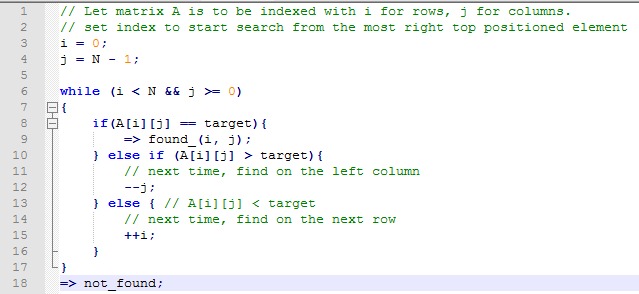
1. **(*Algorithm design and complexity analysis*)**

**For a square N x N matrix A, assume the elements are sorted in ascending order along the horizontal and vertical directions already, i.e., A[i][k] ≤ A[j][k] and A[k][i] ≤ A[k][j], where i<j. Develop an efficient algorithm to search for the query value v from A, return the location if found, None otherwise. Analyze the time complexity of your algorithm.**

**ANSWER:**

Time complexity- best and worst case

**Best**- if the target is the same spot supposedly O(1) constant

**Worst** – if the target doesn’t exist and it ends up to the bottom left square which means that there are several routes that go to the bottom square with the length of the path being 2n, which is O(n).

CODE- i = rows

J= columns

Starts from the top right element

N = size of length of columns and rows

When target value is less than the current elements value, the path to the target is shifted one element to the left and if the target values is greater than the current elements value, the path is shifted down 1 element. Rows and columns are ascending in element value.

This process will continue until it reaches its target value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | **7** | **12** | **15** | **20** |
| **3** | **9** | **14** | **17** | **21** |
| **8** | **11** | **16** | **18** | **26** |
| **22** | **23** | **25** | **27** | **30** |
| **28** | **29** | **30** | **32** | **35** |