# Assignment 3

Manan Patel

September 2024

**1: Design this data structure by specifying the variables (array, counter, ...) involved..**

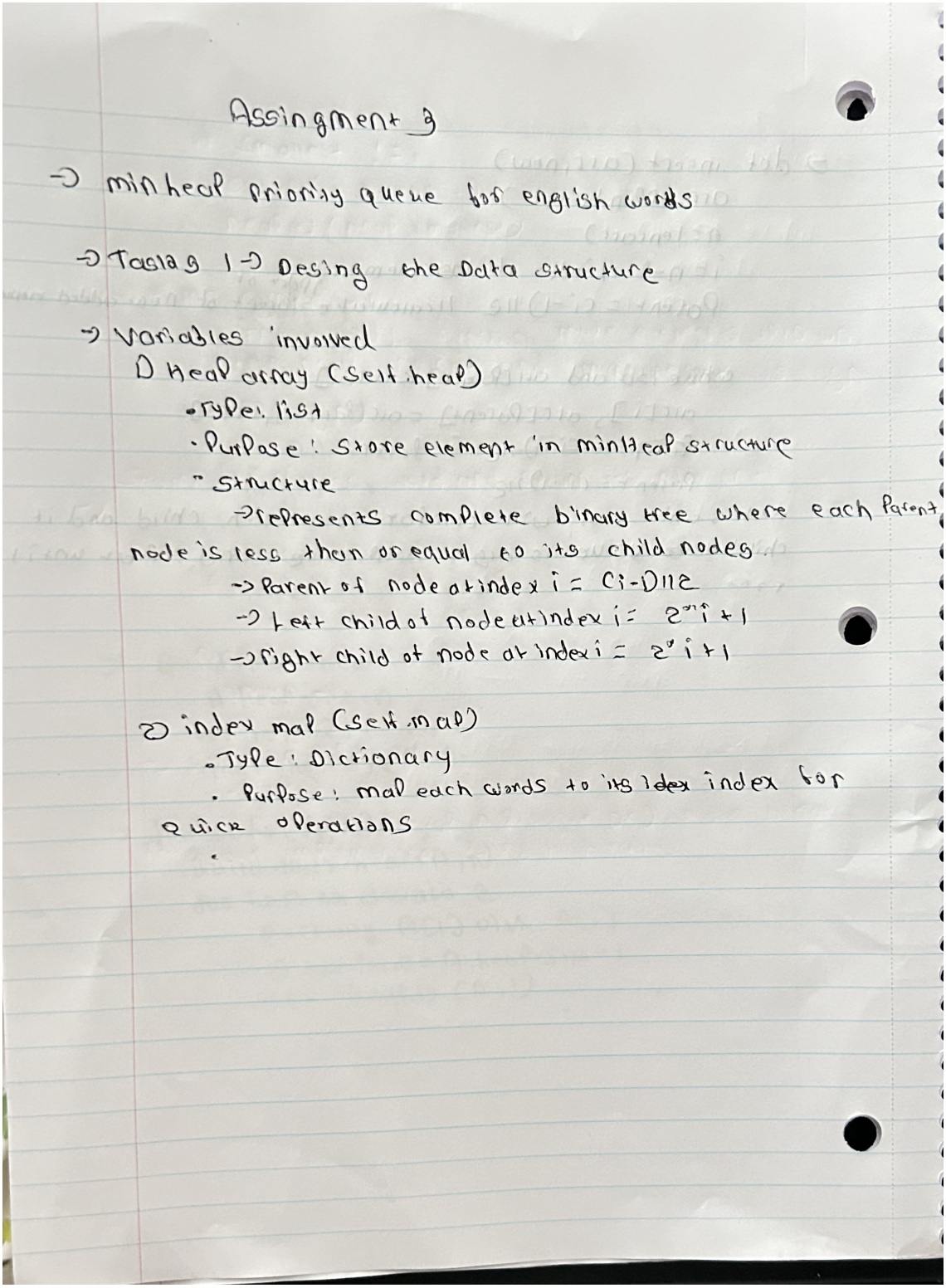
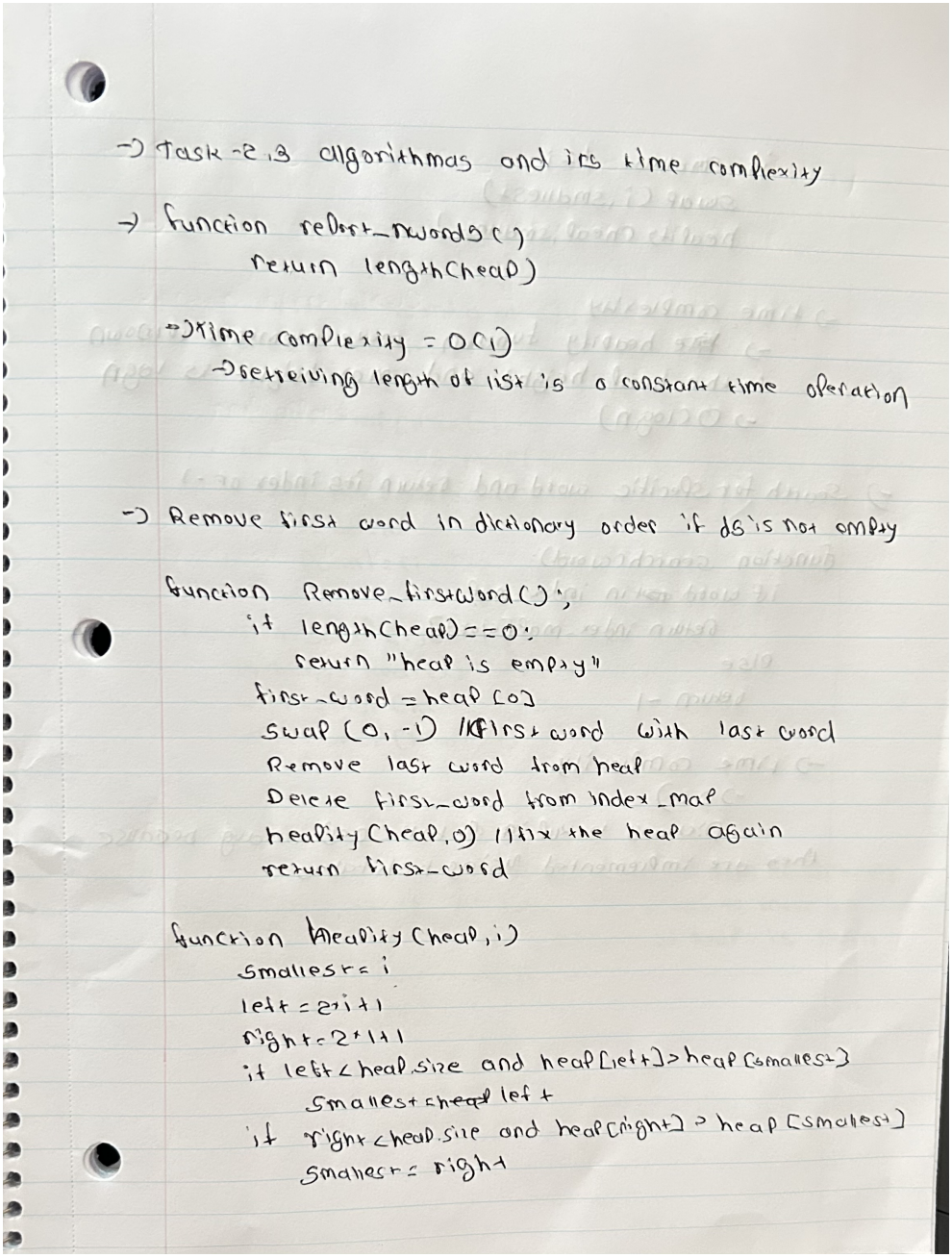



