Quick sort:

6/6/24

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void swap(int *p1, int* p2)
{
  int temp;
  temp = *p1;
  *p1 = *p2;
  *p2 = temp;
}
int partition(int arr[], int low, int high)
{
  int pivot = arr[high];
  int i = (low-1);
  for (int j=low; j<high; j++){
      if(arr[j] < pivot){
          i++;
          swap(&arr[i], &arr[j]);
      }
  }
  swap(&arr[i+1], &arr[high]);
  return (i+1);
}
void quicksort(int arr[], int low, int high)
{
  if(low < high) {
          int pi = partition(arr, low, high);
          quickSort(arr, low, pi-1);
          quickSort(arr, pi+1, high);
  }
}
int main()
{
  int a[15000], n, i, j, ch, temp;
  clock_t start, end;
  while(1)
  {
    printf("\n1: for manual entry of N value & array elements");
    printf("\n2: To display time taken for sorting number of elements
                N in the range 500 to 14500");
    printf("\n3: To exit");
    printf("\nEnter your choice: ");
    scanf("%d", &ch);
    switch(ch)
    {
      case 1:
          printf("\nEnter the number of elements: ");
          scanf("%d", &n);
```

```
printf("\n Enter array elements : ");
for(i=0; i<n; i++)
{
  scanf("%d", &a[i]);
}
start = clock();
quickSort(a, 0, n-1);
end = clock();
printf("\n Sorted array is : ");
for(i=0; i<n; i++)
{
  printf("%d\t", a[i]);
}
printf("\n time taken to sort %d numbers is %f secs", n,
    ((double)(end - start)) / CLOCKS_PER_SEC));
break;

Case 2:
  n=500;
  while(n<= 14500)
  {
    for(i=0; i<n; i++)
    {
      a[i] = n-i;
    }
    start = clock();
    quickSort(a, 0, n-1);
    for(j=0; j<500000; j++)
    {
      temp = 38/600;
    }
    end = clock();
    quickSort(a, 0, n-1);
    for(j=0; j<50
    printf("\n Time taken to sort %d numbers
        is %f secs", n, ((double)(end-start))/
                        CLOCKS_PER_SEC));
    n = n+1000;
  }
  break;
Case 3:
  exit(0);
}
get char();
}
return 0;
}
```

o/p:-

1. For manual entry of N value & array elements
2. To display time taken for sorting number of elements N in the range 500 to 14500
3. To exit

Enter your choice : 1

Enter the number of elements : 8

Enter array elements : -5

3
1
9
8
2
4
7

Sorted array :- -5 1 2 3 4 7 8 9

Time taken to sort 8 numbers is 0.000000 Secs

Enter your choice: 2

Time taken to sort 500 numbers is 0.000000 Secs.
Time taken to sort 1500 numbers is 0.000000 Secs
Time taken to sort 2500 numbers is 0.016000 Secs
" 3500 numbers is 0.015000 Secs
" 4500 numbers is 0.016000 Secs.
" 5500 numbers is 0.031000 Secs
" 6500 numbers is 0.047000 Secs
" 7500 numbers is 0.062000 Secs
" 8500 numbers is 0.063000 Secs
" 9500 numbers is 0.109000 Secs
" 10500 numbers is 0.094000 Secs
" 11500 numbers is 0.094000 Secs
" 12500 numbers is 0.094000 Secs
" 13500 numbers is 0.172000 Secs
" 14500 numbers is 0.187000 Secs
0.219000 Secs

6/6/24