*#1260 DFS와 BFS*

|  |
| --- |
| n,m,start = map(int, input().split())  **import** sys file = [[]**for** \_ **in** range(n)] **for** \_ **in** range(m):  a,b = map(int, sys.stdin.readline().split())  file[a-1].append(b-1)  file[b-1].append(a-1) **for** i **in** range(n):  **if** file[i]:  file[i].sort()  *#DFS 구현(깊이 우선 탐색)* **def** dfs(v, ans):  ans += [v+1]  **for** i **in** file[v]:  **if** i+1 **not in** ans:  ans = dfs(i, ans)  **return** ans print(\*dfs(start-1, []))  *#BFS 구현(넓이 우선 탐색)* **from** collections **import** deque left = deque() ans = str(start-1) left.append(str(start-1)) res = [] res.append(start-1) **def** bfs(left):  **while** left:  a = left.popleft()  **for** k **in** file[int(a)]:  **if** k **not in** res:  res.append(k)  left.append(k)  **for** i **in** res:  print(int(i)+1, end = **' '**)  **return** bfs(left)  *#원래는 재귀 탈출 조건으로 count를 해서 더이상 해당 노드의 자식 노드가 없을때, 즉 count == 0일때에 출력을 하도록 했었다 #그러나 그렇게 하니까 당연히 출력초과, 혹은 부족한 길이로 출력이 되었다 #그래서 count와 check필요 없이, 특히 하나의 종류만 출려하면 되므로 check리스트는 무의미하다. #그냥 현재 출력 예정인 리스트에만 집중해 해당 리스트에 이미 포함 되어있는지 아닌지의 여부만 확인한 뒤에 #dfs는 재귀의 특징을 이용해 재귀함수를 한번 부를때 마다 리스트에 current node의 번호를 더하고 #bfs는 queue에 아무 값도 없을 떄 출력하면 된다.* |

*#6603 로또*

**def** search(now, v):  
 **if** now == 6:  
 **for** i **in** range(6):  
 print(ans[i], end = **' '**)  
 print()  
 **return  
 for** i **in** range(v, len(num)):  
 **if** check[i] == 0:  
 ans.append(num[i])  
 check[i] = 1  
 search(now+1, i)  
 ans.pop()  
 check[i] = 0  
**while True**:  
 file = list(map(int, input().split()))  
 **if** file == [0]:  
 **break  
 else**:  
 k,num = file[0], file[1:]  
 num.sort()  
 ans, check = [], [0]\*len(num)  
 search(0,0)  
 print()

*#2109 빵집*

|  |  |
| --- | --- |
| r,c = map(int, input().split()) **import** sys dx,dy = [-1,0,1], [1,1,1] board = [] **for** \_ **in** range(r):  file = list(map(str, sys.stdin.readline().strip()))  **for** i **in** range(c):  **if** file[i] == **'.'**:  file[i] = 0  **else**:  file[i] = 1  board.append(file)  **def** dfs(i,j):  check[i][j] = 1  **if** j == c-1:  **return True  for** k **in** range(3):  x,y = i+dx[k], j+dy[k]  **if** 0<=x<r **and** 0<=y<c:  **if** check[x][y] == 0 **and** board[x][y] == 0:  now = dfs(x,y)  **if** now:  **return** now  **return False** res = 0 check = [[0]\*c **for** \_ **in** range(r)] **for** i **in** range(r):  **if** dfs(i,0):  res += 1 print(res)   |  | | --- | | 이 문제의 경우에는 greedy algorithm과 dfs를 합쳐서 풀어야 하는 문제였다. 대각선 위, 직선, 대각선 아래 세 방향으로 이동이 가능한데, 위의 행부터 더 윗방향으로, 즉 더 위로 도착할수록 많은 경우의 파이프라인을 연결할 수 있을 것이다. 그렇게 하기 위해 for문으로 r개만큼 dfs를 진행하고, dfs로는 재귀적으로 boolean값을, 즉 해당 순서가 가능한지 아닌지 확인해주도록 하고 만약 가능하면 답 += 1을 해준다.  내가 취약한 재귀를 재귀 함수안에서 교묘하게 실행하는 부분이었기 때문에 여러가지로 의미가 있으니 나중에 한번 더 풀면 좋을 것 같다. | |

