Implementation of empty: 5 points

Very simple, if the size of the heap is 0 means it’s empty, it will return True, and False otherwise

Implementation of min: 5 points

For the min function, it will return the value of the root because this is a Min Heap

Implementation of parent: 5 points

Implementation of left\_child: 5 points

Implementation of right\_child: 5 points

Parent, leftchild and right child method will return the index of the parent, left or right child of the current index x.

Implementation of swap: 5 points

Swap will swap the value between index x and y

Implementation of insert+upheap: 15 points

Insert will first check if the heap is full capacity, if not, it will add the new value to the last element of the array and then call upheap to move the new node to the right location.

Upheap will then check if the value of the current node is smaller than the parent’s value. If yes, it will have to swap with the parent, then recursively call upheap again to move the new node up to it’s correct location

Implementation of deletemin+downheap: 15 points

Deletemin will swap the root with the last element to delete the root and then call downheap to update the heap.

The downheap will move the node down the heap. If no children exist or the index of both left and right child larger than the current size, it will return. Or if the current node has value less than it’s children then it also return. Otherwise, it will swap the current node value with the minimum value of the two children and then recursively call itself until it find the right location for the new node.

Implementation of heapsort: 10 points

Heapsort basically insert all element in array A to the heap, the heap is min heap so it will auto sort the element. Then, it will gradually copy the root value back to the array A and call deletemin to get the next minimum value in the heap.

Passes all pytests: 10 points

1. Analysis of timings: 10 pointassImplementation of empty: 5 points
2. Implementation of min: 5 points
3. Implementation of parent: 5 points
4. Implementation of left\_child: 5 points
5. Implementation of right\_child: 5 points
6. Implementation of swap: 5 points
7. Implementation of insert+upheap: 15 points
8. Implementation of deletemin+downheap: 15 points
9. Implementation of heapsort: 10 points
10. Passes all pytests: 10 points
11. Collection of timings: 10 points
12. Analysis of timings: 10 points

I imported and added the heapsort to the sortcode and update the timesort. As we can see from the output, Quicksort would be the fastest one if we ignore the builtin sort. The Quicksort is also quite easy to code and implement as well. Bubblesort and insertion sort is the fastest to code but the time cost is too long. Heapsort is only faster than bubblesort and insertion sort because it has many other operations in it like downheap, upheap, insert and delete and so on. So overally, quicksort is my best approach.