여 사용
    - f'{val}'
    - f'{'':<num}': 10칸 왼쪽 정렬(>, ^은 오른쪽, 가운데 정렬)
    - f'{'':<`num}': 10칸 왼쪽 정렬, 빈칸`로 채움(>, ^은 오른쪽, 가운데 정렬)

    - f'{'':0.numf}': 소수점 num자리까지 표현
    - f'{'':numm.numf}': 소수점 num자리까지 표현: numm자리까지 표현, num자리까지 표현
- **list**.sort vs sorted
    - **list**.sort는 리스트 자체를 정렬, 반환값 없음
    - sorted는 정렬후에, 정렬된 요소를 리스트로 반환
        - key로 함수를 줘서 정렬 방법, 정렬 순위를 정할수 있다.
- 슬라이싱으로 뒤집기
    - iterable[::-1]로 뒤집을수 있다.
- r스트링
    - 정규식을 사용할때 로우 스트링(r'')을 사용하여 이스케이프를 피하자
- not, None, 0, False, ..
    - not을 이용해서 거짓 같은 값을 걸러내지말자
        - 오류 발생시 찾기 어려움
- *, **
    - *은 튜플, 리스트를 언패킹 한다
        - 함수의 인자, 변수 에서는 패킹을 한다
    - **은 키/값 페어를 언패킹 한다
- permutations, combinations
    - nPr: n! / (n - r)!
    - nCr: n! / r! (n - r)!