프로그래머스

해시

스택/큐

힙

정렬

완전탐색

탐욕법

동적계획법

DFS/BFS

이분탐색

그래프

완전탐색

모의고사

https://programmers.co.kr/learn/courses/30/lessons/42840?language=python3

문제 분류 : 완전 탐색

문제 상황

1번 수포자 : 1, 2, 3, 4, 5 반복으로 찍음

2번 수포자 : 2, 1, 2, 2, 2, 3, 2, 4, 2, 5 반복으로 찍음

3번 수포자 : 3, 3, 1, 1, 2, 2, 4, 4, 5, 5 반복으로 찍음

위 상황에서 답안지의 역할을 하는 배열이 입력으로 주어짐. 원소 개수는 10,000 이하

이때, 가장 많은 답을 맞춘 사람을 출력. 같으면 둘 다 출력

입력

1~5범위의 10,000개 이하의 원소를 갖는 배열

풀이 아이디어

10,000개 이하 -> 수포자 3명 다 돌려도 매우 적은 범위.

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

[소수 찾기]

https://programmers.co.kr/learn/courses/30/lessons/42839?language=python3

문제 분류 : 완전 탐색

문제 상황
한자리 숫자가 적힌 종이 조각이 흩어져 있을때, 흩어진 종이 조각을 붙여 소수를 몇 개 만들 수 있는가.
종이조각의 수는 1이상, 7이하.

입력
String으로 이루어진 숫자. "011" -> 0, 1, 1 종이조각이 있다는 뜻

주의
11과 011은 같은 수로 취급함.

풀이 아이디어
일단 한 수를 소수인지 판별해주는 함수를 빠르게 팜.
7! 돌면 다 검사할 수 있겠음.

참고로 가져온 풀이는 좀 고수용
소수를 판별하기 위해서 에라토스테네스의 체를 사용함.
(https://ko.wikipedia.org/wiki/%EC%97%90%EB%9D%BC%ED%86%A0%EC%8A%A4%ED%85%8C%EB%84%A4%EC%8A%A4%EC%9D%98_%EC%B2%B4) - 위키
근데 그거 사용하는 것도 set을 이용해 좀 고급지게 구함.

코드

```
from itertools import permutations
def solution(n):
    a = set()
    for i in range(len(n)):
        a |= set(map(int, map("".join, permutations(list(n), i + 1))))
    a -= set(range(0, 2))

  # 에라토스테네스의 체
    for i in range(2, int(max(a) ** 0.5) + 1):
        a -= set(range(i * 2, max(a) + 1, i))

    return len(a)
```

[숫자야구]

https://programmers.co.kr/learn/courses/30/lessons/42841?language=python3

문제 분류 : 완전 탐색

문제 상황

3자리로 하는 숫자야구를 진행함. 문제 입력이 2중 배열로 주어지는데

배열 안의 세 원소 [n, s, b]는 n=물어본 세 자리 수, s=스트라이크 수, b=볼의 수 로 이루어짐.

입력으로 주어진 모든 질문을 다 썼을 때 가능한 답의 개수를 리턴함.

질문의 수는 100 이하임.

야구 게임의 답이 되는 숫자는 각자 다른 1~9로 이루어진 임의의 3자리 숫자.

Ex) 012 -> (x), 234->(o), 224->(x)

입출력 예

입력 : [[123, 1, 1], [356, 1, 0], [327, 2, 0], [489, 0, 1]]

출력 : 2

풀이 아이디어

일단 123 ~ 987 까지 들어있는 set나 배열을 마련한다음(permutatio으로) 지워 나가는게 맞는거 같다.

가져온 풀이가 정말 깔끔함.

우선 답이 될 숫자와 질문으로 들어온 숫자 둘을 넣으면 몇 스트라이크, 몇 볼인지 판별해주는 함수를 파 놓음(st_B)

그 다음 답을 낼땐 정답 후보를 list(itertools.permutations([1,2,3,4,5,6,7,8,9], 3)) 이거로 만들어 내고

N개의 질문을 돌면서 1번째 질문을 통과한 얘들을 모으고 통과한 얘들중에 2번째 질문을 통과한 얘들 모으고... 이런식으로 돌면 빠르게 쳐 낼 수 있음.

코드

```
def st_B(given, chosen):
    st = 0
    B = 0
    chosen_dif = []
    given_dif = []
    for i in range(3):
        if given[i] == chosen[i]:
            st += 1
        else:
            given_dif.append(given[i])
            chosen_dif.append(chosen[i])
    for num in chosen_dif:
        if num in given_dif:
            B += 1
    return st, B

import itertools

def solution(baseball):

    first = list(itertools.permutations([1,2,3,4,5,6,7,8,9], 3))
    second = []

    for each in baseball:
        given = [int(i) for i in str(each[0])]
        stb = (each[1], each[2])
        for chosen in first:
            if st_B(given, chosen) == stb:      # 만들어 놓은 함수로 각 정답후보가 몇스 몇볼인지 판별
                second.append(chosen)           # 통과하면 통과한 얘들끼리의 배열에 넣어줌

        first = second                          # 통과한 얘들을 다시 후보에 올려둠
        second = []                             # 다음번에 통과한 얘들을 모으기 위해 비워둠
    return len(first)
```

[카펫]

https://programmers.co.kr/learn/courses/30/lessons/42842?language=python3

문제 분류 : 완전 탐색

문제 상황

레오가 브라운, 레드가 칠해진 격자모양의 카펫을 보고 왔음. 아래와 같이 카펫의 테두리가 브라운으로 칠해지고 안쪽이 레드로 칠해짐. 근데 레오쉑이 브라운, 레드 개수만 세오고 카펫의 크기를 못세움.

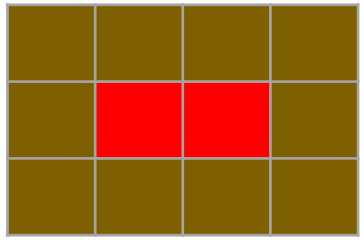
브라운, 레드 개수가 주어질때 카펫의 가로, 세로를 출력하면 됨. (단, 가로가 세로보다 같거나 김)

입력

Brown : 10, red: 2, return : [4, 3]

Brown : 8, red: 1, return : [3, 3]

Brown : 24, red: 24, return : [8, 6]



풀이 아이디어

우선 브라운+레드 개수 해서 n*m의 후보를 만들 것 같음.

후보들 중 테두리의 개수(2n+2m-4)가 브라운의 개수랑 같은 것만 뽑으면 되지 않을까?

내 풀이나 다른 사람 풀이나 구조는 같음. 다만 얼마나 깔끔하게 정리했냐차이

코드

```
# 내 풀이
def solution(brown, red):
    area = brown+red
    nominated = set([i if area%i==0 else 1 for i in range(3, area//2)])
    for i in nominated:
        n, m = area//i, i
        if 2*n+2*m-4==brown:
            answer=[max(n, m), min(n, m)]
            return answer
```

```
# 다른 사람 풀이
def solution(brown, red):
    for i in range(1, int(red**(1/2))+1):
        if red % i == 0:
            if 2*(i + red//i) == brown-4:
                return [red//i+2, i+2]
```

탐욕법(Greedy)

[체육복]

https://programmers.co.kr/learn/courses/30/lessons/42862?language=python3

문제 분류 : 탐욕법

문제 상황

학생수 n, 체육복을 도난단한 학생들의 번호가 담긴 배열 lost, 여벌의 체육복을 가져온 학생들 번호가 담긴 배열 reserve가 주어짐.

여벌 체육복 가진 학생이 도난당한 학생들안테 체육복을 빌려줄 수 있음. 하지만 번호는 체격순이기 때문에 바로 앞번호나 바로 뒷번호의 학생에게만 빌려줄 수 있음.

체육복을 절절하게 빌려 최대한 많은 학생이 체육수업을 들을수 있게 해줘라.

조건

n은 2이상 30 이하

체육복을 도난당한 학생 배열, 여벌을 가져온 학생 배열엔 중복이 없음.

여벌 체육복이 있는 학생만 다른 학생에게 빌려줄 수 있음.

여벌 체육복 가져온 학생이 도난당했을 수도 있음. 이때, 도난은 1벌만 당했다고 가정하며, 남은 체육복이 하나기 때문에 다른 학생에게는 체육복 빌려줄 수 없음.

