Experiment 6

WAP to Perform the Comparative analysis of Quick sort and Heap Quick.

Quick sort

Program:-

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
int partition(int a[], int first, int last)
{
  int i, j, pivot, temp;
  if (first < last)
  {
    pivot = first;
    i = first;
    j = last;
    while (i < j)
    {
      while (a[i] <= a[pivot] && i < last)
        i++;
      while (a[j] > a[pivot])
        j--;
      if (i < j)
      {
        temp = a[i];
        a[i] = a[j];
        a[j] = temp;
      }
    }
    temp = a[pivot];
    a[pivot] = a[j];
```

```
    a[j] = temp;
    return j;
  }
}

void quicksort(int a[], int first, int last)
{
  int p;
  if (first < last)
  {
    p = partition(a, first, last);
    quicksort(a, first, p - 1);
    quicksort(a, p + 1, last);
  }
}
int main()
{
  int a[50000], n, i, randNum;
  double time;
  clock_t start, end;
  printf("Enter the total input size:");
  scanf("%d", &n);
  start = clock();
  for (i = 0; i < n; i++)
  {
    randNum = rand() % 10000;
    a[i] = randNum;
    printf("%d\t", a[i]);
  }
  quicksort(a, 0, n - 1);
```

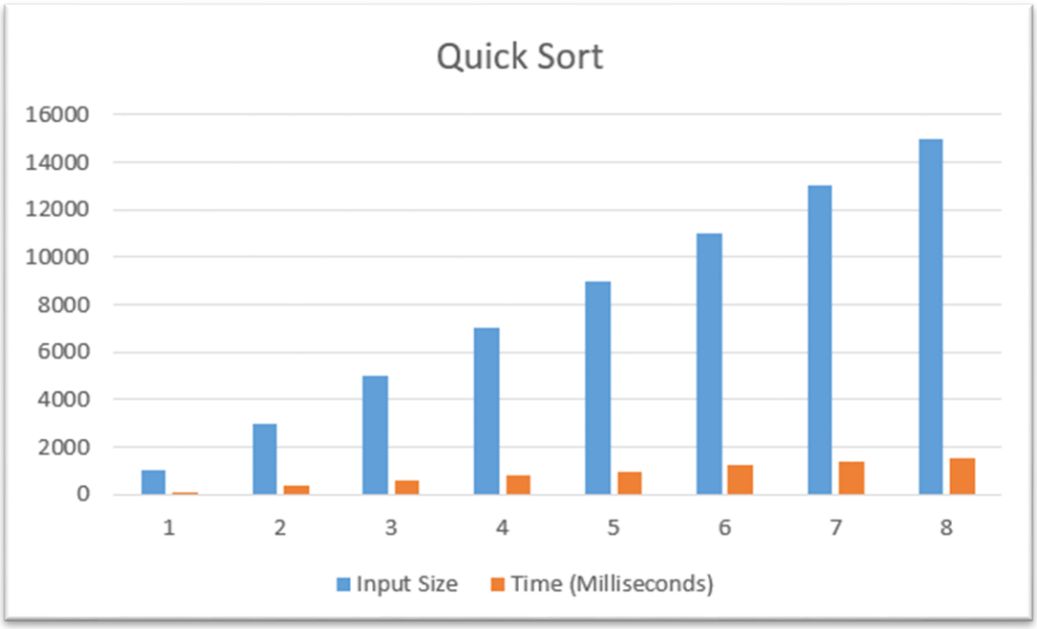
```
  printf("\nSorted list:");
  for (i = 0; i < n; i++)
  {
    printf("%d \t", a[i]);
  }
  end = clock();
  time = ((double)(end - start) * 1000) / CLOCKS_PER_SEC;
  printf("\nTime=%lf millseconds", time);
  return 0;
}
```

Result Analysis and Discussion:

This experiment has been conducted in a 64-bit system with 16 GB RAM and Processor 12th Gen Intel(R) Core (TM) i5-12500H 3.10 GHz. The algorithm is implemented in C programming language in Visual Studio Code 1.83.1 Code Editor. In this experiment the algorithm to sort an array of size “n” using Quick Sort Technique has been implemented and executed for different value of n. During this experiment for different value of n, the time taken by the algorithm has been measured and tabulated as shown in table below.

Input Size	**Time (Milliseconds)**
1000	120
3000	405
5000	585
7000	816
9000	981
11000	1235
13000	1387
15000	1546

The graph shown below is the plot of input n and the time in milliseconds taken by the algorithm while running on a system recorded in table above.



Based on the above table and graph it is clearly seen that the size of array n has linear relationship with the time taken by the system to sort an array.

Heap sort

Program:-

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
int main()
{
  int heap[500000], no, i, j, c, root, temp;
  int randNum;
  double time;
  clock_t start, end;
  printf("\n Enter no of elements :");
  scanf("%d", &no);
  start = clock();
  for (i = 0; i < no; i++)
  {
    randNum = (rand() % 10000);
    heap[i] = randNum;
  }
  for (i = 1; i < no; i++)
  {
    c = i;
    do {
      root = (c - 1) / 2;
      if (heap[root] < heap[c])
      {
        temp = heap[root];
        heap[root] = heap[c];
        heap[c] = temp;
```

```
        }
        c = root;
    } while (c != 0);
}
printf("\n Heap array: ");
for (i = 0; i < no; i++)
{
    printf("%d\t", heap[i]);
}
for (j = no - 1; j >= 0; j--)
{
    temp = heap[0];
    heap[0] = heap[j];
    heap[j] = temp;
    root = 0;
    do
    {
        c = 2 * root + 1; /* left node of root element */
        if ((heap[c] < heap[c + 1]) && c < j - 1)
        {
            c++;
        }
        if (heap[root] < heap[c] && c < j)
        { /* again rearrange to max heap array */
            temp = heap[root];
            heap[root] = heap[c];
            heap[c] = temp;
        }
        root = c;
    } while (c < j);
```

