|  |  |  |
| --- | --- | --- |
|  | Name: Md Habibur Rony  Student ID: 984582  Weekday: Week 1- Day 5 |  |

Answer to the Q. No. R-4.2:

Algorithm margeSort(s,c)

Input:S is the secquence of n elements and C is the comparator

Output: S the sorted sequence of n elements

if s.size() = 1 then

return s

(s1,s2)<--pertition(s,n/2)

s1<--margeSort(s1,c)

s2<--margeSort(s2,c)

S<--marge(s1,s2, c)

return s

Answer to the Q. No. R-4.5:

|  |  |
| --- | --- |
| Algorithm GetCountRemovingDuplicate(A, B)  Input: A,B sequence of elements  Output: count value of the merging elements  S1 <- RemoveDuplication(A)  S2 <- RemoveDuplication(B)  i<--0  while i<s1.size() do  s.insert(s1[i])  count <--count+1 i<--i+1  i<--0  while i<s2.size() do  s.insert(s2[i])  count <--count+1 i<--i+1  return count | O(n)  O(n)  O(1)  O(n)  O(n)  O(n)  O(n)  O(1)  O(n)  O(n)  O(n)  O(n)  T(n) = O(n) |
|  |  |
| Algorithm RemoveDuplication(S)  Input: S sequence of elements  Output: S1 Sequence of elements without duplication  previous<--s[0]  s1.insert(s[0])  i<--0  while i<s.size() do  if previous != s[i] then  s1.insert(s[i])  endIf  i<--i+1  return s1 | O(1)  O(1)  O(1)  O(n)  O(n)  O(n)  O(n)  O(1)  T(n) = O(n) |

Answer to the Q. No. C-4.9:

The best case of the Quicksort is O(n log n) and worst case is O(n2) . If the pivot is in the middle of the element range then it is good with probability 1/2. Therefore, in this case the run time will be O(n logn)

Answer to the Q. No. C-4.10:

|  |  |
| --- | --- |
| Algorithm CountingVoteForGettingWiner(s)  Input: S is a list of elements  Output: Winer ID  ss<-MargeSort(s,c)  previousVote<--0  previousId<--ss[0]  CurrentVote<--0  sz<-s.sidze()  currentSize <--0  winerId<-Nul  while currentSize <sz do  if previousId = ss[courentSize] then  CurrentVote <-- CurrentVote+1  else  if currentVote >previousVote then  previousVote<- correntVote  previousId<- currentId  currentVote<-1  endIf  endIf  currentId<-- ss[sz]  currentSize <- currentSize+1  if currentVote>previousVote then  winerId<- currentId  else  winerId<- previousId  endIf  return winerId | O(nlogn)  O(1)  O(1)  O(1)  O(1)  O(1)  O(1)  O(n)  O(n)  O(n)  O(n)  O(n)  O(n)  O(n)  O(n)  O(n)  O(1)  O(1)  O(1)  O(1)  T(n) = O(nlogn) |