

HANDS ON-5

DESIGN ANALYSIS AND ALGORITHMS

RASHMITHA RAMASANI

ID:1002233393

HEAP SORT: Heap Sort is a sorting technique that relies on a binary heap structure to organize elements in an array. The method consists of two key steps: constructing the heap and then extracting elements from it.

Time Complexity:

- **Building the heap** takes $O(n)$ time.
- **Heapifying the heap** takes $O(\log n)$ time, and this happens for each of the n elements, making the sorting phase $O(n \log n)$.

Given handson description:

The ability to build the heap initially (build_min_heap):

- **Method: heapify_list():** This method constructs a min heap from an unsorted list by heapifying the elements starting from the last non-leaf node to the root. This achieves the heap-building process.

The ability to heapify:

- **Method: _balance_heap():** This method ensures the heap property is maintained after elements are added or removed. It is used when building the heap and during insertions/removals to restore the heap's structure.

The ability to get and remove ("pop") the root node from the heap (and re-heapify):

- **Method: extract_minimum():** This method removes and returns the root node, which is the smallest element in the min heap. It then re-heapifies the structure to maintain the heap property after removal.

The heap is generic and can store different data types:

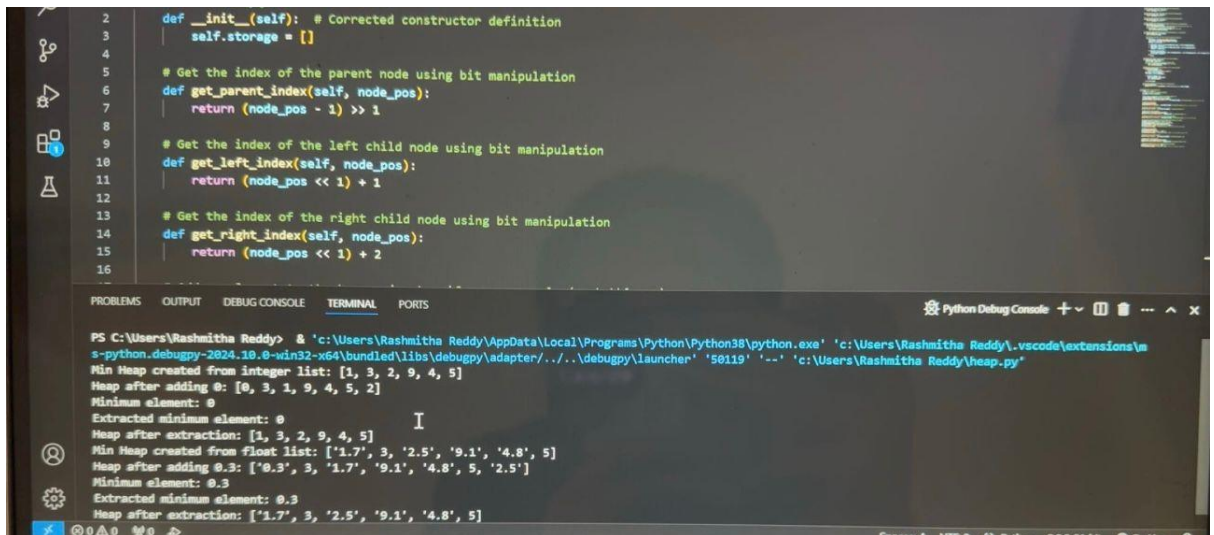
- The implementation uses a Python list (self.storage) that can hold any data type as long as it supports comparisons using the $<$ operator. It works with both integers and floats, as shown in the test cases.

Demonstrated functionality:

- **Building the heap:** The heapify_list() method correctly demonstrates building a heap from unsorted integer and float lists, producing the expected heap structure.
- **Inserting elements:** The add_element() method correctly inserts new elements into the heap and reorganizes the structure as needed.

- **Extracting the minimum element:** The `extract_minimum()` method removes and returns the root node, then re-heapifies the structure to maintain the min heap's properties. This demonstrates that all the required functionality has been implemented.

