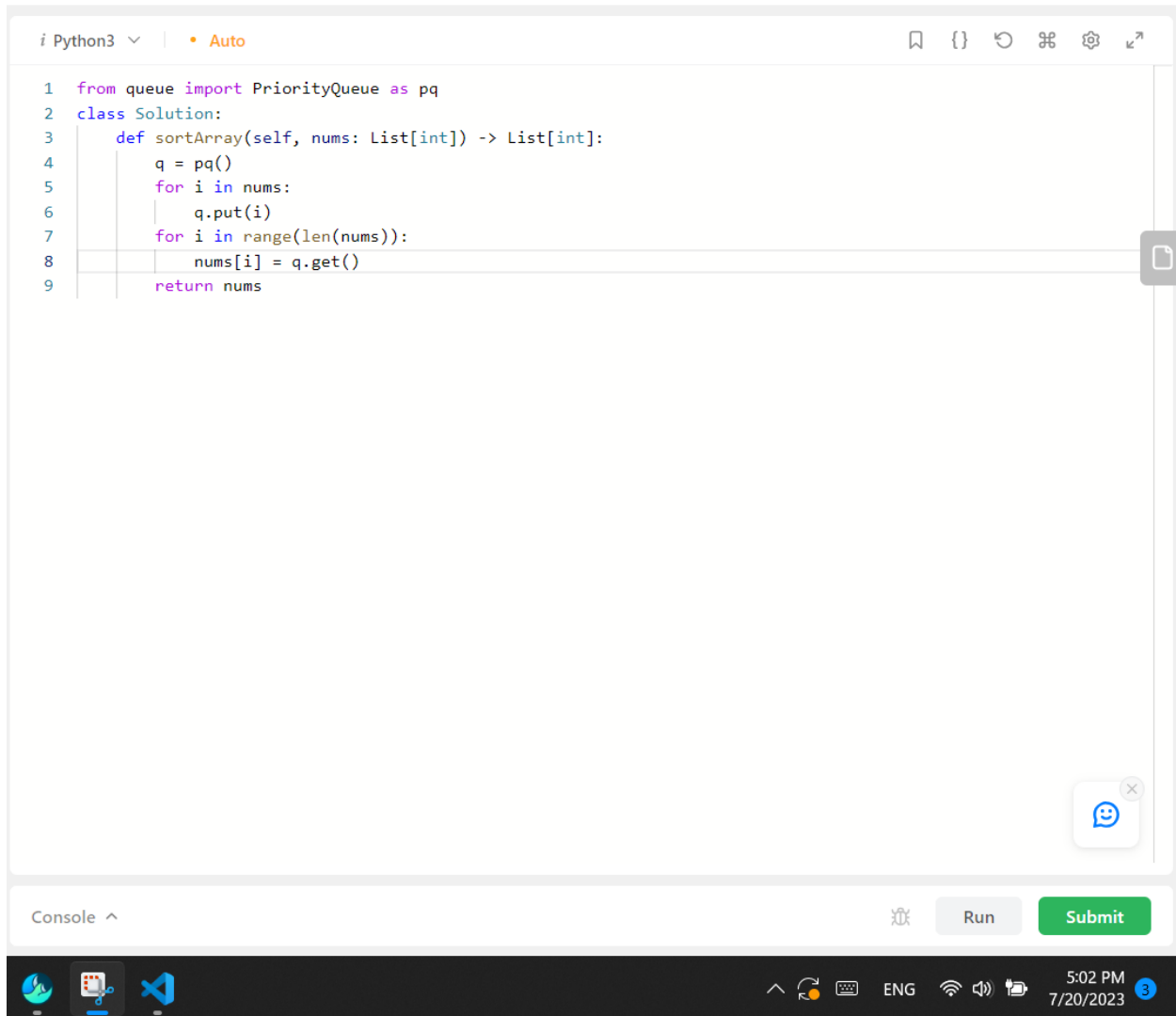


Step 1: Study the LeetCode question 912. Sort an Array - (local copy) – Medium



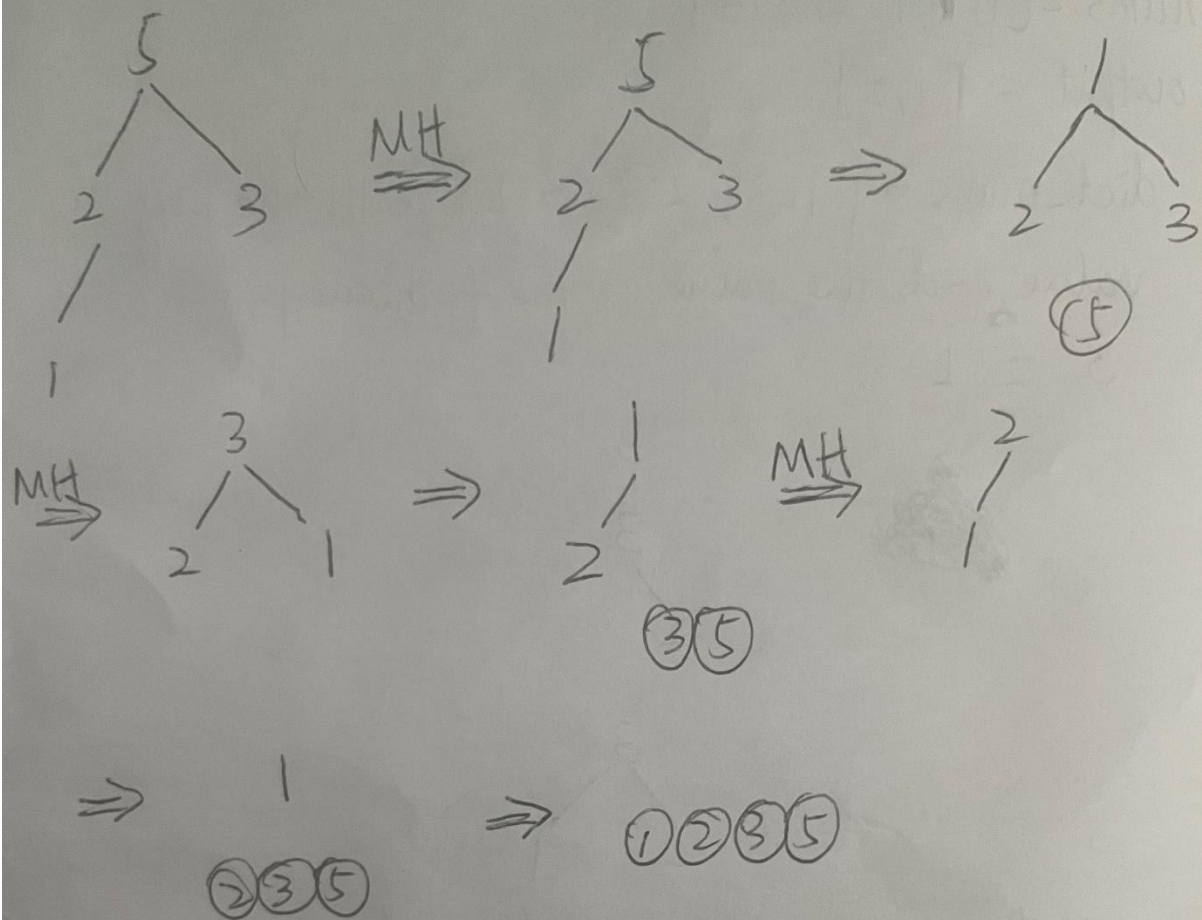
The screenshot shows a LeetCode editor interface. At the top, it says 'i Python3' and '• Auto'. The code is as follows:

```
1 from queue import PriorityQueue as pq
2 class Solution:
3     def sortArray(self, nums: List[int]) -> List[int]:
4         q = pq()
5         for i in nums:
6             q.put(i)
7         for i in range(len(nums)):
8             nums[i] = q.get()
9         return nums
```

Below the code editor is a console area with a 'Console ^' label, a 'Run' button, and a green 'Submit' button. At the bottom of the screen is a Windows taskbar with icons for a globe, a calendar, and a blue 'X' icon. The system clock shows '5:02 PM 7/20/2023'.

Step 2: Following the examples shown on Heapsort to manually solve this LeetCode problem with the test data: 'Input: nums = [5,2,3,1], Output: [1,2,3,5]'

This step is to demonstrate that you know the concept of the algorithm by showing something manually created such as diagrams.



Step 3: Ask ChatGPT: "How to use Heapsort approach to implement a Python code to solve the LeetCode question: '912. Sort an Array', with the test data: 'Input: nums = [5,2,3,1], Output: [1,2,3,5]'"

To implement the Heapsort approach in Python to solve the LeetCode question "912. Sort an Array," you can create a max-heap and repeatedly extract the maximum element to sort the array in ascending order. Here's a Python code for the problem:

```
python Copy code

def heapify(nums, n, i):
