App Academy

Matrix BFS

Matrix BFS

- Matrix BFS is similar to binary tree BFS with a few minor changes
  - When pushing neighbors into the queue, there will usually be up to 4 neighbors (assuming 4-directional) instead of just two.
  - With graphs, we must ensure that we do not re-visit visited nodes.
    - One way we can solve this is by tracking visited nodes in a “visited set”.
  - We must ensure that we do not go out of bounds.
- Demo:
  - Shortest Path in Binary Matrix

start			X	
	X			
	X		X	X
	X			
	X			finish

start	1	2	X	5
1	X	3	3	4
2	X	4	X	X
3	X	5	5	6
4	X	6	6	finish 6

Matrix BFS

```
const shortestPathBinaryMatrix = function(grid) {
    if(grid[0][0] === 1) return -1;
    const queue = [];
    queue.push([0,0,1])
    const directions = [[1,0], [0,1], [-1,0], [0,-1], [1,1], [-1,-1], [-1,1], [1,-1]];
    const visited = new Set();
    visited.add(`${0}-${0}`)

    while (queue.length > 0) {
        let [row, col, level] = queue.shift();
        if (row === grid.length-1 && col === grid[0].length-1) return level;

        for (let dir of directions) {
            const newRow = row+dir[0];
            const newCol = col+dir[0];
            if (inBound(grid, newRow, newCol) && grid[newRow][newCol] !== 1 && !visited.has(`${newRow}-${newCol}`)) {
                queue.push([newRow, newCol, level+1]);
                visited.add(`${newRow}-${newCol}`)
            }
        }
    }
    return -1;
};
```

```
const inBound = function(grid, row, col) {
    const rowInBound = row >= 0 && row < grid.length;
    const colInBound = col >= 0 && col < grid[0].length;
    return rowInBound && colInBound;
}
```

Questions?

Let’s practice!

- Review
  - Rotting Oranges
  - 01 Matrix
- Bonus
  - Shortest Bridge