reflection lab 4

changed maze.py to track:

- path length — # of star cells
- cells explored — push/pop count
- runtime for 3 maze sizes — 10×10, 20×20, 30×30 with 3 blockage densities — 10%, 30%, 50%
- tested search orders (NSWE, NESW, RANDOM) with fixes seeds

key findings

path length comparison

Maze Size	Blockage %	DFS path length	BFS path length
10×10	10%	28	18
10×10	30%	42	24
20×20	10%	89	38

BFS consistently finds shorter paths which is optimal for unweighted grids

DFS paths are on average longer because of deep exploration before backtracking

search direction impact

NSWE vs NESW (10×10, 30% blocks):

alg	NSWE path length	NESW path length
DFS	42	37
BFS	24	24

RANDOM — 3 runs, same maze:

- DFS path lengths: 39, 42, 35

- BFS stayed consistent — 24

analysis:

- DFS is highly sensitive to search order (up to 20% difference)
- BFS is order-invariant for path length but not exploration count

bloackage density effects

low blockage - 10%:

- DFS explores 22% fewer cells than BFS on avg
- both algs find path quickly

high blockage - 50%:

- DFS explores 15-30% more cells than BFS
- BFS occasionally gets trapped inn dead ends

maze comparison: DFS vs BFS in 30% blocked maze. DFS explores more dead ends

Big-O analysis:

data structure	operation	complexity	justification
stack - py list	push/pop	O(1)*	list append/pop from end
queue - py list	pop	O(n)	pop(0) shifts all ele
queue- LinkedList	push/pop	O(1)	appendRight/popLeft are constant

conclusion:

- BFS when optimal path length needed
- DFS when memory efficiency needed (lower average queue size)
- search order significantly impacts DFS performance

    - RANDOM helps avoid worst-case scenario
- LinkedList-based queue does better than list-based for large mazes because of O(1) operations