    largest = i
    left_child = 2 * i + 1
    right_child = 2 * i + 2

    if left_child < n and nums[i] < nums[left_child]:
        largest = left_child

    if right_child < n and nums[largest] < nums[right_child]:
        largest = right_child

    if largest != i:
        nums[i], nums[largest] = nums[largest], nums[i]
        heapify(nums, n, largest)

def heapSort(nums):
    n = len(nums)

    # Build a max-heap
    for i in range(n // 2 - 1, -1, -1):
        heapify(nums, n, i)

    # Extract elements from the max-heap one by one to get a sorted array
    for i in range(n - 1, 0, -1):
        nums[0], nums[i] = nums[i], nums[0]
        heapify(nums, i, 0)
```

🔄 Regenerate response

```
    return nums

# Test the function with the given test data
nums = [5, 2, 3, 1]
output = heapSort(nums)
print(output) # Output: [1, 2, 3, 5]
```

Explanation:

1. The `heapify` function is used to maintain the max-heap property in the array. It takes three parameters: `nums` (the array to be heapified), `n` (the size of the heap), and `i` (the index of the element to be considered as the root of the subtree).
2. The `heapSort` function is the main function to perform heapsort. It first builds a max-heap by calling `heapify` for each internal node in the array.
3. After building the max-heap, the function repeatedly extracts the maximum element (the root of the heap) and places it at the end of the array. Then, it calls `heapify` on the reduced heap to maintain the max-heap property again. This process continues until the array is sorted.
4. The sorted array is returned as the result.