입력

N	lost	reserve	return
5	[2, 4]	[1, 3, 5]	5
5	[2, 4]	[3]	4
3	[3]	[1]	2

풀이 아이디어

일단 n이 30 이하이기 때문에 생각나는 어떤 방법으로 풀어도 시간, 메모리 문제는 없을 것 같음.

나는 일단 [1]*n 한 다음, 그 위에 여벌 가진 학생들 +1 해주고 도난당한 학생들 -1 해줌.

그럼 0, 1, 2로 이루어진 배열이 만들어질 텐데 여벌 체육복 가진 학생이 앞 or 뒤로 빌려주는 상황을 모두 돌려봄. 그러면 2^(2의 갯수) 만큼 돌아가면 될 거임.

코드

굳이 앞or뒤 빌려주는 상황 다 안돌려도 됨. 그냥 앞에서 부터 채워주면 됨

```
Def solution(n, lost, reserve):
    _reserve = [r for r in reserve if r not in lost]   -> reserve중 lost에 없는 얘들 == 찐reserve
    _lost = [l for l in lost if l not in reserve]      -> lost중 reserv에 없는 얘들 == 찐 lost
    for r in _reserve:
        f = r – 1       # 앞
        b = r + 1       # 뒤
        if f in _lost:    # 없으면 빌려주기
            _lost.remove(f)
        elif b in _lost:
            _lost.remove(b)
    return n - len(_lost)
```

아무래도 이 문제의 핵심은 추가 조건 부분을 잘 컨트롤 하는 것 같음

```
def solution(n, lost, reserve):
    answer = 0
    for i in range(1, n+1):
        if i not in lost: #안 잃어버린 학생
            answer += 1
        else:
            if i in reserve: #잃어버렸지만 여분도 있는 학생
                answer += 1
                reserve.remove(i)
                lost.remove(i)

    for i in lost: #잃어버리고 여분도 없어서 빌려야 하는 학생
        if i-1 in reserve:
            answer += 1
            reserve.remove(i-1)

        elif i+1 in reserve:
            answer +=1
            reserve.remove(i+1)

    return answer
```

[큰 수 만들기]

https://programmers.co.kr/learn/courses/30/lessons/42883?language=python3

문제 분류 : 탐욕법

문제 상황

어떤 숫자에서 k개의 수를 제거 했을때 얻을 수 있는 가장 큰 수를 구해라.

Ex) 1924에서 2개의 수를 제거하면 [19,12,14,92,94,24]를 만들 수 있는데 이중 제일 큰건 94.

Number는 1자리 이상 1,000,000자리 이하인 수

K는 1이상 number자릿수 미만 자연수

입력

Number	k	return
1924	2	94
1231234	3	3234
4177252841	4	775841

풀이 아이디어

일단 문제분류인 '탐욕법 ' 과 1,000,000을 봤을때(처음엔 1,000,000 자리 이하가 아니라 1,000,000 이하인줄 알았음) number를 숫자별로 쪼갠 배열에서 순열 쓰려고 했는데 1,000,000자리여서 이건 아닌거 같음. -> 아 일단 이렇게 하면 기존에 배열 순서가 달라질 수 잇음.

두번째로 생각난 풀이는 numbe를 정렬하고 앞에서 부터 k개를 뽑음. Ex) 1924 -> 1249 -> [1, 2] 얘들이 없어져야할 얘들인데 numbe에서 큰 자리수 부터 돌아가면서 있으면 제거 해줌.

-> 방금 해봤는데 이 방법은 아님. 3번쨔 예시 케이스에서 부터 에러임.

그냥 기존 number에서 k개 빼는 경우의 수를 다 돌리는거네. -> number를 쪼갠 배열 넣고 조합 돌리면 number순서 유지되면서 k개 뺀 경우의 수 모두 체크가능한데 number가 좀 커지면 시간초과남.

코드

```
def solution(number, k):
    stack = [number[0]]
    for num in number[1:]:
        while len(stack) > 0 and stack[-1] < num and k > 0:
            k -= 1
            stack.pop()
        stack.append(num)
    if k != 0:
        stack = stack[:-k]
    return ''.join(stack)
```

```
def solution(number, k):
    i=0
    while i<len(number)-1 and k>0:
        if number[i]<number[i+1]:
            number = number[:i]+number[i+1:]
            if i!=0:
                i-=1
            k-=1
        else:
            i+=1
    if k>0:
        return number[:-k]
    return number
```

조이스틱

https://programmers.co.kr/learn/courses/30/lessons/42860?language=python3

문제 분류 : 탐욕법

문제 상황

조이스틱을 최소한으로 조작해 알파벳 이름을 완성하라. 처음엔 A로만 이루어짐.

▲ : 다음 알파벳, ▼ : 이전 알파벳(A에서 아래 -> Z)

◀ : 커서를 왼쪽으로(첫번째에서 왼쪽 -> 마지막), ▶ : 커서를 오른쪽으로

Ex) AAA –위로9번-> JAA –왼쪽으로 1번 아래로 1번-> JAZ

조건

Name은 알파벳 대문자로만 이루어짐. Name의 길이는 1이상 20이하

입력

Name	return
JEROEN	56
JAN	23

풀이 아이디어

음... 일단 길이가 n인 name을 받으면 name중에 어디부터 완성할지 정하는 경우의 수가 n!이겠고.

탐욕법 돌리면 20!을 기본으로 깔고 들어가는데 가능한가?

야 이건 level2는 아닌거 같은데

코드

```
def solution(name):
    count=0
    alpha='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    d={}
    indexes=[]
    current_idx=0
    n=len(name)
    for i in range(len(alpha)):
        d[alpha[i]]=min(i,26-i)
    #print(d)
    for i in range(n):
        num=d[name[i]]
        count+=num
        if num !=0 :
            indexes.append(i)
    while True:
        if len(indexes)==0:
            break
        min_dist=99
        min_idx=0
        for it in indexes:
            min_dist2=min(abs(it-current_idx),n-abs(it-current_idx))
            if min_dist2 < min_dist:
                min_dist=min_dist2
                min_idx=it
        count+=min_dist
        indexes.remove(min_idx)
        current_idx = min_idx

    return count
```

[구명보트]

https://programmers.co.kr/learn/courses/30/lessons/42885?language=python3

문제 분류 : 탐욕법

문제 상황

무인도에서 구명보트를 이용해 탈출하려함. 구명보트는 한번에 2명씩 밖에 못타고 무게제한도 있음.

Ex) 무게제한이 100이고 사람들 몸무게가 [70,50,80,50] 이면 0번, 2번은 같이 못탐

사람들의 몸무게를 담은 배열과 무게제한이 주어질때, 모든 사람을 구출하기 위한 구명보트 최소값을 retur하라.

조건

사람은 1명이상 50,000명 이하 각 사람의 몸무게는 40kg이상 240kg이하

구명보트 무게제한은 40kg이상 240kg이하 사람들을 구출할 수 없는 경우는 주어지지 않음.

입력

People	limit	return
[70, 50, 80, 50]	100	3
[70, 80, 50]	100	3

풀이 아이디어

일단 사람들을 무거운 순으로 정렬하고 제일 무거운놈 잡고, 가벼운 순으로 훑으면서 같이 탈 수 있으면 같이 태우고, 없으면 혼자 보내고 하는 식으로 하면 되지 않을까?

-> 답은 맞는데 효율성 테스트에서 실패하네. 비효율적이진 않은거 같은뎁..

```
# 내 풀이 - 답은 맞는데 효율성 탈락.
# pop(0) 의 경우 데이터를 지우고 한칸씩 앞으로 당기기 때문에 O(1)이 아니라 O(n)이 됩니다. 그래서 people을 collections.deque()로 만들어 popleft()를 사용하면 시간초과가 나지 않고 해결됩니다.
# pop(0)가 문제였음.
def solution(people, limit):
    people.sort(reverse = True)
    count = 0
    while people:
        heavy = people.pop(0)
        find = False
        for i in range(len(people)-1, -1, -1):
            light = people[i]
            if heavy+light <= limit:
                del people[i]
                count += 1
                find = True
                break
        if not find:
            count+=1
    return count
```

```
def solution(people, limit) :                    # 그냥 아예 이렇게 인덱스로 만 다뤄도 됨
    answer = 0
    people.sort()

    a = 0
    b = len(people) - 1
    while a < b :
        if people[b] + people[a] <= limit :
            a += 1
            answer += 1
        b -= 1
    return len(people) - answer
```

[섬 연결하기]

https://programmers.co.kr/learn/courses/30/lessons/42861?language=python3

문제 분류 : 탐욕법

문제 상황

N개의 섬이 있고 각각 섬 사이에 다리를 건설하는 비용이 있음. 이때, 최소의 비용으로 모든 섬이 통행 가능하도록 다리를 건설하도록 한다면 그때 총 비용을 RETURN 해라.