*#7576 토마토*

|  |
| --- |
| **import** sys **from** collections **import** deque m, n = map(int, sys.stdin.readline().split()) file, dQ = [], deque() **for** i **in** range(n):  file.append(list(map(int, sys.stdin.readline().split())))  **for** j **in** range(m):  **if** file[i][j] == 1:  dQ.append((i,j)) dx, dy = [-1,1,0,0], [0,0,-1,1] ans = -1 **while** dQ:  now = dQ.popleft()  **for** i **in** range(4):  a,b = now[0] + dx[i], now[1] + dy[i]  **if** 0<=a<n **and** 0<=b<m **and** file[a][b] == 0:  file[a][b] = file[now[0]][now[1]] + 1  dQ.append((a,b)) **for** i **in** file:  ans = max(max(i)-1, ans)  **if** 0 **in** i:  ans = -1  **break** print(ans) |

*#10026 적록 색약*

|  |
| --- |
| n = int(input()) **import** sys board = [list(map(str, sys.stdin.readline().strip())) **for** x **in** range(n)] sys.setrecursionlimit(1000000) ans = [0]\*2 dx, dy = [-1,1,0,0],[0,0,-1,1] **def** DFS(i,j):  check[i][j] = 1  c = board[i][j]  now = c  **if** board[i][j] == **'R'**:  board[i][j] = **'G'  for** k **in** range(4):  x,y = i+dx[k], j+dy[k]  **if** 0<= x < n **and** 0<= y< n **and** check[x][y] == 0:  **if** now == board[x][y]:  DFS(x,y)   **for** k **in** range(2):  check = [[0] \* n **for** x **in** range(n)]  **for** i **in** range(n):  **for** j **in** range(n):  **if** check[i][j] == 0:  DFS(i,j)  ans[k]+= 1 print(ans[0], ans[1]) |

*#11724 연결요소의 개수*

|  |
| --- |
| **import** sys sys.setrecursionlimit(10000) n,m = map(int, input().split()) file = [] board = [[0]\*n **for** x **in** range(n)] visit = [0]\*n **for** i **in** range(m):  A = (list(map(int, sys.stdin.readline().split())))  a,b = A[0],A[1]  board[a-1][b-1] = 1  board[b-1][a-1] = 1  file.append(A) **def** DFS(a):  visit[a] = 1  **for** i **in** range(n):  **if** visit[i] == 0 **and** board[a][i] == 1:  DFS(i) count = 0 **for** i **in** range(n):  **if** visit[i] == 0:  DFS(i)  count += 1 print(count) |

*#10472 십자 뒤집기*

|  |
| --- |
| #pypy3으로 했을 때는 통과가 되었으나 python3로 해결 했을 때는 틀렸다고 나옴. Set의 영향인 것으로 보임  t = int(input()) **import** sys **import** copy dx,dy = [-1,1,0,0],[0,0,-1,1] **def** bfs(left):  ans = 0  **while** left:  v,b = list(left)[0][0], list(left)[0][1]  left.remove((v,b))  add = 0  **for** i **in** b:  add += int(i)  **if** add == 0:  ans = v  **break  for** l **in** range(9):  i,j =l//3,l%3  now = []  **for** item **in** b:  now.append(int(item))  now[l] = abs(now[l]-1)  **for** k **in** range(4):  x,y = dx[k]+i, dy[k]+j  **if** 0 <=x<3 **and** 0<=y<3:  now[x\*3+y] = abs(now[x\*3+y]-1)  left.add((v+1, **''**.join(str(i) **for** i **in** now)))  **return** ans **for** \_ **in** range(t):  board = []  **for** i **in** range(3):  file = list(map(str, sys.stdin.readline().strip()))  **for** j **in** range(3):  **if** file[j] == **'\*'**:  file[j] = 1  **else**:  file[j] = 0  board.append(file)  cur = **''**.join(str(i) **for** i **in** sum(board, []))  left = set()  left.add((0,cur))  print(bfs(left)) |

*#1012 유기농 배추*

|  |
| --- |
| **import** sys  sys.setrecursionlimit(10\*\*8) **def** DFS(x,y):  **for** i **in** range(4):  a,b = x+dx[i], y+dy[i]  **if** 0<=a<n **and** 0<=b<m **and** file[a][b] == 1:  file[a][b] = 0  DFS(a,b)  dx,dy = [-1,1,0,0],[0,0,-1,1] t = int(input()) res = [] **for** x **in** range(t):  count = 0  m,n,k = map(int, input().split())  file = [[0]\*m **for** x **in** range(n)]  **for** x **in** range(k):  a,b = map(int, input().split())  file[b][a] = 1  **for** i **in** range(n):  **for** j **in** range(m):  **if** file[i][j] == 1:  count += 1  file[i][j] = 0  DFS(i, j)  res.append(count) **for** i **in** res:  print(i) |

*#1194 달이 차오른다, 가자*

*#2379 트리 탐색하기*