

## Experiment Results

### Enhancing code retrieval with human-in-the-loop through Continuous Active Learning in GPT-3

1. Easy - 5 mins - Zeroes in the list

- a. <https://practice.geeksforgeeks.org/problems/count-the-zeros2550/1/?page=11&sortBy=submissions> - 54.06% success

Given an array of size **N** consisting of only 0's and 1's. The array is **sorted** in such a manner that all the 1's are placed first and then they are followed by all the 0's. Find the **count** of all the 0's.

Best answer:

```
return arr.count(0)
```

GPT3 with human comments, untimed: 1, 1/5, 1, 1/2

```
count = 0
for i in range(n):
    if arr[i] == 0:
        count += 1
return count
```

Human-2: 1, 1, 1, 1

```
return arr.count(0)
```

Human-1: 1, 1/5, 1, 1/2

```
t = 0
for i in range(n):
    if arr[i] == 0:
        t += 1
return t
```

Human with GPT-3: 1, 1/5, 1, 1/2

```
count = 0
for i in range(n):
    if arr[i] == 0:
```

```

        count += 1
    return count

```

- b. <https://practice.geeksforgeeks.org/viewSol.php?subId=006ca28f92ffbfd13d87c4eab568397&pid=700815&user=h1m4n3hu> -

51.56% success - "Need Some Change"

Given an array **arr** of **n** positive integers. The task is to swap every **i**<sup>th</sup> element of the array with **(i+2)**<sup>th</sup> element.

Best answer:

```

for i in range(n-2):
    arr[i], arr[i+2] = arr[i+2], arr[i]

```

GPT3 with human comments, untimed: 1,  $\frac{2}{3}$ , 1,  $\frac{2}{3}$

```

for i in range(n-2):
    arr[i], arr[i+2] = arr[i+2], arr[i]
return arr

```

Human-1: 1,  $\frac{2}{3}$ , 1,  $\frac{1}{2}$

```

for i in range(n-2):
    tmp = arr[i]
    arr[i] = arr[i+2]
    arr[i+2] = tmp
return arr

```

Human-2: 1,  $\frac{2}{3}$ , 1,  $\frac{1}{2}$

```

for i in range(n-2):
    t = arr[i]
    arr[i] = arr[i+2]
    arr[i+2] = t

```

Human with GPT-3: 1,  $\frac{2}{3}$ , 1,  $\frac{2}{3}$

```

for i in range(n-2):

```

```

        arr[i], arr[i+2] = arr[i+2], arr[i]
    return arr

```

- c. <https://practice.geeksforgeeks.org/problems/array-of-alternate-ve-and-ve-nos1401/1/?page=15&sortBy=submissions> - 46.41% -

“Alternate positive and negative numbers”

Given an unsorted array **Arr** of **N** positive and negative numbers. Your task is to create an array of alternate positive and negative numbers without changing the relative order of positive and negative numbers.

Optimal answer:

```

def rearrange(self, arr, n):
    plist = []; nlist = []
    for num in arr:
        if num >= 0:
            plist.append(num)
        else:
            nlist.append(num)
    arr.clear() #clear list
    arr[:] = plist[:]
    index = 1
    for num in nlist:
        arr.insert(index, num)
        index += 2

```

GPT-3 with human comments, untimed:

```

def rearrange(self, arr, n):
    l = []
    l1 = []
    for i in arr:
        if i < 0:
            l.append(i)
        else:
            l1.append(i)
    l3 = []
    m = max(len(l), len(l1))
    if m == len(l1):
        for i in range(m):

```

```

        l3.append(l1[i])
        if i<len(l):
            l3.append(l[i])

    else:
        for i in range(m):
            if i<len(l1):
                l3.append(l1[i])
            l3.append(l[i])

    for i in l3:
        print(i,end = " ")
    arr.clear()
    arr = l3[:]

```

**Human-1:  $\frac{2}{3}$ ,  $\frac{3}{5}$ ,  $\frac{3}{4}$ ,  $\frac{2}{5}$  - WRONG**

```

def rearrange(self, arr, n):
    for i in range(n - 1):
        if i % 2 == 0:
            if arr[i] < 0:
                arr[i], arr[i+1] = arr[i+1], arr[i]
            elif arr[i] > 0:
                arr[i], arr[i+1] = arr[i+1], arr[i]
    return arr

```

**Human with GPT-3, 10 mins: 1, 13/22, 1,  $\frac{2}{5}$  - RIGHT**

```

def rearrange(self, arr, n):
    l = []
    l1 = []
    for i in arr:
        if i<0:
            l.append(i)
        else:
            l1.append(i)

