R-4.2 **Answer:**

|  |
| --- |
| Algorithm mergeSort**(**S**,**C**)**  Input**:**sequence S with n elements**,** Comparator C**.**  Ouput**:** Sequence S sorted according to C**.**    **if(**S**.**size**()** **>** 1**)**  S1 **<-** partition**(**S**,**0**,**n**/**2**-**1**)**  S2 **<-** partition**(**S**,**n**/**2**,**n**)**  mergeSort**(**S1**,**C**)**  mergeSort**(**S2**,**C**)**  S**->**merge**(**S1**,**S2**)** |
| Algorithm partition**(**S**,**i**,**j**)**  Input**:**Sequence S with the position of start and end **for** making partition  Output**:** Sequence S1 with the elements from i to j    S1**<-**empty sequence  **while** i **<=** j **do**  S1**.**insertLast**(**S**.**elementAtRank**(**i**))**  i **<-** i **+** 1  **return** S1 |
| Algorithm merge**(**A**,** B**,** C**)**  Input**:**sequences A and B with n**/**2 elements each**,**comparator C  Output**:**sorted sequence of A U B    S **<-** empty sequence  **while** !A**.**isEmpty**()** **^** !B**.**isEmpty**()** **do**  **if** C**.**isLessThan**(** B**.**first**().**element**(),**A**.**first**().**element**()** **)** then  S**.**insertLast**(**B**.**remove**(**B**.**first**()))**  **else**  S**.**insertLast**(**A**.**remove**(**A**.**first**()))**  **while** not A**.**isEmpty**()** **do**  S**.**insertLast**(**A**.**remove**(**A**.**first**()))**  **while** not B**.**isEmpty**()** **do**  S**.**insertLast**(**B**.**remove**(**B**.**first**()))**  **return** S |

R-4.5 **Answer:**

|  |
| --- |
| Algorithm removeDuplicateAndUnion**(**A**,** B**)**  Input**:**sequences A and B with n elements each  Output**:**sorted sequence of A U B    S **<-** empty sequence  **while** **!**A**.**isEmpty**()** **^** **!**B**.**isEmpty**()** **do**  **if** B**.**first**().**element**()** **<** A**.**first**().**element**()** then  S**.**insertLast**(**B**.**remove**(**B**.**first**()))**  **else** B**.**first**().**element**()** **>** A**.**first**().**element**()** then  S**.**insertLast**(**A**.**remove**(**A**.**first**()))**  **else**  S**.**insertLast**(**A**.**remove**(**A**.**first**()))**  B**.**remove**(**B**.**first**())**  **return** S |

R-4. **Answer:**

If the middle element of a sorted sequence ‘S’ is selected as a pivot then,

-Size of both Lower Partition (L) and Greater partition (G) will be always at least S/4.

-So the height of the quick-sort tree will be **log4/3n**

-The running time for each depth is **O(n)**

**-Therefore,** total running time of quick sort will be **O(nlog4/3n)🡪O(nlogn).**

C-4.10 **Answer:**

|  |
| --- |
| Algorithm electionWinner**(**S**)**  Input**:** n**-**element sequence S where each element represent a different vote  Ouput**:** ID of winning candidate  mergeSort**(**S**,**C**)**  winnerId **<-** S**.**first**()**  maxVote **<-** 0  previousId **<-** S**.**first**()**  **while** **!**S**.**isEmpty**()** **do**  currentId **<-** S**.**remove**(**S**.**first**())**  **if** currentId **!=** previousId then  **if** maxVote **<** noOfVote  maxVote **<-** noOfVote  winnerId **<-** currentId  **else**  previousId **<-** currentId  noOfVote**<-** noOfVote **+** 1 |