다리를 여러 번 건너도 됨

조건

섬의 개수는 1이상 100 이하. Costs길이는 n(n-1)/2 이하

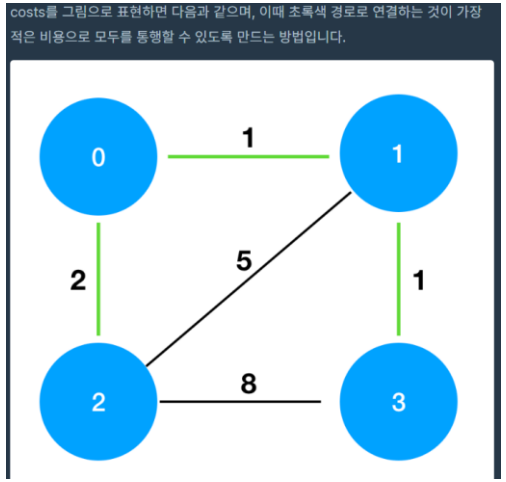
Costs[i][0], cost[i][1]에는 각각 섬 본호가, cost[i][2]엔 비용이 있음.

같은 연결은 두번 주어지지 않고 순서가 바뀌어도 같은 연결로 봄.(양방향 도로)

어느 두 섬 사이 비용이 안주어지는 경우도 있음.(두 섬을 연결할 수 없는 경우)

입력

N	costs	return
4	[[0,1,1],[0,2,2],[1,2,5],[1,3,1],[2,3,8]]	4



풀이 아이디어

Level 3 치고는 좀 어려운거 같은데 이거 그 다익스트라 써야되는거 아닌가? 유니온 파인드랑

다른 사람들 풀이 보니까 union-find 맞는거 같음.

```
parent={}#각 노드의 부모
rank={}#트리의 노드 수
def make_set(v):#각 노드를 집합으로 만들기
    parent[v]=v#일단 부모는 자기 자신
    rank[v]=0#
def findRoot(v):
    if parent[v]!=v:#부모가 자기 자신이 아니면
        parent[v]=findRoot(parent[v])#최상위의 부모로 갱신
    return parent[v]#부모가 자기 자신이라면 최상위이므로 반환
def union(r1,r2):
    if r1!=r2:#루트값이 서로 다르면 다른 집합임
        if rank[r1]>rank[r2]:#노드 수가 적은 집합의 루트가 노드 수가 많은 집합의 루트로 변경됨
            parent[r2]=r1
        else:
            parent[r1]=r2
            if rank[r1]==rank[r2]:#집합의 개수가 같다면
                rank[r2]+=1# r1이 속한 집합의 부모 루트가 r2로 변경되었으므로 r2의 개수를 더 많다고 해주기
def solution(n,costs):
    for i in range(n):#모든 노드에 대해 집합화
        make_set(i)
    #mst=[]#최소 비용 신장(spanning) 트리
    s=0#최소 비용 누적을 위한 변수
    costs=sorted(costs,key=lambda costs:costs[2])#비용 기준으로 정렬
    for j in costs:
        v,u,w=j# v=노드1 u=노드2 w=비용
        r1=findRoot(v)#노드 v에 대한 루트
        r2=findRoot(u)
        if r1!=r2:#노드의 루트가 다르면
            union(r1,r2)#두 노드 중 하나의 집합 수가 많은 집합에 넣기
            s+=w
            #mst.append(j)
    return s
```

[단속카메라]

https://programmers.co.kr/learn/courses/30/lessons/42884?language=python3

문제 분류 : 탐욕법

문제 상황

Routes배열에 차들의 진입 지점과 진출 지점이 배열로 들어있음. 이때, 모든 차들이 단속 카메라를 최소 한 번은 만나도록 카메라를 설치하려고 할때, 최소 몇대의 카메라를 설치해야 하는지 구해라

ex) [[-20,15], [-14,-5], [-18,-13], [-5,-3]] 에서 0번째 차량은 -20에서 들어오고 15에서 나갔다는 뜻.

이렇게 할때 0~3번째 차량이 모두 카메라를 한번씩은 거치게 하기 위해선 -5, -15에 설치하면 됨.

조건

차량 대수는 1대 이상 10,000대 이하

차량 진입/진출 지점에 카메라가 있어도 카메라를 만난것으로 간주

진입/진출 지점은 -30,000이상 30,000이하임.

입력

Routes	return
[[-20,15], [-14,-5], [-18,-13], [-5,-3]]	2

풀이 아이디어

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

https://programmers.co.kr/learn/courses/30/lessons/42840?language=python3

문제 분류 : 완전 탐색

문제 상황
1번 수포자 : 1, 2, 3, 4, 5 반복으로 찍음
2번 수포자 : 2, 1, 2, 2, 2, 3, 2, 4, 2, 5 반복으로 찍음
3번 수포자 : 3, 3, 1, 1, 2, 2, 4, 4, 5, 5 반복으로 찍음
위 상황에서 답안지의 역할을 하는 배열이 입력으로 주어짐. 원소 개수는 10,000 이하
이때, 가장 많은 답을 맞춘 사람을 출력. 같으면 둘 다 출력

입력
1~5범위의 10,000개 이하의 원소를 갖는 배열

풀이 아이디어
10,000개 이하 -> 수포자 3명 다 돌려도 매우 적은 범위.

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

동적계획법

https://programmers.co.kr/learn/courses/30/lessons/42840?language=python3

문제 분류 : 완전 탐색

문제 상황
1번 수포자 : 1, 2, 3, 4, 5 반복으로 찍음
2번 수포자 : 2, 1, 2, 2, 2, 3, 2, 4, 2, 5 반복으로 찍음
3번 수포자 : 3, 3, 1, 1, 2, 2, 4, 4, 5, 5 반복으로 찍음
위 상황에서 답안지의 역할을 하는 배열이 입력으로 주어짐. 원소 개수는 10,000 이하
이때, 가장 많은 답을 맞춘 사람을 출력. 같으면 둘 다 출력

입력
1~5범위의 10,000개 이하의 원소를 갖는 배열

풀이 아이디어
10,000개 이하 -> 수포자 3명 다 돌려도 매우 적은 범위.

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

https://programmers.co.kr/learn/courses/30/lessons/42840?language=python3

문제 분류 : 완전 탐색

문제 상황
1번 수포자 : 1, 2, 3, 4, 5 반복으로 찍음
2번 수포자 : 2, 1, 2, 2, 2, 3, 2, 4, 2, 5 반복으로 찍음
3번 수포자 : 3, 3, 1, 1, 2, 2, 4, 4, 5, 5 반복으로 찍음
위 상황에서 답안지의 역할을 하는 배열이 입력으로 주어짐. 원소 개수는 10,000 이하
이때, 가장 많은 답을 맞춘 사람을 출력. 같으면 둘 다 출력

입력
1~5범위의 10,000개 이하의 원소를 갖는 배열

풀이 아이디어
10,000개 이하 -> 수포자 3명 다 돌려도 매우 적은 범위.

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

https://programmers.co.kr/learn/courses/30/lessons/42840?language=python3

문제 분류 : 완전 탐색

문제 상황
1번 수포자 : 1, 2, 3, 4, 5 반복으로 찍음
2번 수포자 : 2, 1, 2, 2, 2, 3, 2, 4, 2, 5 반복으로 찍음
3번 수포자 : 3, 3, 1, 1, 2, 2, 4, 4, 5, 5 반복으로 찍음
위 상황에서 답안지의 역할을 하는 배열이 입력으로 주어짐. 원소 개수는 10,000 이하
이때, 가장 많은 답을 맞춘 사람을 출력. 같으면 둘 다 출력

입력
1~5범위의 10,000개 이하의 원소를 갖는 배열

풀이 아이디어
10,000개 이하 -> 수포자 3명 다 돌려도 매우 적은 범위.

