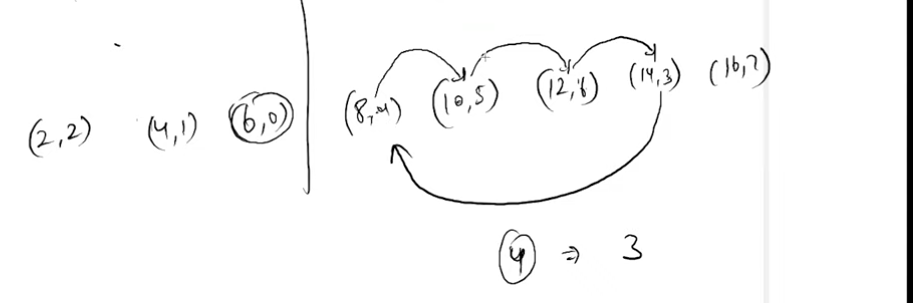
**Input:** {4, 3, 2, 1}  
**Output:** 2  
**Explanation:** Swap index 0 with 3 and 1 with 2 to form the sorted array {1, 2, 3, 4}

**Input:** {1, 5, 4, 3, 2}  
**Output:** 2

Consider this as a graph:



Steps:

1. Create a new array , where arr[i] = [val,index]
2. Sort array based on index
3. Traverse the array:
   1. Let val,index = arr[i]
   2. If index already visited or val at correct index(i==index): do nothing
   3. Otherwise, set c-len = 0 and position = i
   4. While position not in visited:
      1. Add position to visited
      2. Reset position to new\_arr[position].index
      3. Increase cycle length
   5. Res += c\_len -1(a cycle with n nodes requires n-1 swaps)
4. Return res

CODE:  
 def minSwaps(self, nums):

#Code here

# create a new array to store element, index

pair = []

for idx,num in enumerate(nums):

pair.append((num,idx))

# sort based on num-value

pair.sort(key = lambda x: x[0])

visited = set()

res = 0

for i in range(n):

val, initial\_position\_of\_val = pair[i]

if initial\_position\_of\_val in visited or initial\_position\_of\_val == i :

continue

# otherwise, start from an unvisited node(its index)

position = initial\_position\_of\_val

c\_len = 0

while(position not in visited):

# add node to visited

visited.add(position)

# move to next node to visited, which is nothing but the original position of current node

position = pair[position][1]

# increment

c\_len += 1

res += c\_len-1

return res