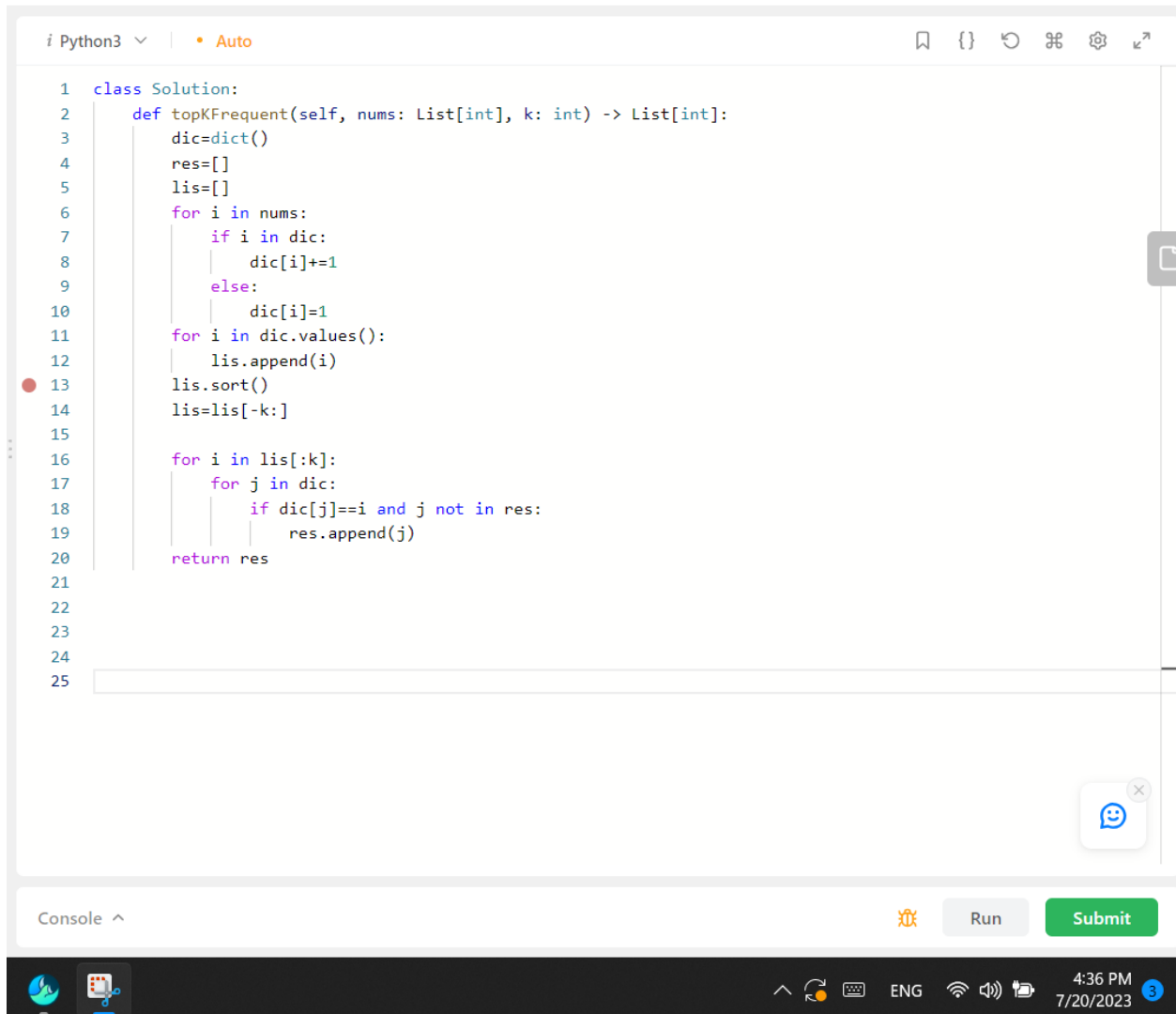


Step 1: Study 347. Top K Frequent Elements - (local copy) – Medium



```
1 class Solution:
2     def topKFrequent(self, nums: List[int], k: int) -> List[int]:
3         dic=dict()
4         res=[]
5         lis=[]
6         for i in nums:
7             if i in dic:
8                 dic[i]+=1
9             else:
10                dic[i]=1
11        for i in dic.values():
12            lis.append(i)
13        lis.sort()
14        lis=lis[-k:]
15
16        for i in lis[:k]:
17            for j in dic:
18                if dic[j]==i and j not in res:
19                    res.append(j)
20        return res
21
22
23
24
25
```

Console ^

Run Submit

4:36 PM 7/20/2023

Step 2: Following the examples shown on Heap to manually solve the LeetCode problem with the test data: 'Input: nums = [1,1,1,2,2,3], k = 2; Output: [1,2]'. This step is to demonstrate that you know the concept of the algorithm by showing something manually created such as diagrams.

nums = [1, 1, 1, 2, 2, 3], output = [1, 2]

dict_nums = {1: 3, 2: 2, 3: 1}

With the key as the element of nums, and the value as their corresponding frequency, we have 3, 2, 1.

3
3
/
2
3

2 1

Thus, the frequency in the descending order is 3, 2, 1 with the top two frequencies as 3 and 2. And we take back 3 and 2 to the dict_nums and identify the top two highest frequent numbers as 1 and 2.