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

https://programmers.co.kr/learn/courses/30/lessons/42840?language=python3

문제 분류 : 완전 탐색

문제 상황
1번 수포자 : 1, 2, 3, 4, 5 반복으로 찍음
2번 수포자 : 2, 1, 2, 2, 2, 3, 2, 4, 2, 5 반복으로 찍음
3번 수포자 : 3, 3, 1, 1, 2, 2, 4, 4, 5, 5 반복으로 찍음
위 상황에서 답안지의 역할을 하는 배열이 입력으로 주어짐. 원소 개수는 10,000 이하
이때, 가장 많은 답을 맞춘 사람을 출력. 같으면 둘 다 출력

입력
1~5범위의 10,000개 이하의 원소를 갖는 배열

풀이 아이디어
10,000개 이하 -> 수포자 3명 다 돌려도 매우 적은 범위.

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

https://programmers.co.kr/learn/courses/30/lessons/42840?language=python3

문제 분류 : 완전 탐색

문제 상황
1번 수포자 : 1, 2, 3, 4, 5 반복으로 찍음
2번 수포자 : 2, 1, 2, 2, 2, 3, 2, 4, 2, 5 반복으로 찍음
3번 수포자 : 3, 3, 1, 1, 2, 2, 4, 4, 5, 5 반복으로 찍음
위 상황에서 답안지의 역할을 하는 배열이 입력으로 주어짐. 원소 개수는 10,000 이하
이때, 가장 많은 답을 맞춘 사람을 출력. 같으면 둘 다 출력

입력
1~5범위의 10,000개 이하의 원소를 갖는 배열

풀이 아이디어
10,000개 이하 -> 수포자 3명 다 돌려도 매우 적은 범위.

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

https://programmers.co.kr/learn/courses/30/lessons/42840?language=python3

문제 분류 : 완전 탐색

문제 상황
1번 수포자 : 1, 2, 3, 4, 5 반복으로 찍음
2번 수포자 : 2, 1, 2, 2, 2, 3, 2, 4, 2, 5 반복으로 찍음
3번 수포자 : 3, 3, 1, 1, 2, 2, 4, 4, 5, 5 반복으로 찍음
위 상황에서 답안지의 역할을 하는 배열이 입력으로 주어짐. 원소 개수는 10,000 이하
이때, 가장 많은 답을 맞춘 사람을 출력. 같으면 둘 다 출력

입력
1~5범위의 10,000개 이하의 원소를 갖는 배열

풀이 아이디어
10,000개 이하 -> 수포자 3명 다 돌려도 매우 적은 범위.

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

https://programmers.co.kr/learn/courses/30/lessons/42840?language=python3

문제 분류 : 완전 탐색

문제 상황
1번 수포자 : 1, 2, 3, 4, 5 반복으로 찍음
2번 수포자 : 2, 1, 2, 2, 2, 3, 2, 4, 2, 5 반복으로 찍음
3번 수포자 : 3, 3, 1, 1, 2, 2, 4, 4, 5, 5 반복으로 찍음
위 상황에서 답안지의 역할을 하는 배열이 입력으로 주어짐. 원소 개수는 10,000 이하
이때, 가장 많은 답을 맞춘 사람을 출력. 같으면 둘 다 출력

입력
1~5범위의 10,000개 이하의 원소를 갖는 배열

풀이 아이디어
10,000개 이하 -> 수포자 3명 다 돌려도 매우 적은 범위.

코드

```
def solution(answers):
    p = [[1, 2, 3, 4, 5],
         [2, 1, 2, 3, 2, 4, 2, 5],
         [3, 3, 1, 1, 2, 2, 4, 4, 5, 5]]
    s = [0] * len(p)

    for q, a in enumerate(answers):
        for i, v in enumerate(p):                    # 여기가 중요
            if a == v[q % len(v)]:                   # 여기가 중요
                s[i] += 1
    return [i + 1 for i, v in enumerate(s) if v == max(s)]
```

BFS/DFS

[타겟 넘버]

https://programmers.co.kr/learn/courses/30/lessons/43165?language=python3

문제 분류 : BFS/DFS

문제 상황

n개의 음이 아닌 정수 배열을 줌. 그 정수 배열을 +-로 조합해서 타겟 넘버를 만들 수 있는 경우의 수를 리턴하면 됨.

[1, 1, 1, 1, 1]이 입력된다면

-1+1+1+1+1 = 3

+1-1+1+1+1 = 3

+1+1-1+1+1 = 3

+1+1+1-1+1 = 3

+1+1+1+1-1 = 3

5가지의 경우의 수가 있는 것.

조건

배열의 원소는 2개이상, 20개 이하. 각 원소는 1이상 50이하

타겟넘버는 1이상 1000이하.

입력

Numbers = [1, 1, 1, 1, 1], target = 3, return = 5

풀이 아이디어

일단 범위 적당함. 20개 이하에 +, - 조합해주려면 최대 2^20개의 경우의 수가 나옴 2^20=1,048,576 컴터에겐 부담스럽지 않은 수임. 2^n개의 경우의 수 하나마다 타겟넘버 인지아닌지 구하려면 또 각각 n을 돌아야됨. 그래서 이런 식으로 구성하면 n*2^n 의 복잡도가 나옴. 복잡도 자체는 크지만 n이 작아서 할만할거 같긴함. 2^n가지 경우의수 만드는 과정이 BFS/DFS일거 같고.

코드

```
# BFS/DFS를 이용한 가장 정석적인 풀이
import collections

def solution(numbers, target):
    answer = 0
    stack = collections.deque([(0, 0)])
    while stack:
        current_sum, num_idx = stack.popleft()

        if num_idx == len(numbers):
            if current_sum == target:
                answer += 1
        else:
            number = numbers[num_idx]
            stack.append((current_sum+number, num_idx + 1))
            stack.append((current_sum-number, num_idx + 1))

    return answer
```

```
# 재귀 써도 괜찮을 정도로 범위가 작은 문제.
# 이 문제는 일단 이게 베스트 인듯.
# 놀랍다. 아름답다. 와우.
def solution(numbers, target):
    if not numbers and target == 0 :
        return 1
    elif not numbers:
        return 0
    else:
        return solution(numbers[1:], target-numbers[0]) + solution(numbers[1:], target+numbers[0])
```

```
# 이것도 찢었다...
# produc는 곱집합임.  오른쪽 처럼 작동하는
from itertools import product
def solution(numbers, target):
    l = [(x, -x) for x in numbers]
    s = list(map(sum, product(*l)))
    return s.count(target)
```

```
it1 = ['x', 'y', 'z']
it2 = [1, 2, 3]
for i in product(it1, it2):
    print(i)

('x', 1)
('x', 2)
('x', 3)
('y', 1)
('y', 2)
('y', 3)
('z', 1)
('z', 2)
('z', 3)
```

파이썬의 곱집합

map

배열앞에 * 곱하는 거

셋 다 알고 있어야됨. Map에서 첫번째 인자를 뭐를 주냐에 따라 map의 활용도가 많이 달라지네. 파이썬 고수용 풀이네.

[네트워크]

https://programmers.co.kr/learn/courses/30/lessons/43162?language=python3

문제 분류 : BFS/DFS

문제 상황

컴퓨터의 개수 n, 연결에 대한 정보가 담긴 2차원 배열 computeres가 주어질때

네트워크의 개수를 구하는 문제.

연결된 컴퓨터의 수가 아니라 연결된 컴퓨터들의 묶음 단위로 네트워크 개수를 구하는 문제.

입력

n : 3, computers : [[1, 1, 0], [1, 1, 0], [0, 0, 1]], return : 2

풀이 아이디어

어떤 노드를 주고 그 노드에 연결된 개수를 구하는게 아니라 묶음 단위를 구하는 걸로 보아

Union-find 알고리즘을 써야되는거 같음.