```

```

l3 = []
m = max(len(l),len(l1))
if m == len(l1):
    for i in range(m):
        l3.append(l1[i])
        if i<len(l):
            l3.append(l[i])

else:
    for i in range(m):
        if i<len(l1):
            l3.append(l1[i])
        l3.append(l[i])

for i in l3:
    print(i,end = " ")
arr.clear()
arr = l3[:]

```

**Human-2: 1, 13/21, 1,  $\frac{2}{3}$**

```

def rearrange(self,arr, n):
    pos = deque()
    neg = deque()
    fin = []
    for i in arr:
        if i>= 0:
            pos.append(i)
        else:
            neg.append(i)

    k = 0
    while(len(pos)>0 and len(neg)> 0):
        arr[k] = pos.popleft()
        k+=1
        arr[k] = neg.popleft()

```

```

        k+=1
    while (len(pos)):
        arr[k] = pos.popleft()
        k+=1
    while (len(neg)):
        arr[k] = neg.popleft()
        k+=1

```

- d. <https://practice.geeksforgeeks.org/problems/next-permutation5226/1/?page=16&sortBy=submissions> - “Next Permutation” - 44.7%  
 - Implement the next permutation, which rearranges the list of numbers into Lexicographically next greater permutation of list of numbers. If such arrangement is not possible, it must be rearranged to the lowest possible order *i.e.* sorted in an ascending order. You are given a list of numbers **arr[ ]** of size **N**.

**Best answer:**

```

def nextPermutation(self, N, arr):
    i=N-1
    while i>0 and arr[i]<=arr[i-1]:
        i-=1
    arr = arr[:i]+ list(reversed(arr[i:N]))
    left=i-1
    r=i
    while(left>=0 and r<N):
        if(arr[r] > arr[left]):
            arr[left],arr[r]=arr[r],arr[left]
            break
        r+=1
    return arr

```

**GPT-3, unlimited time: 1, 12/13, 1, 5/7**

```

def nextPermutation(self, N, arr):
    # code here
    i = -1
    for j in range(N - 1):
        if arr[j] < arr[j + 1]:
            i = j
    if i == -1:
        return
    # find the first index j such that arr[j] > arr[i]
    j = -1

```

```

for k in range(N):
    if arr[k] > arr[i]:
        j = k
# swap arr[i] and arr[j]
arr[i], arr[j] = arr[j], arr[i]
# reverse the suffix
arr[i + 1:] = arr[N - 1:i:-1]
return

```

### Human with GPT-3: 1, 12/13, 1, 5/7

```

def nextPermutation(self, N, arr):
    # code here
    i = -1
    for j in range(N - 1):
        if arr[j] < arr[j + 1]:
            i = j
    if i == -1:
        return
    # find the first index j such that arr[j] > arr[i]
    j = -1
    for k in range(N):
        if arr[k] > arr[i]:
            j = k
    # swap arr[i] and arr[j]
    arr[i], arr[j] = arr[j], arr[i]
    # reverse the suffix
    arr[i + 1:] = arr[N - 1:i:-1]
    return

```

### Human-1: 1, 11/17, 1, 5/8

```

def nextPermutation(self, N, arr):
    # code here
    i = 0
    j = 1
    k = -1
    while j <= len(arr)-1:
        if arr[j] > arr[i]:
            k = i
            i += 1
            j += 1
    if k < 0:
        return arr[::-1]
    i = k + 1
    val_to_be_swapped_with_farthest_large_val = arr[k]
    index_of_farthest_value = -1
    while i <= len(arr)-1:
        if arr[i] > val_to_be_swapped_with_farthest_large_val:
            index_of_farthest_value = i

```

```

        i += 1
        arr[k], arr[index_of_farthest_value]=
arr[index_of_farthest_value], arr[k]
        # now reverse the ele after k
        i = k + 1
        j = len(arr) - 1
        while i < j:
            arr[i], arr[j] = arr[j], arr[i]
            i += 1
            j -= 1
        return arr

```

**Human-2: 1, 11/14, 1, 5/7**

```

def nextPermutation(self, N, arr):

    maxed = True
    for i in range(N-2, -1,-1):
        if arr[i]<arr[i+1]:
            maxed = False
            breakingPoint = i
            break

    if maxed==True:
        return arr[::-1]

    for i in range(N-1,0,-1):
        if arr[breakingPoint]<arr[i]:
            arr[breakingPoint],arr[i] =arr[i],arr[breakingPoint]
            break
    return arr[:breakingPoint+1]+sorted(arr[breakingPoint+1:])

