**RadixSort and FindSmallestInterval**

1. Consider the following array A=[234, 29, 88, 456, 674, 123, 547, 867, 745]. Run **radix sort** on this array. Show the intermediate steps of the algorithm and the final result in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Initial Array** | **After 1st Iteration** | **After 2nd Iteration** | **After 3rd Iteration (Sorted Array)** |
| **234** | 123 | 123 | 29 |
| **29** | 234 | 29 | 88 |
| **88** | 674 | 234 | 123 |
| **456** | 745 | 745 | 234 |
| **674** | 456 | 547 | 456 |
| **123** | 547 | 456 | 547 |
| **547** | 867 | 867 | 674 |
| **867** | 88 | 674 | 745 |
| **745** | 29 | 88 | 867 |

1. You are given a list of “n” numbers A = [a0, a1, …, an-1] and an integer k >=2, and you are to find the smallest interval [x..y] such that at least k numbers of the list lie within this interval. (The size of the interval [x..y] is y-x.) For example, if the array A is [3.1, 6.2, 10, 5.8, 2.2, 6.0, 7.3], and k = 4 then the interval [5.8..7.3] is the smallest interval containing 4 elements of the array (namely {5.8, 6.0, 6.2, 7.3}). You may assume that k<=n. Give an O(nlogn) algorithm named FindSmallestInterval(int A[], int N, int k) for this problem. Your algorithm must print the interval at the end. Implement your algorithm and analyze its running time.

|  |
| --- |
| void FindSmallestInterval(double A[], int N, int k){  HeapSort(A, N);    double diff = A[k-1] - A[0];  int i=0, iter=0;  while(i<N) {  int temp= k+i-1;  if(N==temp)  break;  if(A[temp] – A[i] < diff) {  diff = A[temp] – A[i];  iter=i;  }  i++;  } //end while  for(i=iter;i>iter+k;i++) {  cout << A[i];  if(I != iter+k-1)  cout<< “, ”;  } //end for  } //end FindSmallestInterval |