**Group 5: *Return 0;***

Abdur Rahman Khan(11940020)

Anupam Kumar(11940160)

K Satwik Talluri(11940600)

Q1.

Time complexity: O(n!)

Space complexity : O(n^2)

Time: There are n rows and a queen can be placed in row 1 in n ways, row 2 in n-1 ways, row 3 in n-2 and so on.

Steps = n\*(n-1)\*(n-2)....2\*1 = n! in worst case.

Stack access is done in constant time and is done n! times

So Time complexity: O(n!)

Space: The 2D grid array takes n^2 space

Stack's size can go up to atmost n

So Space complexity : O(n^2)

Q2.

Time complexity: O(2^(n^2))

Space complexity : O(n^2)

Time: For each box, there are 8 choices and there are n^n boxes.

Steps = O(8^(n^2)) ~ O(2^(n^2)) in worst case

Stack access is done in constant time and it is done 8^(n^2) times

Space: The 2D grid array takes n^2 space

Stack's size can go up to atmost n^2

Q3.

Time complexity: O(n)

Space complexity : O(n)

Time: There are 2 loops that run n times each.

Queue access is done in constant time and is done n times

Space : array of size n for input.

Queue whose size won’t exceed n

Q4. Time complexity: O(n^2.log(n))

Space complexity : O(n)

Time : The heapify function for priority queue takes O(n.log(n)) time which is called whenever the enqueue function is called utmost n times which makes it n^2.log(n). The deleteRoot() function takes O(n log(n)) time and other functions take less time.

Therefore, time complexity is O(n^2.log(n)).

Space : Queue of size n and two arrays of size n

Q5. Time complexity: O((n+m)^2) (n=size of list 1, m==size of list2)

Space complexity : O(x) where, x=max(n,m)

Time : The insertion of lists takes O(n) , O(m) time each and the merging process also takes O(n) time. The sorting process takes O((n+m)^2) time after the lists are merged and the other functions take O(n) time making the overall time complexity as O((n+m)^2).

Space : The main lists comprise the two input lists of size m and n and a constant number of other temporary lists are made , other space is taken by the temporary variables.

So, Space Complexity is O(x) where, x=max(n,m) .

Q6.Time complexity: O(n)

Space complexity : O(n)

Time : The insertion takes O(n) time and since the list is already sorted , the sorting process is eliminated and hence the overall time complexity remains O(n).

Space : Input linked list of size n and a constant number of other temporary list of same size and constant number of other temporary variables. Hence, the space complexity remains O(n).

Q7. Time Complexity -> O(d\*n\*logn)

Space complexity-> O(n\*logn)

Here d= Maximum number of digit in a number

Time complexity= As the loop is running for number of digit(d) times and for each iteration, it is performing merge sort on linked list( O(n\*logn)). So Total time taken is digit times mergesort, which is O(d\*n\*logn)

Space Complexity= As only addition space required is in storing linked list which is O(n) and other additional space required is in merge sort which is O(n\*logn). SO total additional space required is (n+ n\*logn) which is equal to O(n\*logn).

Q8. Time Complexity -> O(n^2)

Space complexity-> O(n)

Time complexity= As first loop is partially sorting array in chunks in linked list using insertion sort in O(n^2) and again Insertion sort is applied on a partially sorted array, so it takes O(n^2) in worst case and O(n) in average case ( as the array is already almost sorted). So total time complexity would be O(n^2) in the worst case.

Space Complexity= As only additional space required is in storing linked list which is O(n) and no other additional space required is in insertion sort which is O(n\*logn). So total space complexity is O(n).