```

- e. [https://practice.geeksforgeeks.org/problems/intersection-point-in-y-shapped-linked-lists/1/?page=1&company\[\]=Amazon&sortBy=submissions](https://practice.geeksforgeeks.org/problems/intersection-point-in-y-shapped-linked-lists/1/?page=1&company[]=Amazon&sortBy=submissions) - 39.55% - “Intersection Point in Y Shaped Linked Lists”

**Best answer: 12, 5**

```

def intersetPoint(head1,head2):
    a = head1
    b = head2
    while a != b:
        if a == None:
            a = head2
        else:
            a = a.next
        if b == None:
            b = head1
        else:

```



```

        b = b.next
    return a.data

```

**Human-1 - WRONG - 30/140 test cases - 3/14, 12/23, 3/14, 1/2**

```

def interserPoint(head1, head2):
    len1 = 0
    temp = head1
    while temp:
        len1 += 1
        temp = temp.next
    len2 = 0
    temp = head2
    while temp:
        len2 += 1
        temp = temp.next
    diff = abs(len1 - len2)
    if diff != 0:
        return None
    temp1 = head1
    temp2 = head2
    while diff > 0:
        temp1 = temp1.next
        temp2 = temp2.next
        diff -= 1
    while temp1 != temp2:
        temp1 = temp1.next
        temp2 = temp2.next
    return temp1.data

```

**GPT-3 infinite time: 1, 3/4, 1, 5/6**

```

def interserPoint(head1, head2):
    len1 = 0; len2 = 0
    temp1 = head1; temp2 = head2
    while temp1 is not None:
        len1 += 1
        temp1 = temp1.next
    while temp2 is not None:
        len2 += 1
        temp2 = temp2.next
    temp1 = head1; temp2 = head2
    if len1 > len2:
        for i in range(len1 - len2):
            temp1 = temp1.next
    else:
        for i in range(len2 - len1):
            temp2 = temp2.next
    while temp1 is not None and temp2 is not None:
        if temp1 == temp2:
            return temp1.data
        temp1 = temp1.next; temp2 = temp2.next
    return None

```

**Human-2: 1, 6/11, 1, 5/6**

```

def interserPoint(head1, head2):
    lt1 = head1; lt2 = head2

```

```

temp1 = head1; temp2 = head2
len1 = 0; len2 = 0
while lt1 is not None:
    lt1 = lt1.next
    len1+=1
while lt2 is not None:
    lt2 = lt2.next
    len2+=1
diff = (len1-len2) if (len1>len2) else (len2-len1)
count = 0
while(count<diff):
    if len1>len2:
        temp1 = temp1.next
        count+=1
    else:
        temp2 = temp2.next
        count+=1
while (temp1!=temp2):
    temp1 = temp1.next
    temp2 = temp2.next
return temp1.data

```

#### Human-1 with GPT-3: 1, ½, 1, 5/7

```

def interserPoint(head1,head2):
    count1=0; count2=0
    temp1=head1; temp2=head2
    while(head1):
        head1=head1.next
        count1=count1+1
    while(head2):
        head2=head2.next
        count2=count2+1
    if(count1>count2):
        diff=count1-count2
        while(diff>0):
            temp1=temp1.next
            diff=diff-1
    else:
        diff=count2-count1
        while(diff>0):
            temp2=temp2.next
            diff=diff-1
    while(temp1):
        if(temp1==temp2):
            return(temp1.data)
        temp1=temp1.next

```

```

        temp2=temp2.next
    return(-1)

```

- f. [https://practice.geeksforgeeks.org/problems/detect-cycle-in-a-directed-graph/1/?page=1&company\[\]=Amazon&sortBy=submissions](https://practice.geeksforgeeks.org/problems/detect-cycle-in-a-directed-graph/1/?page=1&company[]=Amazon&sortBy=submissions) - "Detect Cycle in a Directed graph" - 30.19%

Best answer: 17, 8

```

def isCyclic(self, V, adj):
    def dfs(v, visited, rec_stack):
        visited[v], rec_stack[v] = True, True
        for n in adj[v]:
            if not visited[n]:
                if dfs(n, visited, rec_stack):
                    return True
            elif rec_stack[n]:
                return True
        rec_stack[v] = False
        return False
    visited, rec_stack = [False] * V, [False] * V
    for node in range(V):
        if not visited[node]:
            if dfs(node, visited, rec_stack):
                return True
    return False

