|  |
| --- |
| **Data Structures and Algorithms**  ***Section*: BSCE2021 Assignment # 4 *Total marks*: 100**  ***Name*** : ***\_\_NIMRA MAQBOOL Roll number* : \_BSCE21012\_** |

***Submission:***

• *Email instructor or TA if there are any questions. You cannot look at others’ solutions or use others’ solutions, however, you can discuss it with each other. Plagiarism will be dealt with according to the course policy.*

*• Submission after due time will not be accepted.*

**There should be a Report explaining your code and highlighting the results. Follow this naming convention for your report RollNumber\_Assignment#.pdf e.g BSCE21001\_Assignment3.pdf.**

**TASK:**

Write an in-depth time complexity analysis of radix sort and count sort.

|  |
| --- |
| **FUNCTION.H:**  #include <iostream>  using namespace std;  class arrayList {  public:  int sizeOfArray; //declaring  int NoOfElement; //declaring  int \*array; public:   arrayList(int s) {  sizeOfArray = s; //copying  NoOfElement = 0; //placing it to zero  array = new int[sizeOfArray];  for (int i = 0; i < sizeOfArray; i++) {  array[i] = 0; //setting values to zero  }  }  int getMax(int arr[], int size1) {  int max;  max = arr[0];  for (int i = 0; i < size1; i++) {  if (max < arr[i]) {  max = arr[i]; //finding the max element  }  }  cout << "\nTHE MAX ELEMENT IN THE ARRAY = " << max << endl;  return max;  }   void countSort(int arr[], int size1) {   cout << "ENTER SIZE = ";  cin >> size1; //taking size  cout << "ENTER ELEMENTS = ";  for (int i = 0; i < size1; i++) {  cin >> arr[i]; //taking input from the user  }   cout << "THE ARRAY BEFORE SORTING = ";  for (int i = 0; i < size1; i++) {  cout << arr[i] << " "; //displaying  }  int max = getMax(arr, size1);   int count[max];  for (int i = 0; i <= max; ++i) {  count[i] = 0; //putting it equal to zero  }  for (int i = 0; i < size1; i++) {  count[arr[i]]++; //counting the no. of elements at that index  }  int j = 0; //incrementer for arr array in which the input is taken  int i = 0; //incrementer for count array  while (i <= max) { //to check that i is less than max  if (count[i] > 0) { //checking that the count is greater than zero  arr[j] = i; //putting the value at i  count[i] = count[i] - 1; //decrementing the counter at that index  j++;  } else {  i++; //if count is less than zero then increment the i and check the next index  }  }  cout << "THE ARRAY AFTER SORTING = ";  for (int i = 0; i < size1; i++) {  cout << arr[i] << " "; //displaying  }  cout << endl;   }   void count(int arr[], int size1, int i) {  int output[size1]; //declaring an array to store  int count[20] = {0}; //declaring array to count  int k = 0;  while (k < size1) {  ++count[(arr[k] / i) % 10]; //we are taking the mode to check the least element  k++;  }  int j=1;  while (j < 10) {  count[j] += count[j - 1]; //counting in the count array  j++;  }  for (int j = size1 - 1; j >= 0; j--) {  output[count[(arr[j] / i) % 10] - 1] = arr[j];  count[(arr[j] / i) % 10]--;  }  for (int j = 0; j < size1; j++) {  arr[j] = output[j]; //copying the output array in the original array  }  }   void radixSort(int arr[], int size1) {  int max = getMax(arr, size1); //calling the max function and storing in the max element  int i=1; //declaring  while(max/i>0){  count(arr, size1, i); //calling the count function  i\*=10; //to go to the previous digit  }  } };  **MAIN.CPP:**  #include <iostream> #include "Functions.h"  using namespace std;  // int main() {   int opt;  cout << "PLEASE ENTER YOUR CHOICE." << endl; //displaying options  cout << "1.COUNT SORT." << endl;  cout << "2.RADIX SORT." << endl;  cout << "3.EXIT." << endl;  cin >> opt;   if (opt == 1) {  arrayList A(5);  int size1;  int arr[size1];  