아니다. 다른 사람들 풀이보고 union-find까진 필요 없다는 걸 알았음.

코드

```
# 정확히 union-find는 아니지만 비슷한 개념을 사용한 것 같음.
# temp라는 배열의 역할을 잘 보자. 부모 배열을 나타내는것 같음.
def solution(n, computers):
    temp = []
    for i in range(n):
        temp.append(i)              # 처음엔 temp = [I for I in range(n)] 으로 자기 자신을 보다가
    for i in range(n):
        for j in range(n):
            if computers[i][j]:         # 자기(i)랑 연결된애(j)가 나오면
                for k in range(n):
                    if temp[k] == temp[i]:  # 이부분이 좀 헷갈리는데..
                        temp[k] = temp[j]   # 자기의 부모를 연결된 애로 잡는 것 같음.
    return len(set(temp))
```

일반적인 풀이

컴퓨터 개수만큼 visited만들어주고 한번 dfs돌때마다 거친애들은 visited = 1 해줌

그리고 이 dfs가 한번 돌아갈때마다 answer+=1을 해주면 됨.

```
def solution(n, computers):
    answer = 0
    visited = [0 for i in range(n)]
    def dfs(computers, visited, start):
        stack = [start]
        while stack:
            j = stack.pop()
            if visited[j] == 0:
                visited[j] = 1
            # for i in range(len(computers)-1, -1, -1):
            for i in range(0, len(computers)):
                if computers[j][i] ==1 and visited[i] == 0:
                    stack.append(i)
    i=0
    while 0 in visited:
        if visited[i] ==0:
            dfs(computers, visited, i)
            answer +=1
        i+=1
    return answer
```

[단어 변환]

https://programmers.co.kr/learn/courses/30/lessons/43163?language=python3

문제 분류 : DFS/BFS

문제 상황

예시와 함께 이해하자.

시작점과 목적지가 될 두개의 단어와 중간 다리가 될 단어들의 배열을 줌.

begin = "hit", target = "cog", words = ["hot", "dot", "dog", "lot", "log", "cog"]

각 단어에서 단어로 변환할땐 알파벳 하나만 바꿀 수 있음.

hit -> hot (가능), hit -> dot(불가능)

이때, begin 단어를 몇 단계 걸쳐 targe의 단어로 변환할 수 있을까. 최소 단계수를 구하세요.

조건

각 단어는 3이상, 10 이하, 각각 단어 길이는 같음.

Words는 3개이상, 50개 이하, 중복 없음

Begin과 target은 다름. 반환 할 수 없으면 0 리턴

입력

Begin	target	words	return
hit	cog	[hot, dot, dog, lot, log, cog]	4
hit	cog	[hot, dot, dog, lot, log]	0

풀이 아이디어

일단 begin도 words안에 넣고, words안에서 그래프를 생성해주자. Nlog(n)으로 그래프 만들 수 있을 거 같고, 양방향으로, 알파벳 하나만 다르다면 연결.

그 다음 begin노드에서 targe노드까지 DFS돌리는 거임. 이때 그냥 DFS 돌리는게 아니라

백준에서 엣지에 코스트 줄 때 처럼 단계수를 기록해가면서 더 적은단계가 나왔을땐 그 단계로 갈아치우기하면서.

일단 적당한 풀이 두개 가져왔는데

내 의문은 이렇게 그냥 DFS 돌리면 DFS 돌아가는 순서에 따라서 최단 경로가 아닌 걸로 target을 잡을 수도 있는거 아닌가?

그래도 일단 이 두 풀이모두 주목할 점은 내가 생각한 대로 그래프를 먼저 그리는 일은 안했다는거

그리고 두 단어가 한 알파벳만 다르다는걸 체크할 때 쓴 방법이 다름.

위 풀이는 슬라이싱을 활용해서 다른 단어를 제외하고 남은 것 끼리 비교했고

아래 풀이는 캐릭터별로 잡아서 다른 것의 개수를 셌음. 아래 풀이가 빠를거 같긴 함.

```
def change(fr, to):
    for i in range(len(fr)):
        if fr[:i]+fr[i+1:] == to[:i]+to[i+1:]:
            return True
    return False

def permutation(begin, target, words, cnt):
    global answer
    if begin == target:
        answer = cnt
    else:
        for i in range(len(words)):
            if change(begin, words[i]):
                else_words = words[:i] + words[i+1:]
                permutation(words[i], target, else_words, cnt+1)

def solution(begin, target, words):
    if target not in words:
        return 0
    permutation(begin, target, words, 0)
    return answer
```

```
def solution(begin, target, words):
    answer = 0
    Q = [begin]

    while True:
        temp_Q = []
        for word_1 in Q:
            if word_1 == target:
                    return answer
            for i in range(len(words)-1, -1, -1):
                word_2 = words[i]
                if sum([x!=y for x, y in zip(word_1, word_2)]) == 1:
                    temp_Q.append(words.pop(i))

        if not temp_Q:
            return 0
        Q = temp_Q
        answer += 1
```

[여행경로]

https://programmers.co.kr/learn/courses/30/lessons/43164?language=python3

문제 분류 : DFS/BFS

문제 상황

비행경로 [from, to]가 담긴 배열을 받음. 각 [from, to]가 하나의 티켓이라고 보면 됨.

이때, 주어진 티켓을 모두 써서 방문하는 공항의 경로를 배열로 리턴해라

조건

모든 도시를 방문할 수 없는 경우는 주어지지 않음.

공항은 알파벳 3글자로 주어짐. 공항수는 3개 이상 10,000개 이하

주어진 티켓은 모두 써야 함.

가능한 경로가 2개 있다면 알파벳 순서가 앞서는 순서를 return함.

입력

Tickets	return
[[ICN, JFK], [HND, IAD], [JFK, HND]]	[ICN, JFK, HND, IAD]
[[ICN, SFO], [ICN, ATL], [SFO, ATL], [ATL, ICN], [ATL,SFO]]	[ICN, ATL, ICN, SFO, ATL, SFO]

** 2번째 예시의 경우 [ICN, SFO, ATL, ICN, ATL, SFO] 도 가능하지만 알파벳 순에서 뒤져서 채택이 안됨

풀이 아이디어

문제 자체는 간단한데 조건이 덕지덕지 붙은 문제네.

단 방향 그래프에서 전체 그래프를 다 돌아야 되는거지. 공항수가 많아서 재귀 돌리기 힘들거 같기도 하고.. 재귀 써서 풀수 있는 가장 간단한 풀이는 각 재귀에 경로를 담아서 보내고 재귀 끝날때 받은 경로가 공항 개수랑 같으면 후보에 올려 놓은 다음 후보들 사이에서 알파벳 순으로 앞서는거 잡으면 되는데.

그냥 재귀 돌려도 되네

```
candidates = []
def visit(start, graph, visited, cnt, route):
    global candidates
    if cnt == len(graph):
        candidates.append(route.split(" "))
    else:
        for i in range(len(graph)):
            if visited[i] == 0 and graph[i][0] == start:
                go = []
                for j in range(len(visited)):
                    go.append(visited[j])
                go[i] = 1
                visit(graph[i][1], graph, go, cnt+1, route+" "+graph[i][1])

def solution(tickets):
    answer = []
    tickets.sort()
    visited = [0] * len(tickets)
    visit("ICN", tickets, visited, 0, "ICN")
    return candidates[0]
```

풀이 구조는 같되, 재귀를 안돌림.

```
def solution(tickets):
    routes = {}
    for t in tickets:
        routes[t[0]] = routes.get(t[0], []) + [t[1]]
    for r in routes:
        routes[r].sort(reverse=True)
    stack = ["ICN"]
    path = []
    while len(stack) > 0:
        top = stack[-1]
        if top not in routes or len(routes[top]) == 0:
            path.append(stack.pop())
        else:
            stack.append(routes[top][-1])
            routes[top] = routes[top][:-1]
    return path[::-1]
```

이분탐색

[예산]

https://programmers.co.kr/learn/courses/30/lessons/43237?language=python3

문제 분류 : 이분 탐색

문제 상황

n개의 지방에서 예산 요청이 들어오고 나에겐 총 예산 m이 있음. 이때, n개 지방의 예산 요청을 다 들어줄 수 있으면 좋지만 문제에서 그런 상황은 배제한 것 같음(다 들어줄 수 있는 경우엔 뭘 리턴하라는지 안나와있음)

암튼, n개의 예산 요청을 다 못들어 줄 경우 상한액을 정해야 함. 상한액 이하의 예산 요청은 그대로 들어주되, 상한액 이상의 예산 요청은 상한액 까지만 주는 거임. 이렇게 예산을 분배해서 총 예산 m에 들어오면 됨.

