Assignment no.2

Q.1

Implement radix sort algorithm using arrays for the input list given below. Deduce the time complexity T(n) for the best, worst and average cases.

CODE:

*#include* <stdio.h>

*#include* <stdlib.h>

void display(int output*[]*, int elements)

{

*for* (int i = 0; i < elements; i++)

{

printf("%d\t", output[i]);

}

printf("\n");

}

void countSort(int exp, int arr*[]*, int elements)

{

int max = 0;

*for* (int i = 0; i < elements; i++)

{

*if* ((arr[i] / exp) % 10 > max)

{

max = (arr[i] / exp) % 10;

}

}

int \*count = (int \*)calloc(max + 1, sizeof(int));

*for* (int i = 0; i < elements; i++)

{

count[(arr[i] / exp) % 10]++;

}

*for* (int i = 1; i <= max; i++)

{

count[i] += count[i - 1];

}

int \*output = (int \*)malloc(elements \* sizeof(int));

*for* (int i = elements - 1; i >= 0; i--)

{

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

*for* (int i = 0; i < elements; i++)

{

arr[i] = output[i];

}

free(count);

free(output);

*if* (max > 0)

{

countSort(exp \* 10, arr, elements);

}

}

int main()

{

int elements;

printf("Enter the number of elements: ");

scanf("%d", &elements);

int \*arr = (int \*)malloc(elements \* sizeof(int));

printf("Enter each element one by one:\n");

*for* (int i = 0; i < elements; i++)

{

scanf("%d", &arr[i]);

}

countSort(1, arr, elements);

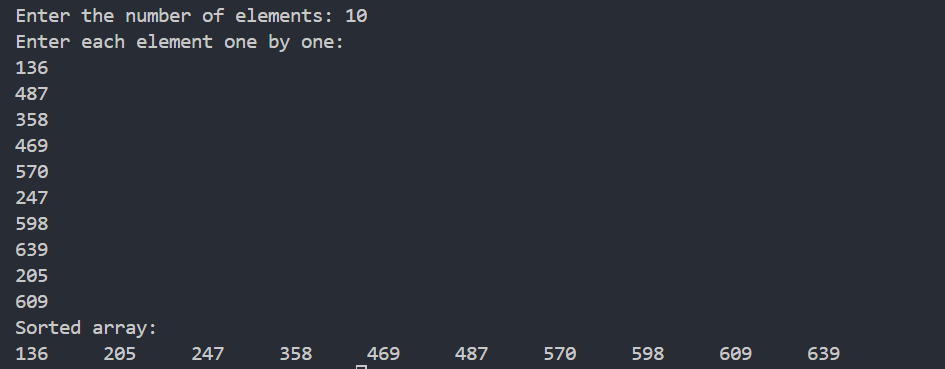
printf("Sorted array:\n");

display(arr, elements);

free(arr);

*return* 0;

}

OUTPUT:  


Q.2

Use linked list for implementation of Radix sort for the same elements given above. Deduce the time complexity T(n) for the best, worst and average cases.

*#include* <stdio.h>

*#include* <stdlib.h>

*struct* node

{

int data;

*struct* node \*next;

};

*typedef* *struct* node nd;

nd \*create(nd \*head, int n, int t)

{

nd \*tmp, \*new = malloc(sizeof(nd));

int temp = (n / t) % 10;

new->data = n;

new->next = NULL;

*if* (head == NULL)

head = new;

*else*

{

*if* (((head->data / t) % 10) > temp)

{

new->next = head;

head = new;

}

*else*

{

tmp = head;

*while* (tmp->next != NULL && (tmp->next->data / t) % 10 <= temp)

{

tmp = tmp->next;

}

new->next = tmp->next;

tmp->next = new;

}

}

*return* head;

}

void radix\_sort(int \*a, int n)

{

int i, j, k, temp, copy[n];

nd \*head;

temp = a[0];

*for* (i = 1; i < n; i++)

{

*if* (a[i] > temp)

temp = a[i];

}

int d = 0, rem;

*while* (temp != 0)

{

d++;

rem = temp % 10;

temp = temp / 10;

}

int t = 1;

*for* (i = 1; i <= d; i++)

{

head = NULL;

*for* (j = 0; j < n; j++)

{

head = create(head, a[j], t);

}

nd \*tmp = head;

*for* (j = 0; j < n && tmp != NULL; j++)

{

a[j] = tmp->data;

tmp = tmp->next;

}

t = t \* 10;

}

}

int main()

{

int n;

printf("Enter number of elements:\n");

scanf("%d", &n);

int a[n], i;

printf("Enter the elements one by one:");

*for* (i = 0; i < n; i++)

{

scanf("%d", &a[i]);

}

printf("Entered elments:\n");

*for* (i = 0; i < n; i++){

printf("%d ", a[i]);

}

printf("\n");

radix\_sort(a, n);

printf("Sorted Elements:\n");

*for* (i = 0; i < n; i++){

printf("%d ", a[i]);

}

}

OUTPUT:

