DFS/BFS 리뷰

노션 주소 : DFS/BFS 리뷰

1. 주식가격

문제 설명

초 단위로 기록된 주식가격이 담긴 배열 prices가 매개변수로 주어질 때, 가격이 떨어지지 않은 기간은 몇 초인지를 return 하도록 solution 함수를 완성하세요.

제한사항

- prices의 각 가격은 1 이상 10,000 이하인 자연수입니다.
- prices의 길이는 2 이상 100,000 이하입니다.

입출력 예

prices	return
[1, 2, 3, 2, 3]	[4, 3, 1, 1, 0]

큐를 이용한 풀이

```
from collections import deque

def solution(prices):
    queue = deque(prices)
    answer = []
    while queue:
        price = queue.popleft()
        sec = 0
        for q in queue:
            sec +=1
            if price > q:
                break
```

```
            answer.append(sec)
        return answer
```

스택을 이용한 풀이

```
def solution(prices):
    length = len(prices)
    answer = [ i for i in range (length - 1, -1, -1)]

    stack = [0]
    for i in range (1, length, 1):
        while stack and prices[stack[-1]] > prices[i]:
            j = stack.pop()
            answer[j] = i - j
        stack.append(i)
    return answer
```

2. 타겟 넘버 ⭐

https://school.programmers.co.kr/learn/courses/30/lessons/43165

타겟 넘버 제출 내역

문제 설명

n개의 음이 아닌 정수들이 있습니다. 이 정수들을 순서를 바꾸지 않고 적절히 더하거나 빼서 타겟 넘버를 만들려고 합니다. 예를 들어 [1, 1, 1, 1, 1]로 숫자 3을 만들려면 다음 다섯 방법을 쓸 수 있습니다.

```
-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
```

사용할 수 있는 숫자가 담긴 배열 numbers, 타겟 넘버 target이 매개변수로 주어질 때 숫자를 적절히 더하고 빼서 타겟 넘버를 만드는 방법의 수를 return 하도록 solution 함수를 작성해주세요.

제한사항

- 주어지는 숫자의 개수는 2개 이상 20개 이하입니다.
- 각 숫자는 1 이상 50 이하인 자연수입니다.
- 타겟 넘버는 1 이상 1000 이하인 자연수입니다.

입출력 예

numbers	target	return
[1, 1, 1, 1, 1]	3	5
[4, 1, 2, 1]	4	2

문제풀이 아이디어

각 숫자에서 +,- 부호로 2가지 경우의 수가 생겨난다는 것

numbers의 각 숫자에 따라 2가지 경우의 자손을 뻗을 수 있게 되고, 모든 경우의 수를 탐색하는 bfs/dfs 문제가 됨을 깨달아야 함

<트리 그림 첨부>

이러한 트리를 너비 우선으로 탐색한다면 ⇒ BFS

깊이 우선으로 탐색한다면 ⇒ DFS

BFS 풀이

```
def solution(numbers, target):
    leaves = [0] # 모든 계산 결과를 담는 곳
    answer = 0

    for num in numbers:
        temp = []

        for leaf in leaves: # 너비 우선 탐색
            temp.append(leaf + num)
            temp.append(leaf - num)

        leaves = temp # 부모 노드 업데이트

    for leaf in leaves: # 모든 계산 결과
        if leaf == target: # target이랑 같을 때
            answer += 1
    return answer
```

ex) numbers = [4,1,2,1], target = 3 인 경우

1. num = 4 일 때

   leaves = [0] ← 첫 노드이기 때문

   temp = [0+4, 0-4] = [4, -4] ← **너비 우선 탐색**

2. num = 1 일 때

   leaves = [4, -4]

   temp = [4+1, 4-1, -4+1, -4-1] = [5, 3, -3, -5]

3. num = 2 일 때

   leaves = [5,3,-3,-5]

   temp = [5+2,5-2,3+2,3-2,-3+2,-5+2,-5-2] = [7,3,5,1,-1,-3,-7]

4. num = 1

   위의 과정 반복

DFS 풀이

DFS는 주로 재귀함수로 구현한다.

```
def dfs(numbers, target, idx, values):     # idx : 깊이, values : 더하고 뺄 특정 leaf

    global cnt
    cnt = 0

    # 깊이가 끝까지 닿았으면
    if idx == len(numbers) & values == target :
        cnt += 1
        return

    # 끝까지 탐색했는데 sum이 target과 다르다면 그냥 넘어간다
    elif idx == len(numbers) :
        return

    # 재귀함수로 구현
    dfs(numbers, target, idx+1, values + numbers[idx])   # 새로운 value 값 세팅
    dfs(numbers, target, idx+1, values - numbers[idx])

 def solution(numbers, target) :

    global cnt
    dfs(numbers, target, 0, 0)

    return cnt
```

3. 백준 1260

https://www.acmicpc.net/problem/1260

문제

그래프를 DFS로 탐색한 결과와 BFS로 탐색한 결과를 출력하는 프로그램을 작성하시오. 단, 방문할 수 있는 정점이 여러 개인 경우에는 정점 번호가 작은 것을 먼저 방문하고, 더 이상 방문할 수 있는 점이 없는 경우 종료한다. 정점 번호는 1번부터 N번까지이다.