OUTPUT:



```

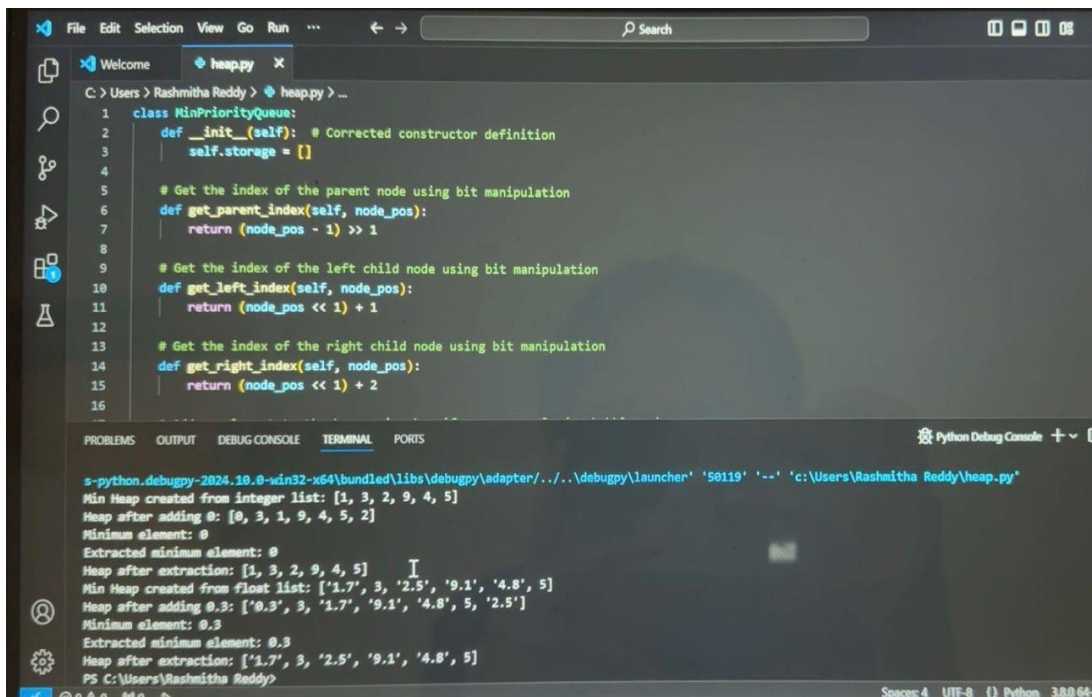
2      def __init__(self): # Corrected constructor definition
3          self.storage = []
4
5          # Get the index of the parent node using bit manipulation
6      def get_parent_index(self, node_pos):
7          return (node_pos - 1) >> 1
8
9          # Get the index of the left child node using bit manipulation
10     def get_left_index(self, node_pos):
11         return (node_pos << 1) + 1
12
13         # Get the index of the right child node using bit manipulation
14     def get_right_index(self, node_pos):
15         return (node_pos << 1) + 2
16

```

```

PS C:\Users\Rashmitha Reddy> & 'c:\Users\Rashmitha Reddy\AppData\Local\Programs\Python\Python38\python.exe' 'c:\Users\Rashmitha Reddy\.vscode\extensions\ms-python.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher' '50119' '--' 'c:\Users\Rashmitha Reddy\heap.py'
Min Heap created from integer list: [1, 3, 2, 9, 4, 5]
Heap after adding 0: [0, 3, 1, 9, 4, 5, 2]
Minimum element: 0
Extracted minimum element: 0
Heap after extraction: [1, 3, 2, 9, 4, 5]
Min Heap created from float list: ['1.7', 3, '2.5', '9.1', '4.8', 5]
Heap after adding 0.3: ['0.3', 3, '1.7', '9.1', '4.8', 5, '2.5']
Minimum element: 0.3
Extracted minimum element: 0.3
Heap after extraction: ['1.7', 3, '2.5', '9.1', '4.8', 5]

```



```

1 class MinPriorityQueue:
2     def __init__(self): # Corrected constructor definition
3         self.storage = []
4
5         # Get the index of the parent node using bit manipulation
6     def get_parent_index(self, node_pos):
7         return (node_pos - 1) >> 1
8
9         # Get the index of the left child node using bit manipulation
10    def get_left_index(self, node_pos):
11        return (node_pos << 1) + 1
12
13        # Get the index of the right child node using bit manipulation
14    def get_right_index(self, node_pos):
15        return (node_pos << 1) + 2
16

```

```

s-python.debugpy-2024.10.0-win32-x64\bundled\libs\debugpy\adapter\..\..\debugpy\launcher' '50119' '--' 'c:\Users\Rashmitha Reddy\heap.py'
Min Heap created from integer list: [1, 3, 2, 9, 4, 5]
Heap after adding 0: [0, 3, 1, 9, 4, 5, 2]
Minimum element: 0
Extracted minimum element: 0
Heap after extraction: [1, 3, 2, 9, 4, 5]
Min Heap created from float list: ['1.7', 3, '2.5', '9.1', '4.8', 5]
Heap after adding 0.3: ['0.3', 3, '1.7', '9.1', '4.8', 5, '2.5']
Minimum element: 0.3
Extracted minimum element: 0.3
Heap after extraction: ['1.7', 3, '2.5', '9.1', '4.8', 5]
PS C:\Users\Rashmitha Reddy>

```