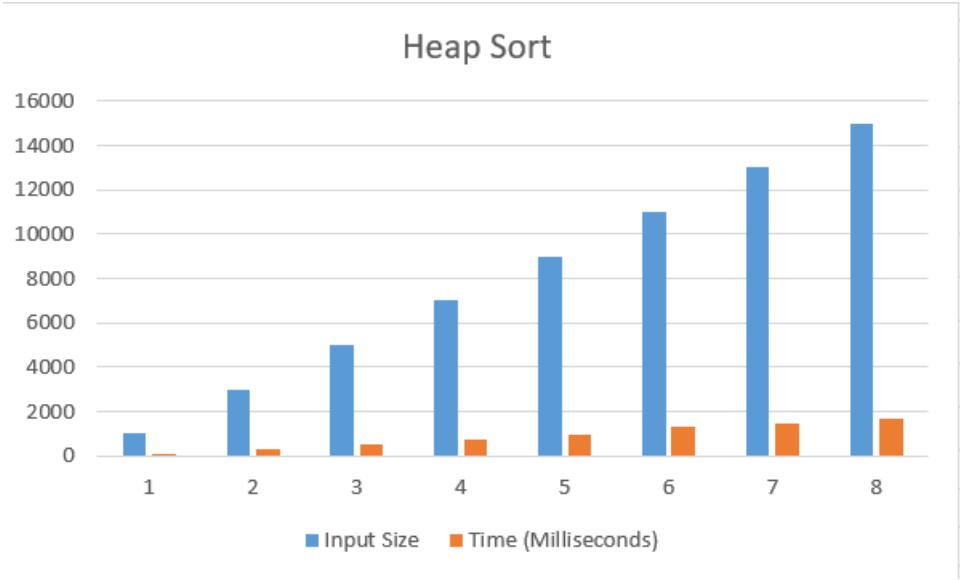
```
  }
  printf("\n The sorted array is: ");
  for (i = 0; i < no; i++)
  {
     printf("\t %d", heap[i]);
  }
  end = clock();
  time = ((double)(end - start) * 1000) / CLOCKS_PER_SEC;
  printf("\n Time=%lf milliseconds", time);
}
```

Result Analysis and Discussion:

This experiment has been conducted in a 64-bit system with 16 GB RAM and Processor 12th Gen Intel(R) Core (TM) i5-12500H 3.10 GHz. The algorithm is implemented in C programming language in Visual Studio Code 1.83.1 Code Editor. In this experiment the algorithm to sort an array of size “n” using Heap Sort Technique has been implemented and executed for different value of n. During this experiment for different value of n, the time taken by the algorithm has been measured and tabulated as shown in table below.

Input Size	**Time (Milliseconds)**
1000	123
3000	337
5000	510
7000	765
9000	979
11000	1280
13000	1434
15000	1656

The graph shown below is the plot of input n and the time in milliseconds taken by the algorithm while running on a system recorded in table above.



Based on the above table and graph it is clearly seen that the size of array n has linear relationship with the time taken by the system to sort an array.

Conclusion:

In this experiment, it has been found that the size of input "n" has linear relationship with the time taken by the system to sort the array .So, Quick sort is better than Heap sort algorithm to sort an array.