입력

첫째 줄에 정점의 개수 N(1 ≤ N ≤ 1,000), 간선의 개수 M(1 ≤ M ≤ 10,000), 탐색을 시작할 정점의 번호 V가 주어진다. 다음 M개의 줄에는 간선이 연결하는 두 정점의 번호가 주어진다. 어떤 두 정점 사이에 여러 개의 간선이 있을 수 있다. 입력으로 주어지는 간선은 양방향이다.

출력

첫째 줄에 DFS를 수행한 결과를, 그 다음 줄에는 BFS를 수행한 결과를 출력한다. V부터 방문된 점을 순서대로 출력하면 된다.

예제 입력 1 복사

```
4 5 1
1 2
1 3
1 4
2 4
3 4
```

예제 출력 1 복사

```
1 2 4 3
1 2 3 4
```

기본적인 DFS, BFS 문제 → 저번시간 배운 내용 토대로 적용해보기!

4. 백준 1743 음식물 피하기 ⭐

https://www.acmicpc.net/problem/1743

문제

코레스코 콘도미니엄 8층은 학생들이 3끼의 식사를 해결하는 공간이다. 그러나 몇몇 비양심적인 학생들의 만행으로 음식물이 통로 중간 중간에 떨어져 있다. 이러한 음식물들은 근처에 있는 것끼리 뭉치게 돼서 큰 음식물 쓰레기가 된다.

이 문제를 출제한 선생님은 개인적으로 이러한 음식물을 실내화에 묻히는 것을 정말 진정으로 싫어한다. 참고로 우리가 구해야 할 답은 이 문제를 낸 조교를 맞추는 것이 아니다.

통로에 떨어진 음식물을 피해가기란 쉬운 일이 아니다. 따라서 선생님은 떨어진 음식물 중에 제일 큰 음식물만은 피해가려고 한다.

선생님을 도와 제일 큰 음식물의 크기를 구해서 “10ra"를 외치지 않게 도와주자.

입력

첫째 줄에 통로의 세로 길이 N(1 ≤ N ≤ 100)과 가로 길이 M(1 ≤ M ≤ 100) 그리고 음식물 쓰레기의 개수 K(1 ≤ K ≤ N×M)이 주어진다. 그리고 다음 K개의 줄에 음식물이 떨어진 좌표 (r, c)가 주어진다.

좌표 (r, c)의 r은 위에서부터, c는 왼쪽에서부터가 기준이다. 입력으로 주어지는 좌표는 중복되지 않는다.

출력

첫째 줄에 음식물 중 가장 큰 음식물의 크기를 출력하라.

예제 입력 1 복사

```
3 4 5
3 2
2 2
3 1
2 3
1 1
```

예제 출력 1 복사

```
4
```

힌트

```
# . . .
. # # .
# # . .
```

위와 같이 음식물이 떨어져있고 제일큰 음식물의 크기는 4가 된다. (인접한 것은 붙어서 크게 된다고 나와 있음. 대각선으로는 음식물 끼리 붙을수 없고 상하좌우로만 붙을수 있다.)

풀이 아이디어

주어진 힌트에 따라 입력받는 좌표를 그래프 형태로 나타내고, 음식물이 있는 곳 = 1로 표현

음식물이 있는 곳을 차례로 방문하고, bfs 알고리즘을 이용하여 가장 큰 음식물의 크기 출력

BFS 풀이

```
# 통로 세로 길이 = n, 가로 길이 = m, 음식물 쓰레기 개수 = k
# k개 줄에 음식물이 떨어진 좌표 주어짐
from collections import deque

n,m,k = map(int, input().split())
graph = [[0 for _ in range(m)] for _ in range(n)]

for _ in range(k): # n*m 위에 음식물 있는 곳 표현하기
  r,c = map(int,input().split())
```

```
  graph[r-1][c-1]=1

d = [(0,1), (0,-1), (1,0), (-1,0)]
```

```
# bfs 재귀함수
def bfs(y,x):
  q = deque() # bfs 탐색을 위한 큐 생성
  q.append((y,x))
  graph[y][x] = 0 # 방문 처리 - 방문한 위치는 0으로 처리하여 중복 방문 방지
  count = 0
  while q:
    y,x = q.popleft() # 현재 방문 노드 꺼내기
    count += 1
    for dy,dx in d: # 상하좌우 탐색
      Y, X = y+dy, x+dx
      if (0 <= Y < n) and (0 <= X < m) and graph[Y][X] == 1: # 방문한 노드 중 1일때
        graph[Y][X] = 0
        q.append((Y,X))
  return count
```

```
# 음식물이 있는 곳 탐색
result = 1
for y in range(n):
  for x in range(m):
    if graph[y][x] == 1:
        result = max(result, bfs(y,x))

print(result)
```