문제에 조건이 막 빠져있는거 같네.

주의 조건

지방의 수는 3개 이상 100,000이하임.

각 지방의 요청 예산은 1 이상 100,000 이하임

총 예산은 n이상, 1,000,000,000 이하.

입력

예산요청: [120, 110, 140, 150], m : 485, return : 127

풀이 아이디어

일단 조건의 범위만 봐도 O(n^2) 이상 가면 힘들어 진다는 걸 직감할 수 있어야 함.

예산요청, 총 예산, 상한액 3개를 파라미터로 받아서 예산 집행 가능한지, 가능하다면 그때 총 배정 예산은 얼마나왔는지를 뱉어주는 함수를 만들고

상한액을 min, max, mid 잡고 이분 탐색 돌리는 문제 같음.

근데 사람들은 그렇게 안풀었네 ㅎㅎ

코드

```
def solution(budgets, M):
    budgets.sort()
    l = len(budgets)
    cap = 0
    for i, budget in enumerate(budgets):
        level = (budget - cap) * (l - i)
        if level <= M:
            cap = budget
            M -= level
        else:
            cap += M // (l - i)
            break
    return cap
```

가장 간단한데 지금은 시간초과 난다함.

```
def solution(budgets, M):
    lim=max(budgets)
    while True:
        if sum(budgets)<M:
            answer=max(budgets)
            break

        budgets=list(map(lambda x: x if x<=lim else lim,budgets))
        lim=lim-1

    return answer
```

내가 풀었다면 이런식으로 나왔을 거 같음

```
def solution(budgets, M):
    if sum(budgets) <= M:
        return max(budgets)

    l, r, mid = 1, max(budgets), 0
    answer = 0

    while l <= r:
        mid = (l+r) // 2
        total = 0

        for budget in budgets:
            if budget <= mid:
                total += budget
            else:
                total += mid

        if total > M:
            r = mid - 1
        else:
            if answer <= mid:
                answer = mid
            l = mid + 1
    return answer
```

[입국심사]

https://programmers.co.kr/learn/courses/30/lessons/43238?language=python3

문제 분류 : 이분 탐색

문제 상황
공항에 심사대기자 n명이 있고 심사관 m명이 있음. m명의 심사관은 각각 심사하는데 걸리는 시간이 다름.
심사 대기자들은 한줄서기 하고 있다가 비는 곳에 각각 가서 심사 받으면 됨.
이렇게 할때 n명을 모두 심사할 수 있는 최단 시간을 구하시오

조건
n은 1이상 1,000,000,000명 이하
각 심사관이 심사하는데 걸리는 시간은 1분이상, 1,000,000,000분 이하
심사관은 1명이상 100,000명 이하

입력
n: 6, times = [7, 10], return : 28
6명이 있고 심사관은 2명, 각각 심사에 7, 10초가 걸림. 이때 최소 시간은 28분.

풀이 아이디어
이것 역시 숫자가 어마어마 함.
시간을 t라고 가정하면 이때까지 각 심사관들이 심사한 사람의 수를 t//i for i in times로 구할 수 있음. t//i들의 합이 n보다 크면 그 시간에 끝낼 수 있는 것.
이제 시간 t를 이분탐색으로 찾으면 되는거 같음.

코드

```
def solution(n, times):
    answer = 0
    start, end, mid = 1, times[-1] * n, 0

    while start < end:          ##### 종료조건 잘 보고
        mid = (start + end) // 2
        total = 0
        for time in times:
            total += mid // time

        if total >= n:
            end = mid       ###### end를 잡을땐 그냥 미드를 주고
        else:
            start = mid + 1 ##### 여기가 중요한거 같음 그냥 mid로 두는게 아니라 mid+1로 두는거
    answer = start
    return answer
```

코드 되게 간단하네

[징검다리](https://programmers.co.kr/learn/courses/30/lessons/43236?language=python3)

문제 분류 : 이분 탐색

문제 상황
출발지점에서 d만큼 떨어진 곳에 도착지점이 있다. 그리고 그 사이엔 m개의 바위들이 있음. 각 바위들의 위치는 배열로주어짐. 여기서 n개의 바위를 제거할때, 제거 후 각 바위들의 거리들 중 최솟값이 가장 크게 만들어라. 거리의 최솟값들중 가장 큰 값을 리턴하면 됨.

조건
D는 1이상 ~ 1,000,000,000 이하임.
바위는 1개 이상 ~ 50,000개 이하
제거할 바위는 1이상 ~ 바위갯수 이하.

입력
d : 25, rocks : [2, 14, 11, 21, 17], n: 2, return : 4

풀이 아이디어
딱 봐도 백준에서 푼 공유기 문제랑 비슷한 구조임. 최소 간격을 설정하고, 그 간격으로 노드들을 배치할 수 있는지 확인하고 되는 지 안 되는지에 따라 이분탐색으로 최소 간격 다시 잡고.
다만, 공유기 문제랑 다른 점은 기존에 있는 바위들에서 n개의 바위를 제거 해야 한다는 것.
n개 바위 제거하고, 최소 간격 d로 배치할 수 있는가를 리턴하는 함수만 파면
나머지는 쉬울 거 같음.

코드

```
def solution(distance, rocks, n):
    answer = 0
    sorted_rocks = sorted(rocks)    # 일단 받은 돌들 정렬
    sorted_rocks.append(distance)  # 마지막에 도착지 돌 추가
    left = 0                                # 이분탐색용 초기값 설정
    right = distance                    # 이분탐색용 초기값 설정
    while (left <= right):        # 종료조건 어떻게 두는지 잘 확인해두자.
        mid = int((left + right) / 2)  # 이분탐색 값 설정
########################################
        cnt = 0                          # 현재 설정한 이분탐색값으로 가능한지 판별해줄 count
        p = 0                            # 이전 돌
        for I in range(len(sorted_rocks)):   ###
            if (sorted_rocks[i] – p < mid):    # 현재 돌(sorted_rocks[i]) 과 이전돌(p)의 간격이
                cnt += 1                          mid가 넘으면 빼도 되는돌이라 count올려줌
            else:
                p = sorted_rocks[i]            # 그게 아니면 p에 이전돌을 다시 할당하고 넘어감
        if cnt > n:                           # 건너 뛸 수 있는 돌이 n개보다 많다면
            right = mid – 1                   # 간격을 좁힘. 여기선 미드에 -1을 줘서 할당하고
        else:                                # 건너뛸수 있는 돌이 n개보다 적으면
            left = mid + 1                 # 간격을 넓힘. 미드에 +1을 줘서 할당했네
            answer = mid
########################################
    return answer
```

내가 생각해논 구조인데 따로 함수를 판 건 아니고 바로 적은 것.
그리고 '전체에서 n개의 돌을 뺀다'는 걸 그대로 받아들이지 않고 '전체에서 n개의 돌을 뺄수 있는가'로 바꿈. Very Clever한 풀이네

그래프

[가장 먼 노드]

https://programmers.co.kr/learn/courses/30/lessons/49189?language=python3

문제 분류 : 그래프

문제 상황

n개의 노드가 있고 간선에 대한 정보가 담긴 배열을 줌. 간선은 양방향.

이때, 1번 노드에서 가장 멀리 떨어진 노드의 개수를 구하면 됨

가장 멀리 떨어졌다 = 지나온 간선이 가장 많다.

조건

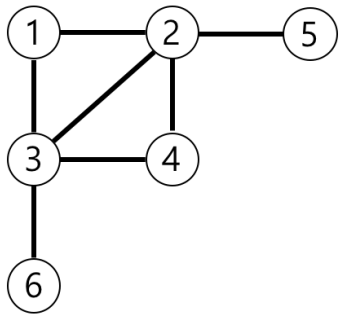
노드의 개수 n은 2이상 20,000 이하

간선은 양방향, 총 1개 이상 50,000개 이하의 간선이 있음.

