Team Cheesy Potatoes -- Alexia Leong, Wenting Li, Bing Li

APCS2 pd01

HW02 -- Speaking in Pseudocode

2018-01-31

Step By Step Process (with a n*n array that is increasing L → R and Up → Down):

1. Start with element in top right corner [0, number of columns - 1] because there are distinctions that can be made if the target is < or > than the element in the array. If you start at [0,0], both elements to the right and below it are greater so you don't know if you have to increase the row or column number.
2. Check if the element you are currently at (let's make it x) is within the bounds of the matrix, put in a while loop (because we can break at any time) with a conditional that row should be less than length - 1 and column should be >= 0
3. If x == target :
    a. Case if true → return true
    b. Case if false →
        i. If x < target, then check the element one place below x, row++ because the elements down the column increases, so the new x will be closer to the target number.
        Note: If x is the last element in the column (at column index n - 1), print out a message saying "the number you are looking for is not in the matrix."
        ii. If x > target, then check the element one place left from x, column --, because the elements left of x decreases, so the new x will be closer to the target number.
        Note: If x is the first element in the row (at row index 0), print out a message saying "the number you are looking for is not in the matrix."

Runtime Categorization:

Given a n*n array that is increasing L → R and Up → Down, running our search algorithm would result in the best case of O(1) (when the target number is the element at the top right corner, the first element we check in the first pass) and the worse case of O(2n-2), which is simply O(n). For example, the worse case scenario of searching a 3*3 array would require 2n-2 (4) passes and 2n-1 (5) comparisons.

Tests:

Given a 3*3 matrix of

```
| 1   3   5  |
| 2   6   9  |
| 7  10 13 |
```

...and a target number of 7 (worst case):

```
(First, we compare the element at the top right corner to the target number)
(target = 7, x = 5)
5 == 7?
(False) → 5 < 7?
          (Result) → move down 1 row (row++)
                      (x = 9)

9 == 7?
(False) → 9 < 7?
          (False) → 9 > 7?
                    (Result) → move one left of the current element (col--)
                                (x = 6)

6 == 7?
(False) → 6 < 7?
          (Result) → move down 1 row (row++)
                      (x = 10)

10 == 7?
(False) → 10 < 7?
          (False) → 10 > 7?
                    (Result) → move one left of the current element (col--)
                                (x = 7)

7 == 7?
(Result) → True!! Wooo!
```