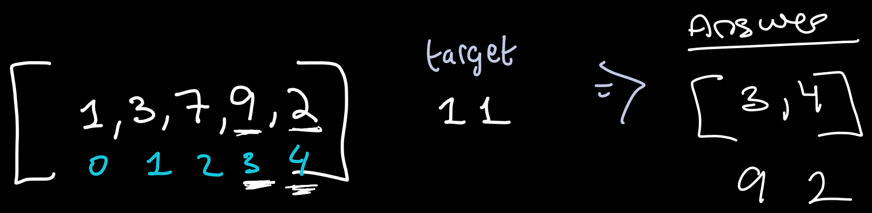
<u>Problem:</u> Given an array of integers, return the **indices** of the two #s that add up to a given target.



<u>Intro:</u>

- Verify Constraints
    - Are all #s positives? Yes
    - Duplicates? No
    - Always be a solution? No
    - What to return if no solution? null
    - Multiple pairs to add up to target? No
- Create Testcases
    - [1, 3, 7, 9, 2]   11 -> [3, 4]
    - [1, 3, 7, 9, 2].   25 -> null
    - []   1 -> null
    - [5]   5 -> null
    - [1, 6]   7 -> [0, 1]

<u>Brute Force:</u>

- Brainstorming & Pattern Observations

→ simplest method:

2 pointers + double for loop

- Pseudocode   : 2 pointer technique   P1 outer index   P2 inner index

```
Return null if array is empty or of size=1
Create array of size 2
Store first index in array
Track the first index, iterate thru array to find # that adds to target
with first index
Store second index in array
Else track 2nd index, then iterate thru rest of index
Once target found, add new index to array
Return array
Else return null
```

outer (beside "Store first index in array"); inner (under "iterate")

- Write code
- Run through testcases
- Analyze time and space complexity
    Time : outer for loop: O(n) * inner for loop: O(n) = O(n^2)
    Space: O(1)

<u>Optimal:</u>

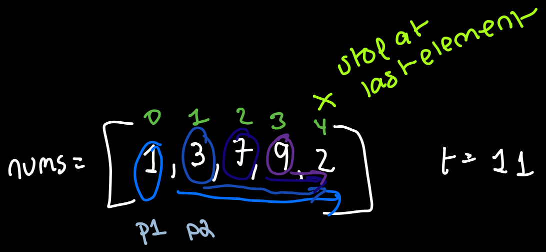
- Brainstorming & Pattern Observations
    Hint: if space significantly better (like O(1)) than time O(n^2), can improve the time complexity?

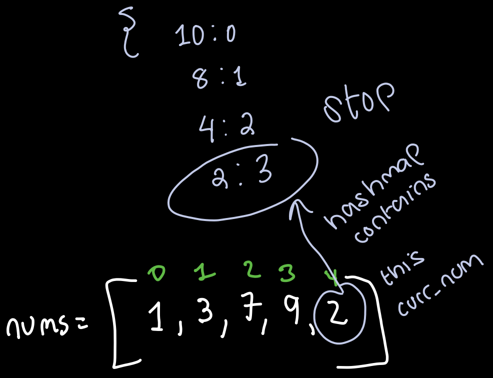
    Trade off between Time and Space Complexity: can we use more space to bring down the time complexity?

    Outer for loop: calcuate the numToFind
    Inner for loop: nums[p2] === ntf

    the inner for loop is wasteful, we can use a hash map
    why hash map? hash map lookup is O(1)

    calculate the ntf and check if the hashmap contains the ntf
    hashmap (key = ntf, value = index)





= target − nums[p1]
= 11 − num[0] = 10

3 ≠ 10
7 ≠ 10
9 ≠ 10
2 ≠ 10

The sol'n cannot contain (1)

(1) p1++ increase the index of p1
(2) p2's index = p1+1

- Pseudocode optimal solution

```
iterate thru whole array
for each index:
    check if ntf already in hashmap
    calculate the ntf
    store ntf in hashmap
```

- Write code
- Run through testcases
- Analyze time and space complexity
  - Time: O(n): for loop iterates thru all items in array: calculates and finds the ntf
  - Space: O(n): hashmap key is created for every item in array, stores ntf

Updated Emails List
rucsllc@gmail.com (NO PASSWORD ACCESS)
rutgerscsllc@gmail.com | rutger$123
~~rucsllc.web@gmail.com~~
~~csllc@dimacs.rutgers.edu~~
ddcc@dimacs.rutgers.edu | WeLoveDDCC! (Alias)
douglassdimacs@gmail.com | WeLoveDDCC!
douglassdimacscomputingcorp@gmail.com | @DDCCddcc