입력

n	vertex	return
6	[[3, 6], [4, 3], [3, 2], [1, 3], [1, 2], [2, 4], [5, 2]]	3

예시 같은 경우, 1노드에서 가장 멀리떨어진 노드는 4, 5, 6 3개임.



풀이 아이디어

visited 배열을 true/false로 쓰지 말고 지금까지 거쳐온 간선 수로 잡으면 될듯. 지나온 간선 개수를 재귀에 계속 태워 보냄.

visited[i]가 0이면 visted[i]에 지나온 간선 개수를 주고

visited[i]가 0이 아닐땐, 지금 온 간선수가 입려되어 있는 것보다 작다면 갈아치워주고 재귀 계속 진행, 지금 온 간선수가 입력되어 있는 것보다 크면 재귀 중단.

코드

```
def solution(n, edge):
    graph =[  [] for _ in range(n + 1) ]
    distances = [ 0 for _ in range(n) ]
    is_visit = [False for _ in range(n)]
    queue = [0]
    is_visit[0] = True
    for (a, b) in edge:
        graph[a-1].append(b-1)
        graph[b-1].append(a-1)

    while queue:
        i = queue.pop(0)

        for j in graph[i]:
            if is_visit[j] == False:
                is_visit[j] = True
                queue.append(j)
                distances[j] = distances[i] + 1

    distances.sort(reverse=True)
    answer = distances.count(distances[0])

    return answer
```

근데 이 DFS/BFS에서도 같은 의문이었는데 이렇게 단순하게 짜면 최단 거리가 아닌 얘가 먼저 도착하는 경우는 배제할 수 있나? BFS라서 이렇게 해도 되는건가?

[순위](https://programmers.co.kr/learn/courses/30/lessons/49191?language=python3)

https://programmers.co.kr/learn/courses/30/lessons/49191?language=python3

문제 분류 : 그래프

문제 상황

권투선수의 수 n과 경기결과가 담긴 배열이 주어짐. 각 권투선수는 1번~n번을 부여받고, 경기결과 배열의 원소 [a, b]는 a선수가 b선수를 이겼다는 뜻임.

이때, 주어진 경기결과로 정확히 순위를 매길 수 있는 선수의 수를 retur하시오.

조건

1<= n <= 100, 1<= len(경기결과) <= 4,500

주어진 경기결과엔 모순이 없음.

입력

N	result	return
5	[[4, 3], [4, 2], [3, 2], [1, 2], [2, 5]]	2

풀이 아이디어

오 신선해. 처음으로 아디이어에서 막히는 문제네.

이거 좀 있다 다시 보자.

코드

```
def solution(n, results):
    fight = [[set(), set(), 0] for _ in range(n)]
    for win, lose in results:
        fight[win - 1][1].add(lose - 1)
        fight[win - 1][2] += 1
        fight[lose - 1][0].add(win - 1)
        fight[lose - 1][2] += 1
    for i, (win, lose, cnt) in enumerate(fight):
        for w in win:
            fight[w][1].add(i)
            fight[w][1].update(lose)
            fight[w][2] = len(fight[w][1]) + len(fight[w][0])

        for l in lose:
            fight[l][0].add(i)
            fight[l][0].update(win)
            fight[l][2] = len(fight[l][1]) + len(fight[l][0])

    return len(list(filter(lambda x: x[2] == n-1, fight)))
```

```
def solution(n, results):
    answer = 0
    matrix =  [ [ 0 for _ in range(n+1) ] for _ in range(n+1) ]
    for (win, lose) in results: matrix[win][lose],matrix[lose][win] = win,win

    for _ in range(2):
        for win in range(1,n+1):
            for lose in range(1,n+1):
                if win == matrix[win][lose]:
                    for i, j in enumerate(matrix[lose]):
                        if j == lose: matrix[win][i],matrix[i][win] = win,win

    for i in range(1, n+1):
        if matrix[i].count(0)-1 == 1: answer += 1

    return answer
```

```
def solution(n, results):
    answer = 0
    graph = [[0 for cols in range(n)] for rows in range(n)]
    for r in results:   #그래프 초기화
        graph[r[0]-1][r[1]-1] = 1
        graph[r[1]-1][r[0]-1] = -1
    for i in range(n):  #각 행별로 처리
        for j in range(n):
            if graph[i][j] == 1:
                for m in range(n):
                    if graph[j][m] == 1 and graph[i][m] == 0:
                        graph[i][m] = 1
                        graph[m][i] = -1
            elif graph[i][j] == -1:
                for m in range(n):
                    if graph[j][m] == -1 and graph[i][m] == 0:
                        graph[i][m] = -1
                        graph[m][i] = 1
    for i in range(n):
        if graph[i].count(0) == 1:
            answer += 1
    return answer
```

```
from collections import defaultdict
def solution(n, results):
    player_link = [0]*n  # 선수들의 상관관계수 기록 리스트
    그래프 그리기 일반적인 dict를 사용하여도 된다
    graph = defaultdict(list)
    for r in results:
        if r[1] not in graph: graph[r[1]] = []
        graph[r[0]].append(r[1])

    for g in graph:
        '''
        노드를 방문해나가면서 그래프에 상관관계를 기록함
        '''
        queue = graph[g].copy()
        while queue:
            current = queue.pop()
            for _current in graph[current]:
                if _current not in graph[g]:  # if-in을 사용하는 경우 시간복잡도에 있어서 그리
                    graph[g].append(_current)
                    queue.append(_current)
        상관관계수 리스트에 기록
        player_link[g-1]+=len(graph[g])
        for i in graph[g]: player_link[i-1] += 1

    answer = player_link.count(n-1)

    return answer
```

[사이클 제거]

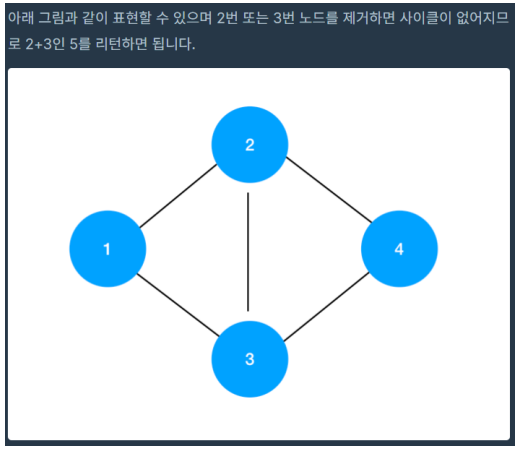
https://programmers.co.kr/learn/courses/30/lessons/49188?language=java

문제 분류 : 그래프

문제 상황

노드의 개수 n, 연결 정보 edges가 주어짐. 이때 노드 딱 하나만 제거해 그래프 사이 사이클을 제거할 수 있다면 그 노드의 번호를 retrun함.

단, 노드가 여러 개라면 노드들의 합을 retur하고, 어느 노드를 지워도 사이클이 안없어진다면 0을 리턴함.



조건

노드번호는 1부터 시작, 노드는 2개 이상 5,000개 이하

연결 정보는 1개 이상, n(n+1)/2개 이하

노드간 연결에 방향 없음

주어진 그래프엔 반드시 하나 이상의 사이클이 있음.

입력

N	edges	return
4	[[1,2],[1,3],[2,3],[2,4],[3,4]]	5
8	[[1,2],[2,3],[3,4],[4,5],[5,6],[6,7],[7,8],[8,1],[2,7],[3,6]]	0

풀이 아이디어

야 확실히 level 4 느낌 나네. 일단 한 그래프에서 사이클을 검출하는 것부터 해야겠네.

