59

kitt ★ 1170 Last Edit: October 11, 2018 8:41 AM 11.9K VIEWS

Use `hit` to record how many times a `0` grid has been reached and use `distSum` to record the sum of distance from all `1` grids to this `0` grid. A powerful pruning is that during the BFS we use `count1` to count how many `1` grids we reached. If `count1 < buildings` then we know not all `1` grids are connected are we can return `-1` immediately, which greatly improved speed (beat 100% submissions).

```
def shortestDistance(self, grid):
    if not grid or not grid[0]: return -1
    M, N, buildings = len(grid), len(grid[0]), sum(val for line in grid for val in line if val == 1)
    hit, distSum = [[0] * N for i in range(M)], [[0] * N for i in range(M)]

    def BFS(start_x, start_y):
        visited = [[False] * N for k in range(M)]
        visited[start_x][start_y], count1, queue = True, 1, collections.deque([(start_x, start_y, 0)])
        while queue:
            x, y, dist = queue.popleft()
            for i, j in ((x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)):
                if 0 <= i < M and 0 <= j < N and not visited[i][j]:
                    visited[i][j] = True
                    if not grid[i][j]:
                        queue.append((i, j, dist + 1))
                        hit[i][j] += 1
                        distSum[i][j] += dist + 1
                    elif grid[i][j] == 1:
                        count1 += 1
        return count1 == buildings

    for x in range(M):
        for y in range(N):
            if grid[x][y] == 1:
                if not BFS(x, y): return -1
    return min([distSum[i][j] for i in range(M) for j in range(N) if not grid[i][j] and hit[i][j] == buildings] or [-1])
```