# RADAR

FRAMED BY: SOUMYADEEP PAUL

GITHUB: https://github.com/Soumyadeepaul/Radar

## INTRODUCTION

Radar, a radio detection and ranging system that uses radio waves to trace the presence on objects in a particular range. By analyzing the delay of the wave to and fro from the object the distance, speed, and direction could be determined. This Radar system is widely used in aviation, military defense, marine navigation, and automotive systems.

## STRUCTURAL OVERVIEW

The Radar Grid is converted to a Matrix of $[n, m]$ where the number of rows is n and the number of columns is m.

1. Positions marked as 'X' are the obstacles through which radar can't look.
2. Positions marked as 'E' indicated enemy.
3. The radar system(Station) is always located at mid of the grid marked as 0.
4. Positions marked as 0 are where radar can look through.

Example 1:

```
[
 [0,E,0,X,0,0,0,0],
 [0,X,X,0,0,0,0,0],
 [0,0,0,0,0,0,X,0],
 [0,0,0,0,0,X,E,0]
]
```

Example 2:

```
[
 [0,X,X,X,E],
 [0,X,0,X,0],
 [0,X,X,X,0]
]
```

Example 3:

```
[
 [0,0,0,X,0],
 [0,0,0,X,0],
 [0,0,0,X,E],
]
```

## PROBLEM STATEMENT

Problem 1:

Check the presence of the enemy on the grid spotted by radar. [Medium]

Input: Matrix Output: Boolean

Problem 2:

The total number of enemies present on the grid spotted by radar. [Medium]

Input: Matrix Output: Integer

Problem 3:

Locate the closest enemy and its distance from the radar. [Hard]

Input: Matrix Output: List[List] : Position(x,y) and Distance

Problem 4:

The Direction to reach the closest enemy. [Hard]

Input: Matrix Output: List[List]

Problem 5:

Will Station be taken down? [Hard]

Input: Matrix Output: Boolean

## EXPLANATION

Problem 1's Explanation:

A search operation has to be performed from the station to find the presence of the enemy. If a wall of obstruction is present between the station and the enemy, as shown in Example 3, the enemy will not be spotted. Similarly of the station is surrounded by obstructions, and the enemy hides beyond that; the enemy will not be spotted as shown in Example 2. In such scenario enemy also can't invade

Problem 2's Explanation:

A continuation of problem 1, to find out all possible enemies traceable from the station.

Problem 3's Explanation:

The enemy, closest to the station, must be located and return it's position with the minimum distance of the enemy to the station, keeping all the necessary conditions intact mentioned in problem 1.

Problem 4's Explanation:

A continuation of problem 3, get the positions to reach the closest enemy.

Problem 5's Explanation:

1. Destroy the closest enemy first, and so on.
2. While destroying the enemy one after another, the other enemies will take one step closer to the station.
3. If in the meantime any of the enemy invades the station, it gets destroyed.
4. Each missile takes 1 unit time to take down one enemy at a time.

5. Enemy can be merged into a single enemy if present at the same index; but k separate missiles to take them down, i.e. k unit time.

## Solution Approach

### Problem 1's Approach

A Graph traversal technique Breadth First Search is to be used until we locate any enemy return True, if no enemy found return False.

### Problem 2's Approach

A Graph traversal technique Breadth First Search is to be used over the grid and keep count of enemies spotted.

### Problem 3's Approach

A Graph traversal technique Breadth First Search is to be used until we locate any enemy and keep count of the shortest distance to travel the enemy location, and return the index position of the enemy and the optimal distance.

### Problem 4's Approach

It is the continuation of problem 3 where we have to backtrack from the enemy position to the mid position (Station) and store the direction in reverse order.

### Problem 5's Approach

Following the rules leads to a complex algorithm of:

Algorithmic Approach 1:

1. Breadth First Search for all the enemy locations and the distance. Time complexity of $O(n * m)$.
2. Backtrack to the station from all the enemies(k) to store the direction. Time Complexity of $O(k * n * m)$.
3. The minimum distance to be targeted and killed first, Time Complexity of $O(k)$. (Searching for minimum).