The output of the code for the given test data is `[1, 2, 3, 5]`, which is the correct result for sorting the `nums` list in ascending order using the Heapsort algorithm.

Step 4: Implement the Python code created by ChatGPT

```
test.py X
test.py > ...
1 def heapify(nums, n, i):
2     largest = i
3     left_child = 2 * i + 1
4     right_child = 2 * i + 2
5
6     if left_child < n and nums[i] < nums[left_child]:
7         largest = left_child
8
9     if right_child < n and nums[largest] < nums[right_child]:
10        largest = right_child
11
12    if largest != i:
13        nums[i], nums[largest] = nums[largest], nums[i]
14        heapify(nums, n, largest)
15
16 def heapSort(nums):
17     n = len(nums)
18
19     # Build a max-heap
20     for i in range(n // 2 - 1, -1, -1):
21         heapify(nums, n, i)
22
23     # Extract elements from the max-heap one by one to get a sorted array
24     for i in range(n - 1, 0, -1):
25         nums[0], nums[i] = nums[i], nums[0]
26         heapify(nums, i, 0)
27
28     return nums
29
30 # Test the function with the given test data
31 nums = [5, 2, 3, 1]
32 output = heapSort(nums)
33 print(output) # Output: [1, 2, 3, 5]
34

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code
[Running] python -u "d:\MSCS\CS455 Algorithm\w9\test.py"
[1, 2, 3, 5]
[Done] exited with code 0 in 0.25 seconds

Live Share Column Selection Ln 34, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live 5:19 PM 7/20/2023
```

Step 5: Test the Python code with all the test cases provided by 912. Sort an Array - (local copy)

```
test.py X
test.py > ...
1 def heapify(nums, n, i):
2     largest = i
3     left_child = 2 * i + 1
4     right_child = 2 * i + 2
5
6     if left_child < n and nums[i] < nums[left_child]:
7         largest = left_child
8
9     if right_child < n and nums[largest] < nums[right_child]:
10        largest = right_child
11
12    if largest != i:
13        nums[i], nums[largest] = nums[largest], nums[i]
14        heapify(nums, n, largest)
15
16 def heapSort(nums):
17     n = len(nums)
18
19     # Build a max-heap
20     for i in range(n // 2 - 1, -1, -1):
21         heapify(nums, n, i)
22
23     # Extract elements from the max-heap one by one to get a sorted array
24     for i in range(n - 1, 0, -1):
25         nums[0], nums[i] = nums[i], nums[0]
26         heapify(nums, i, 0)
27
28     return nums
29
30 # Test the function with the given test data
31 nums = [5, 2, 3, 1]
32 output = heapSort(nums)
33 print(output) # Output: [1, 2, 3, 5]
34

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code
[Running] python -u "d:\MSCS\CS455 Algorithm\w9\test.py"
[1, 2, 3, 5]
[Done] exited with code 0 in 0.25 seconds

Live Share Column Selection Ln 34, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live 5:19 PM 7/20/2023
```

```
test.py x
test.py > ...
1 def heapify(nums, n, i):
2     largest = i
3     left_child = 2 * i + 1
4     right_child = 2 * i + 2
5
6     if left_child < n and nums[i] < nums[left_child]:
7         largest = left_child
8
9     if right_child < n and nums[largest] < nums[right_child]:
10        largest = right_child
11
12    if largest != i:
13        nums[i], nums[largest] = nums[largest], nums[i]
14        heapify(nums, n, largest)
15
16 def heapSort(nums):
17     n = len(nums)
18
19     # Build a max-heap
20     for i in range(n // 2 - 1, -1, -1):
21         heapify(nums, n, i)
22
23     # Extract elements from the max-heap one by one to get a sorted array
24     for i in range(n - 1, 0, -1):
25         nums[0], nums[i] = nums[i], nums[0]
26         heapify(nums, i, 0)
27
28     return nums
29
30 # Test the function with the given test data
31 nums = [5,1,1,2,0,0]
32 output = heapSort(nums)
33 print(output) # Output: [1, 2, 3, 5]
34

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code
[Running] python -u "d:\MSCS\CS455 Algorithm\w9\test.py"
[0, 0, 1, 1, 2, 5]
[Debug] exited with code 0 in 0.133 seconds

Live Share Column Selection Ln 31, Col 21 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live
5:19 PM 7/20/2023
```