Figure 1: Answer of Task 1

**2-3. Write the most efficient algorithms you can find for each of the required operation. Analyze the (worst-case) time complexity of each algorithm.**



-> Task-2,3 algorithmas and its time complexity

-> function reboot_words()
    return length(heap)

->Time complexity = O(1)
  ->Retreiving length of list is a constant time operation

-> Remove first word in dictionary order if ds is not empty

function Remove_firstWord();
    if length(heap)==0:
      return "heap is empty"
    first_word = heap[0]
    swap(0,-1) //first word with last word
    Remove last word from heap
    Delete first_word from index_map
    heapify(heap,0) //fix the heap again
    return first_word

function heapify(heap,i)
    smallest = i
    left = 2*i+1
    right = 2*i+1
    if left < heap.size and heap[left] > heap[smallest]
        smallest = left
    if right < heap.size and heap[right] > heap[smallest]
        smallest = right

Figure 2: Answer of Task 2

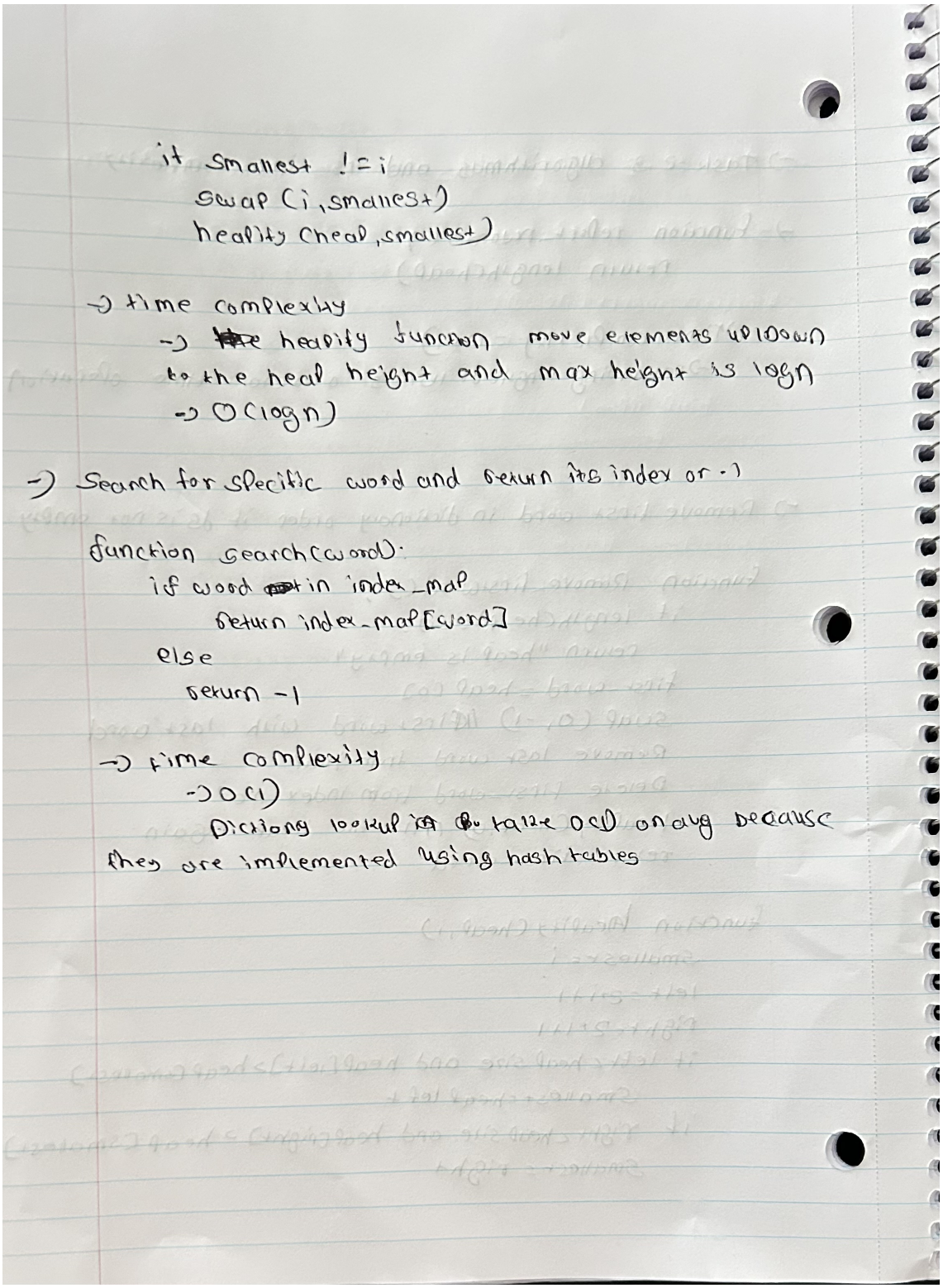



Figure 3: Answer of Task 2

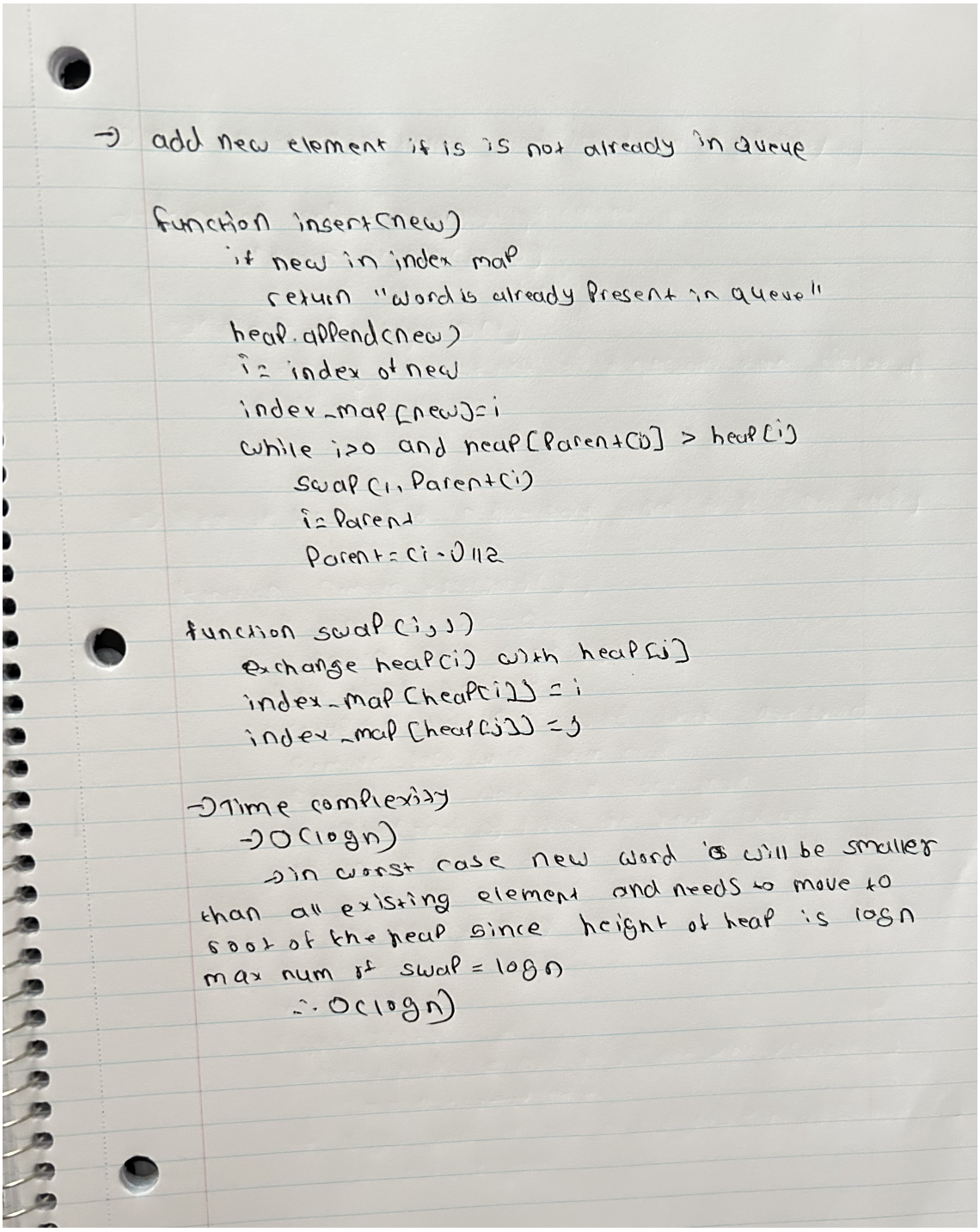



Figure 4: Answer of Task 2

| Operation | Worst-Case Time Complexity |
|---|---|
| Insert a new word | $O(\log n)$ |
| Remove the first word | $O(\log n)$ |
| Search for a specific word | $O(1)$ |
| Report the number of words | $O(1)$ |

Table 1: Worst-Case Time Complexity of Each Operation

**4-5: Implement the data structure with its operations. Use a program to test the correctness of the design.**

**Program Output**

```
if __name__ == "__main__":
    pq = priorityqueue()
    pq.insert("apple")
    pq.insert("cherry")
    pq.insert("pineapple")
    pq.insert("banana")
    # Print the current state of the queue
    pq.print()
    print()
    # Report the number of words
    pq.report_nwords()
    print(pq.insert("banana"))
    # Remove the first word in dictionary order
    print(f"first word is : {pq.remove_first_word()}")
    # Print the queue after removal
    pq.print()
    print()
    # Report the number of words after removal
    pq.report_nwords()
    # Search for a specific word
    print(f"word banana is present at: {pq.search('banana')} index")
    # Search for a word not in the queue
    print(pq.search("durian"))
```

✓ 0.0s

```
Current heap: ['apple', 'banana', 'pineapple', 'cherry']

Number of words in the queue is: 4
Word  banana is already present in the queue
first word is : apple
Current heap: ['banana', 'cherry', 'pineapple']

Number of words in the queue is: 3
word banana is present at: 0 index
-1
```

Figure 5: Answer of Task 2