Time required: $(k-1) + (k-2) + \cdots + 1 = \frac{(k-1)*k}{2}$

$$\rightarrow \frac{1}{2}(k^2 - k)$$

Time Complexity: $O(k^2)$

Because for every iteration minimum to be searched.

4. At the same time all enemies will move one step forward. In the worst case, to kill all the k enemies, time needed will be for updating the enemies' position.
    a. For k enemies, the next k-1 enemies will come forward.
    b. For k-1 enemies, the next k-2 enemies will come forward.
    c. For 2 enemies, the next 1 enemy will come forward.

    Time required: $(k-1)+(k-2)+\cdots+1=\frac{(k-1)*k}{2}$

    $\rightarrow \frac{1}{2}(k^2-k)$

    Time Complexity: $O(k^2)$

If at any unit of time, if enemy's distance gets 0. The station gets defeated.

$\therefore Over\ all\ Time\ Complexity$

$O((n*m)+(k*n*m)+k^2+k^2)$

Algorithmic Approach 2:

1. Breadth First Search for all the enemy locations and the distance. Time complexity of $O(n*m)$.
2. Sort the distances and target/kill the minimum distance. Time complexity of $O(k*logk)$.
3. At the same time all enemies will move one step forward, therefore k-1 elements will come forward means distance array will get k-1 element now with every distance get reduce by 1 unit.

$$Distances: [2,3,3,3,4,5]$$

$$Iter\ 1: [2,2,2,3,4]$$

$$Iter\ 2: [1,1,2,3]$$

$$Iter\ 3: [0,1,2]\ (Station\ gets\ defeated)$$

$$Time\ required: (k-1)+(k-2)+\cdots+1=\frac{(k-1)*k}{2}$$

$$\rightarrow \frac{1}{2}(k^2-k)$$

$$Time\ Complexity: O(k^2)$$

If at any unit of time, if enemy's distance gets 0. The station gets defeated.

$\therefore Over\ all\ Time\ Complexity$

$O((n*m)+(k*logk)+k^2)$

Algorithmic Approach 3:

1. Breadth First Search for all the enemy locations and the distance. Time complexity of $O(n*m)$.
2. Sort the distances and target/kill the minimum distance. Time complexity of $O(k*logk)$.

3. If distance array elements if greater than or equal to the index at which it is present result defeat. (Intuition: to convert any element to be 0 from distance d, d times 1 should be deducted, for converting to 0 distance, d or more than d number of elements must be present).

$$Distances: [2,3,3,3,4,5]$$

Here 3rd 3 is present at index 3. Therefore, station gets defeated.

Time Complexity: $O(k)$

$\therefore Over\ all\ Time\ Complexity$

$$O((n * m) + (k * logk) + k)$$

## COMPLEXITY ANALYSIS

| | |
|---|---|
| Problem 1: | $O(n * m)$ |
| Problem 2: | $O(n * m)$ |
| Problem 3: | $O(n * m)$ |
| Problem 4: | $O((n * m) + (n * m))$ |

Problem 5:

| Approach 1: | Approach 2: | Approach 3: |
|---|---|---|
| $O((n * m) + (k * n * m) + k^2 + k^2)$ | $O((n * m) + (k * logk) + k^2)$ | $O((n * m) + (k * logk) + k)$ |

## CONCLUSION

The time complexity mostly compromised by how the grid is plotted, using a dense adjacency matrix. If converted into less dense adjacency list would help reduce complexity as the obstacles will not be considered as a vertex and no edge for such.

## FUTURE SCOPE

Breadth First Search is one of the significant approaches to search in all directions simultaneously. The algorithmic approach might change with the change of the adjacency matrix to an adjacency list. The time complexity of $O(n * m)$ will reduce to $O(v + e)$. Due to which problem 1 to 4 would reduce to $O(v + e)$ and problem 5 approach 1 to $O(k * e)$, approach 2 to $O(k^2)$ and approach 3 to $O(k * logk)$. [Needed to be verified, assumed time complexity mentioned].