A.countSort(arr, size1); //calling the function  }  if (opt == 2) {  arrayList A(5);  int size1; //declaring  cout<<"ENTER SIZE OF ARRAY = ";  cin>>size1; //taking size  int arr[size1];  cout<<"ENTER ELEMENTS = ";  for(int j=0;j<size1;j++){  cin>>arr[j]; //taking input in the array  }  cout<<endl;  cout<<"ARRAY BEFORE SWAPPING = ";  for(int j=0;j<size1;j++){  cout<<arr[j]<<" "; //showing the elements of the array before swapping  }  cout<<endl;  A.radixSort(arr,size1); //calling function  cout<<"ARRAY AFTER SWAPPING = ";  for(int j=0;j<size1;j++){  cout<<arr[j]<<" "; //showing the elements of the array after swapping  }  cout<<endl;  }  if (opt == 3) {  cout << "YOU CHOOSE TO EXIT..." << endl;  exit(2);  }  return 0; }  **COUNT SORT TIME COMPLEXITY:**  void countSort(int arr[], int size1) {  cout << "ENTER SIZE = ";  cin >> size1;   cout << "ENTER ELEMENTS = ";  for (int i = 0; i < size1; i++) {  cin >> arr[i];   }  cout << "THE ARRAY BEFORE SORTING = ";  for (int i = 0; i < size1; i++) {  cout << arr[i] << " ";   }  int max = getMax(arr, size1);  int count[max];  for (int i = 0; i <= max; ++i) {  count[i] = 0;   }  for (int i = 0; i < size1; i++) {  count[arr[i]]++;   }  int j = 0;   int i = 0;   while (i <= max) {   if (count[i] > 0) {   arr[j] = i;   count[i] = count[i] - 1;   j++;  } else {  i++;   }  }  cout << "THE ARRAY AFTER SORTING = ";  for (int i = 0; i < size1; i++) {  cout << arr[i] << " ";   }  cout << endl;  }  **COUNT REPITITION TOTAL**  **For taking input:**  **1 1 1**  **For first for loop:**  **1 1 1**  **1 n(i<size) n**  **1 n (increment) n**  **For second loop:**  **1 1 1**  **1 n(i<size) n**  **1 n (increment) n**  **Calling max function:**  **1 n n**  **Passing max:**  **1 k k**  **Counting the number of elements on the respective index:**  **1 1 1**  **1 n(i<size) n**  **1 n (increment) n**  **Declaration:**  **1 1 1**  **1 1 1**  **For while loop:**  **1 k(i<=max) k**  **Last for loop:**  **1 1 1**  **1 n(i<size) n**  **1 n (increment) n**  **Total time complexity in avg case =O(n+K)**   * **Best case time complexity = O(n+K).** * **Worst case time complexity = O(n+K).** * **Space complexity is O(max/K), as it is dependent of the input max number, as the new array is of max’s size.**   **RADIX SORT TIME COMPLEXITY:**  void count(int arr[], int size1, int i) {  int output[size1];   int count[20] = {0};   int k = 0;  while (k < size1) {  ++count[(arr[k] / i) % 10];   k++;  }  int j=1;  while (j < 10) {  count[j] += count[j - 1];   j++;  }  for (int j = size1 - 1; j >= 0; j--) {  output[count[(arr[j] / i) % 10] - 1] = arr[j];  count[(arr[j] / i) % 10]--;  }  for (int j = 0; j < size1; j++) {  arr[j] = output[j];   } } void radixSort(int arr[], int size1) {  int max = getMax(arr, size1);   int i=1;   while(max/i>0){  count(arr, size1, i);   i\*=10;   } }  **COUNT REPITITION TOTAL**  **Declaration:**  **1 1 1**  **1 k k**  **For the first while loop:**  **1 n n**  **1 n n**  **For the second while loop:**  **1 k k**  **For the third while loop:**  **1 1 1**  **1 n-1 n-1**  **1 n n**  **For the fourth while loop:**  **1 n n**  **1 n n**  **For max function:**  **1 n n**  **IN RADIX SORT FUNCTION:**  **For declaration:**  **1 1 1**  **For calling in loop:**  **1 1 1**  **1 n-1 n-1**  **1 1 1**  **1 1 1**  **1 d d**  **for avg case time complexity = O(d(n+K)).**  **The worst case of radix sort happens when all the numbers have same number of digits except one, which consists of larger number of digits.**  **If no. of digits in that biggest number happens to be n then the runtime complexity is O(n^2), and worst case of counting sort is O(n+K).**  **Now the best case occurs when all numbers given in input have same no. of digits. Best case time complexity is O(dn).** |