***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
2. <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, ⅕, 1, ½

count = 0

for i in range(n):

if arr[i] == 0:

count += 1

return count

Human-1: 1, 1, 1, 1

return arr.count(0)

Human-2: 1, ⅕, 1, ½

t = 0

for i in range(n):

if arr[i] == 0:

t += 1

return t

Human with GPT-3: 1, ⅕, 1, ½

count = 0

for i in range(n):

if arr[i] == 0:

count += 1

return count

1. <https://practice.geeksforgeeks.org/viewSol.php?subId=006ca28f92ffbbfd13d87c4eab568397&pid=700815&user=h1m4n3hu> - 51.56% success - “Need Some Change”

Given an array **arr** of **n** positive integers. The task is to swap every **ith** element of the array with **(i+2)**th 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, ⅔, 1, ⅔

for i in range(n-2):

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

return arr

Human-1: 1, ⅖, 1, ½

for i in range(n-2):

tmp = arr[i]

arr[i] = arr[i+2]

arr[i+2] = tmp

return arr

Human-2: 1, ⅖, 1, ½

for i in range(n-2):

t = arr[i]

arr[i] = arr[i+2]

arr[i+2] = t

Human with GPT-3: 1, ⅔, 1, ⅔

for i in range(n-2):

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

return arr

1. <https://practice.geeksforgeeks.org/problems/array-of-alternate-ve-and-ve-nos1401/1/?page=15&sortBy=submissions> - 49.41% - “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:

Human-1: ⅔, ¾, ¾, ¾ - WRONG

def rearrange(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]

else: if arr[i] > 0:

arr[i], arr[i+1] = arr[i+1], arr[i]

return arr

Human with GPT-3, 10 mins: 1, 1, 1, ⅖ - 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, 1, 1, ⅔

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

1. <https://practice.geeksforgeeks.org/problems/next-permutation5226/1/?page=16&sortBy=submissions> - “Next Permutation” - 54.7% - 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:])

1. [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%5B%5D=Amazon&sortBy=submissions) - 49.55% - “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-2 - WRONG - 30/140 test cases - 3/14, 12/23, 3/14, ½

def intersetPoint(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, ⅗, 1, ⅚

def intersetPoint(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-1: 1, 6/11, 1, ⅝

def intersetPoint(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 with GPT-3: 1, ½, 1, 5/7

def intersetPoint(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)

1. [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%5B%5D=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: 110/401, 9/16, ¾, ¾

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, ⅔, 4/9

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-2 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