Step 3: Ask ChatGPT: "How to use Heap approach to implement a Python code to solve the LeetCode question: '347. Top K Frequent Elements', with the test data: 'Input: nums = [1,1,1,2,2,3], k = 2; Output: [1,2]'"

To implement the Heap approach in Python for solving the LeetCode question "347. Top K Frequent Elements," you can use the `heapq` module, which provides functions to create and manipulate heaps. Here's a Python code for the problem:

```
python Copy code

import heapq
from collections import Counter

def topKFrequent(nums, k):
    # Count the frequency of each element in the nums list
    counter = Counter(nums)


    # Create a min-heap to store the k most frequent elements
    heap = []

    # Add the elements to the heap with their frequencies
    for num, freq in counter.items():
        if len(heap) < k:
            heapq.heappush(heap, (freq, num))
        else:
            if freq > heap[0][0]:
                heapq.heappop(heap)
                heapq.heappush(heap, (freq, num))

    # Extract the k most frequent elements from the heap
    result = [num for freq, num in heap]

    return result

# Test the function with the given test data
nums = [1, 1, 1, 2, 2, 3]
k = 2
```

 Regenerate response

```
output = topKFrequent(nums, k)
print(output) # Output: [1, 2]
```

Explanation:

1. First, we use the `Counter` class from the `collections` module to count the frequency of each element in the `nums` list.
2. Next, we create an empty min-heap to store the k most frequent elements. A min-heap is used because we want to keep track of the k largest elements based on their frequencies.
3. We iterate through the elements and frequencies obtained from the `Counter`. For each element, we check if the heap is not yet full (`size < k`). If the heap is not full, we simply add the element and its frequency to the heap. If the heap is already full, we compare the frequency of the current element with the smallest frequency element in the heap (the root of the heap). If the frequency of the current element is greater than the smallest frequency in the heap, we remove the smallest element from the heap and add the current element.
4. After processing all elements, the heap will contain the k most frequent elements. We extract these elements from the heap and store them in the `result` list.
5. Finally, we return the `result`, which contains the k most frequent elements in descending order of their frequencies.

The output of the code for the given test data is `[1, 2]`, which is the correct result for the top 2 most frequent elements in the `nums` list.

Step 4: Implement the Python code created by ChatGPT

```
test.py ×
test.py > ...
1 import heapq
2 from collections import Counter
3
4 def topKFrequent(nums, k):
5     # Count the frequency of each element in the nums list
6     counter = Counter(nums)
7
8     # Create a min-heap to store the k most frequent elements
9     heap = []
10
11     # Add the elements to the heap with their frequencies
12     for num, freq in counter.items():
13         if len(heap) < k:
14             heapq.heappush(heap, (freq, num))
15         else:
16             if freq > heap[0][0]:
17                 heapq.heappop(heap)
18                 heapq.heappush(heap, (freq, num))
19
20     # Extract the k most frequent elements from the heap
21     result = [num for freq, num in heap]
22
23     return result
24
25 # Test the function with the given test data
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code

```
[Running] python -u "d:\MSCS\CS455 Algorithm\w9\test.py"
[2, 1]

[Done] exited with code=0 in 0.257 seconds
```

Column Selection Ln 30, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live

4:47 PM
7/20/2023

Step 5: Test the Python code with all the test cases provided by 347. Top K Frequent Elements - (local copy)

test.py

test.py > topKFrequent

```
1 import heapq
2 from collections import Counter
3
4 def topKFrequent(nums, k):
5     # Count the frequency of each element in the nums list
6     counter = Counter(nums)
7
8     # Create a min-heap to store the k most frequent elements
9     heap = []
10
11     # Add the elements to the heap with their frequencies
12     for num, freq in counter.items():
13         if len(heap) < k:
14             heapq.heappush(heap, (freq, num))
15         else:
16             if freq > heap[0][0]:
17                 heapq.heappop(heap)
18                 heapq.heappush(heap, (freq, num))
19
20     # Extract the k most frequent elements from the heap
21     result = [num for freq, num in heap]
22
23     return result
24
25 # Test the function with the given test data
26 nums = [1, 1, 1, 2, 2, 3]
27 k = 2
28 output = topKFrequent(nums, k)
29 print(output) # Output: [1, 2]
30
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code

[Running] python -u "d:\MSCS\CS455 Algorithm\w9\test.py"

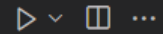
[2, 1]

[Done] exited with code=0 in 0.257 seconds

Column Selection Ln 20, Col 23 Spaces: 4 UTF-8 CRLF Python 3.11.3 64-bit Go Live

4:49 PM
7/20/2023

test.py ×



test.py > ...

```
1 import heapq
2 from collections import Counter
3
4 def topKFrequent(nums, k):
5     # Count the frequency of each element in the nums list
6     counter = Counter(nums)
7
8     # Create a min-heap to store the k most frequent elements
9     heap = []
10
11     # Add the elements to the heap with their frequencies
12     for num, freq in counter.items():
13         if len(heap) < k:
14             heapq.heappush(heap, (freq, num))
15         else:
16             if freq > heap[0][0]:
17                 heapq.heappop(heap)
18                 heapq.heappush(heap, (freq, num))
19
20     # Extract the k most frequent elements from the heap
21     result = [num for freq, num in heap]
22
23     return result
24
25 # Test the function with the given test data
26 nums = [1]
27 k = 1
28 output = topKFrequent(nums, k)
29 print(output) # Output: [1, 2]
30
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

Code



[Running] python -u "d:\MSCS\CS455 Algorithm\w9\test.py"

[1]

[Done] exited with code=0 in 0.165 seconds

Column Selection

Ln 27, Col 6

Spaces: 4

UTF-8

CRLF

{ } Python

3.11.3 64-bit



Go Live



4:50 PM
7/20/2023