Solution to TeleTrip – AIO 2023

Since we are just checking how many unique houses we've visited, we can keep a set of items which manages the houses we've visited. Every time we visit a **new/unique** house; we update the set and add its location. Once we have gone through every single instruction, the answer is the number of items in our set. For example, let's look at the first sample case **LLRRRLL**.

I'll keep a variable `loc` which tracks our current location, this is initialised as 0 which is the location of our house. We'll also create the set `visited` and initialise it with only the value 0 since we have already visited our own house (that is where we start).

```
loc = 0, visited = {0}
```

Ok the first instruction is to go left, which would mean decreasing `loc` by 1.

```
loc = -1, visited = {0}
```

Since -1 isn't in the set visited, we add it:

```
loc = -1, visited = {0, -1}
```

The next instruction is to go left again, which makes $\text{loc} = -2$. Since this isn't a value in the set we add it to visited.

```
loc = -2, visited = {0, -1, -2}
```

The next instruction is to go right, this means $\text{loc} = -1$. But since -1 is in the set (it has already been visited) we don't add any values to our set. The same goes for the next instruction. On the final move to the right, $\text{loc} = 1$. Since this is not in the set, we add it.

```
loc = 1, visited = {0, -1, -2, 1}
```

Finally, we move to the left, where $\text{loc} = 0$. Since 0 is in the set of visited values, we don't add it. The final answer is the length of visited, which is 4 so we output 4.

Written as code this is (shown on following page):

amongus_pvp – uploaded to https://github.com/amongus-pvp/ORACsolns

```python
N = int(input()) # this is the number of instructions
instructions = input() # these are the instructions

visited = set()
visited.add(0) # we add the position of our house which is 0
location = 0 # we start at position 0

# the reason i used a set and not a list is because looking up a value in a
list is very costly of efficiency
# sets are special in the fact that their lookup times are O(1) which means
when looking for an item in a set, the searchtime is incredibly fast compared
to an array
# this searchtime doesn't scale with the size of the set either, where in an
array searching for an item in a larger array can cost you more time, search-
ing for an item in a set of any size will take the exact same amount of time
for any size

for i in range(N):
    instruction = instructions[i] # we attain the relevant instruction

    if instruction == 'L': # we move to the left
        location -= 1
        if location not in visited:
            visited.add(location) # adding the location to our set of visited
values

    if instruction == 'R': # we move to the right
        location += 1
        if location not in visited:
            visited.add(location) # adding the location to our set of visited
values

    if instruction == 'T': # we teleport back to our house
        location = 0 # we dont need to check if this is in our set as our set
was intialised with 0 in it

print(len(visited)) # output the number of unique houses we visited
```