사이클 검출하는 함수 만들고

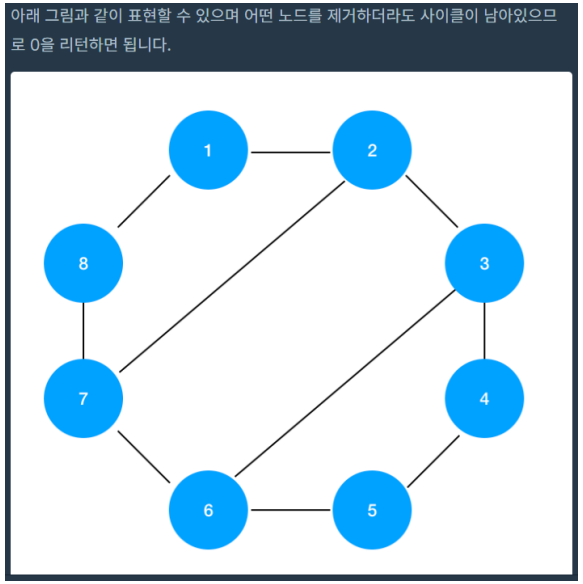
노드를 하나씩 빼보면서 사이클 검출 함수를 계속 돌리면 되나?

시간이 너무 걸리지 않을까.

코드 - 이건 python3 풀이가 없어서 java 풀이로 대체함.
그나마 짧은게 이정도

```
import java.util.*;

class Solution {
    int N, M, C;
    int[] visited;
    int[] parant_edge, weak_edge, strong_edge;
    LinkedList<Integer>[] adj, child;
    public int solution(int n, int[][] edges) {
        int answer = 0;
        visited = new int[n+1];
        parant_edge = new int[n+1];
        weak_edge = new int[n+1];
        strong_edge = new int[n+1];
        adj = new LinkedList[n+1];
        child = new LinkedList[n+1];
        for(int i = 1; i <= n; i++) {
            adj[i] = new LinkedList<>();
            child[i] = new LinkedList<>();
        }
        N = n;
        M = edges.length;
        for(int i = 0; i < M; i++) {
            adj[edges[i][0]].add(edges[i][1]);
            adj[edges[i][1]].add(edges[i][0]);
        }
        visited[1] = 1;
        dfs(1, 0);
        for(int i = 1; i <= n; i++) {
            boolean flag = false;
            for(int j = 0; j < child[i].size(); j++) {
                int v = child[i].get(j);
                if(strong_edge[v]>0 || weak_edge[v]-
parant_edge[v]>1) {
                    flag = true;
                    break;
                }
            }
            if(flag || M - (N-1) - weak_edge[i] != 0) continue;
            answer +=i;
        }
        return answer;
    }
    void dfs(int cur, int from) {
        for(int i = 0; i < adj[cur].size(); i++) {
            int node = adj[cur].get(i);
            if(from != node) {
                if(visited[node] == 0) {
                    visited[node] = visited[cur] + 1;
                    child[cur].add(node);
                    int temp = strong_edge[node];
                    dfs(node, cur);
                    parant_edge[node] = strong_edge[cur] - temp;
                    strong_edge[cur] += strong_edge[node];
                    weak_edge[cur] += weak_edge[node];
                }
                else if(visited[cur] > visited[node]) {
                    weak_edge[cur]++;
                    strong_edge[node]++;
                }
            }
        }
    }
}
```

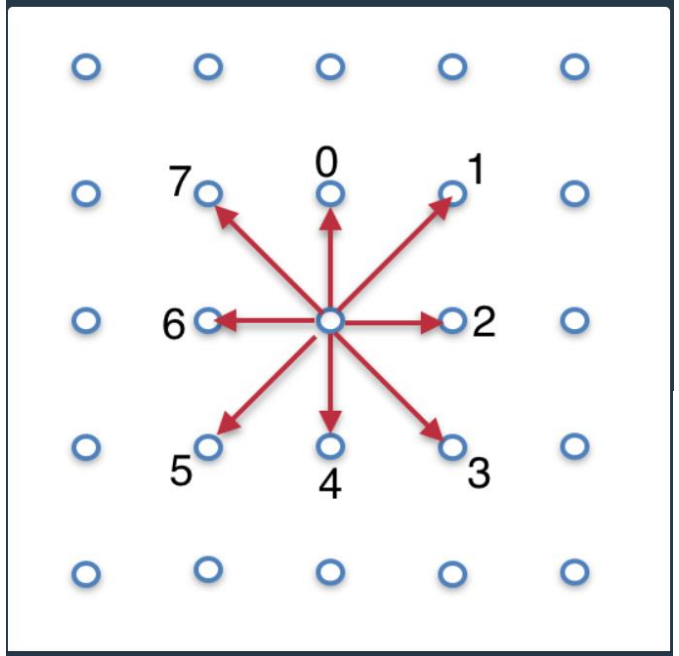


[방의 개수](https://programmers.co.kr/learn/courses/30/lessons/49190?language=python3)

https://programmers.co.kr/learn/courses/30/lessons/49190?language=python3

문제 분류 : 그래프

원점(0,0)에서 시작해서 아래처럼 숫자가 적힌 방향으로 이동하며 선을 긋습니다.



ex) 1일때는 오른쪽 위 로 이동

그림을 그릴 때, 사방이 막히면 방하나로 셉니다.
이동하는 방향이 담긴 배열 arrows가 매개변수로 주어질 때, 방의 갯수를 return 하도록 solution 함수를 작성하세요.

제한사항

- 배열 arrows의 크기는 1 이상 100,000 이하 입니다.
- arrows의 원소는 0 이상 7 이하 입니다.
- 방은 다른 방으로 둘러 싸여질 수 있습니다.

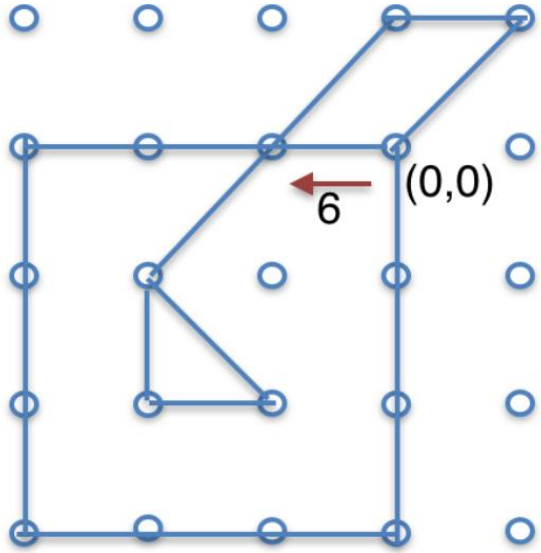
입출력 예

arrows	return
[6, 6, 6, 4, 4, 4, 2, 2, 2, 0, 0, 0, 1, 6, 5, 5, 3, 6, 0]	3

그래프의 이론적인 부분(오일러의 다면체 정리)에 대해 알고 있다면 쉬운문제

다멘체 정리를 안다면 화살표 움직임에 따라 edges만 조절해주면 됨.

아니라면 어려운문제



- (0,0) 부터 시작해서 6(왼쪽) 으로 3번 이동합니다. 그 이후 주어진 arrows 를 따라 그립니다.
- 삼각형 (1), 큰 사각형(1), 평행사변형(1) = 3

```
def solution(arrows): #오일러의 다면체 정리(2차원): f = 1+e-v
    nodes = set()
    edges = set()
    (x,y)=(0,0)
    go=[(0,1),(1,1),(1,0),(1,-1),(0,-1),(-1,-1),(-1,0),(-1,1)]
    nodes.add((x,y))

    for a in arrows:
        for i in range(2):
            (x2,y2)=(x+go[a][0],y+go[a][1])
            nodes.add((x2,y2))
            if (x,y)>(x2,y2):
                edges.add(((x,y),(x2,y2)))
            else:
                edges.add(((x2,y2),(x,y)))
            (x,y) = (x2,y2)

    return 1+len(edges)-len(nodes)
```

```
def solution(arrows):
    point=set([(0,0)])
    line=set()
    move=[[0,2],[2,2],[2,0],[2,-2],[0,-2],[-2,-2],[-2,0],[-2,2]]
    pre_point=(0,0)
    for A in arrows:
        next_point=(pre_point[0]+move[A][0],   pre_point[1]+move[A][1] )
        mid_point=(pre_point[0]+move[A][0]//2,   pre_point[1]+move[A][1]//2 )
        point.add(next_point)
        point.add(mid_point)
        line.add((pre_point,mid_point))
        line.add((mid_point,pre_point))
        line.add((mid_point,next_point))
        line.add((next_point,mid_point))
        pre_point=next_point
    answer = len(line)//2-len(point)+1
    return answer if answer>=0 else 0
```