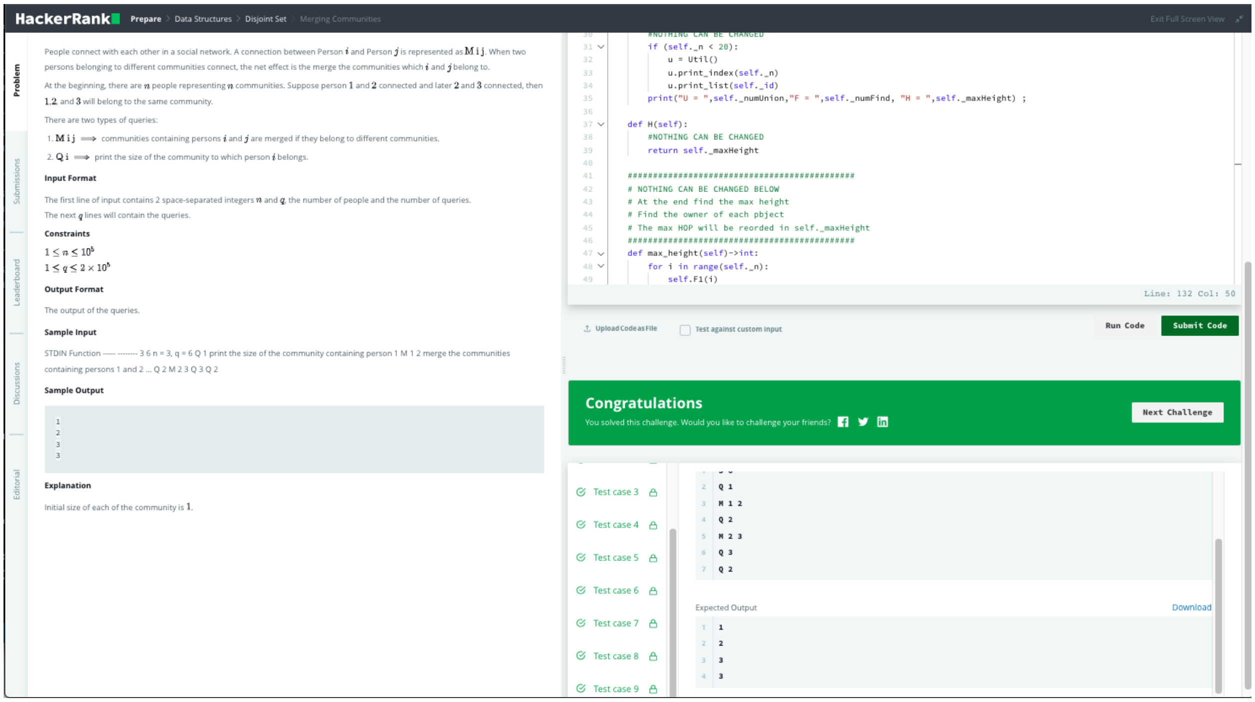
6205-HW9




```
def max_height(self)->int:
```

Time complexity

for the max_height method, since we need to iterate all the elements, and for each element, the worst case is that it needs to hop max_height times to find the parent node. Therefore, time complexity is O(n*max_height)

Space complexity

the space complexity is theta(n)

```
def num(self,a:'int')->'int':
```

Time complexity

for the num method, the time complexity is O(max_height), since the max_height ≤6, we can say that the time complexity is a constant

Space complexity

the space complexity is a constant

```
def F1(self,a:'int')->'int':
```

for the F1 method, since the time complexity is the hop times, which is O(max_height), we can say the time complexity is a constant

space complexity is a constant

```
def F2(self,a:'int',b:'int')->'bool':
```

for the F2 method, the time complexity is the same as the F1 method, therefor the time complexity is a constant, and the space complexity is a constant

```
def U(self,a:'int',b:'int')->'bool':
```

for the U method, the time complexity if the same as F2 as the comparison following has complexity of a constant. therefore the time complexity is a constant, and the space complexity is a constant.

In conclusion, the time complexity of the SUSF function is:

O(n*max_height) + O(max_height) + O(max_height) + C +C =O(n*max_height)

Since max_height is no larger than 6, therefore the time complexity is O(n)

the space complexity is Theta(n)