```

GPT-3 infinite time: 1, 17/19, 1, 1

```

def isCyclicUtil(self, u, visited, recStack, adj):
    visited[u] = True
    recStack[u] = True
    for v in adj[u]:
        if visited[v] == False:
            if self.isCyclicUtil(v, visited, recStack, adj):
                return True
        elif recStack[v] == True:
            return True
    recStack[u] = False
    return False

def isCyclic(self, V, adj):
    visited = [False] * V
    recStack = [False] * V
    for i in range(V):
        if not visited[i]:
            if self.isCyclicUtil(i, visited, recStack, adj):
                return True
    return False

```

Human-1 - WRONG: 103/401, 9/16, 2/3, 4/9

```

def isCyclic(self, V, graph):
    vstd = [False]*V
    instack = [False]*V

    def dfs(start):
        vstd[start] = True
        instack[start] = True

        for node in graph[start]:
            if instack[node]:
                return True
            if not vstd[node] and dfs(node):
                return True
        instack[start] = False

    for node in range(V):
        if not vstd[node] and dfs(node):
            return True
    return False

```

## Human-2 - WRONG: 264/401, 11/19, $\frac{3}{4}$ , $\frac{3}{4}$

```

bool check(vector<bool> &visit, vector<bool> &sv, vector<int>
adj[], int node) {
    visit[node]=1;
    sv[node]=1;
    for(auto it:adj[node]) {
        if(!visit[it]) {
            if(check(visit, sv, adj, it)) {
                return 1;
            }
        } else if(sv[it]) {
            return 1;
        }
    }
    sv[node]=0;
    return 0;
}

bool isCyclic(int n, vector<int> adj[]) {
    vector<bool> visit(n, false);
    for(int i=0; i<n; i++) {
        vector<bool> subvisit(n, false);
        if(!visit[i] && check(visit, subvisit, adj, i)) {
            return 1;
        }
    }
    return 0;
}

```

## Human-1 with GPT-3: 1, 17/19, 1, 1

```
def isCyclicUtil(self, u, visited, recStack, adj):
    visited[u] = True
    recStack[u] = True
    for v in adj[u]:
        if visited[v] == False:
            if self.isCyclicUtil(v, visited, recStack, adj):
                return True
        elif recStack[v] == True:
            return True
    recStack[u] = False
    return False

def isCyclic(self, V, adj):
    visited = [False] * V
    recStack = [False] * V
    for i in range(V):
        if not visited[i]:
            if self.isCyclicUtil(i, visited, recStack, adj):
                return True
    return False
```

<b><i>Coding Construct used</i></b>	<b><i>Cost</i></b>
<b>for loop</b>	<b>2</b>
<b>while loop</b>	<b>2</b>
<b>if:, else:, else if:</b>	<b>1 each</b>
<b>Variable assignment</b>	<b>1</b>

Questions		1	2	3	4	5	6
<b>GPT-3, infinite time</b>	Recall	1	1	1	1	1	1
	Precision	1/2	2/3	3/5	5/7	5/6	1
<b>Human-1 (Master's)</b>	Recall	1	1	3/4	1	3/14	2/3
	Precision	1/2	1/2	2/5	5/8	1/2	4/9
<b>Human-2 (PhD)</b>	Recall	1	1	1	1	1	3/4
	Precision	1	1/2	2/3	5/7	5/8	3/4
<b>Human-1 with GPT-3</b>	Recall	1	1	1	1	1	1
	Precision	1/2	2/3	3/5	5/7	5/7	1

Questions		1	2	3	4	5	6
<b>GPT-3, infinite time</b>	Recall	1	1	1	1	1	1
	Precision	1/5	2/3	13/22	12/13	3/5	17/19
<b>Human-1 (Master's)</b>	Recall	1	1	2/3	1	3/14	103/401
	Precision	1/5	2/5	3/5	11/17	12/23	9/16
<b>Human-2 (PhD)</b>	Recall	1	1	1	1	1	264/401
	Precision	1	2/5	13/21	11/14	6/11	11/19
<b>Human-1 with GPT-3</b>	Recall	1	1	1	1	1	1
	Precision	1/5	2/3	13/22	12/13	1/2	17/19

Questions	Accuracy on GfG	Problem Title
1	54.06%	<b>“Zeroes in the list”</b> - find number of zeroes in the list
2	51.56%	<b>“Need some change”</b> - swap i'th and (i+2)'th elements in the list
3	46.41%	<b>“Alternate positive and negative numbers”</b> - preserve relative order, but reorder the list to obtain alternate positive and negative numbers
4	44.7%	<b>“Next Permutation”</b> - return the lexicographically greater next sequence of the list
5	39.55%	<b>“Intersection point in Y-shaped linked lists”</b>
6	30.19%	<b>“Detect cycle in a Directed Graph”</b>