06/06/24 Week-6

Quick Sort

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
void quick_sort(int a[], int low, int high);
int partition(int a[], int low, int high);
void main() {
     int a[15000], n, i, j, ch, temp;
     clock_t start, end;
     while(1) {
          printf("\n1: For manual Entry of N value and
                    array elements");
          printf("\n2: To display time taken for sorting
                    number of elements N in range 500 to
                         14500");
          printf("\n3: To exit");
          printf("Enter your choice:");
          scanf("%d", &ch);
          switch(ch) {
               case 1:
                    printf("\n Enter the number of elements:");
                    scanf("%d", &n);
                    printf("Enter array elements:");
                    for(i=0; i<n; i++) {
                         scanf("%d", &a[i]);
                    }
                    start=clock();
                    quick_sort(a, 0, n-1);
                    end=clock();
                    printf("\n Sorted array is:");
                    for(i=0; i<n; i++){
```

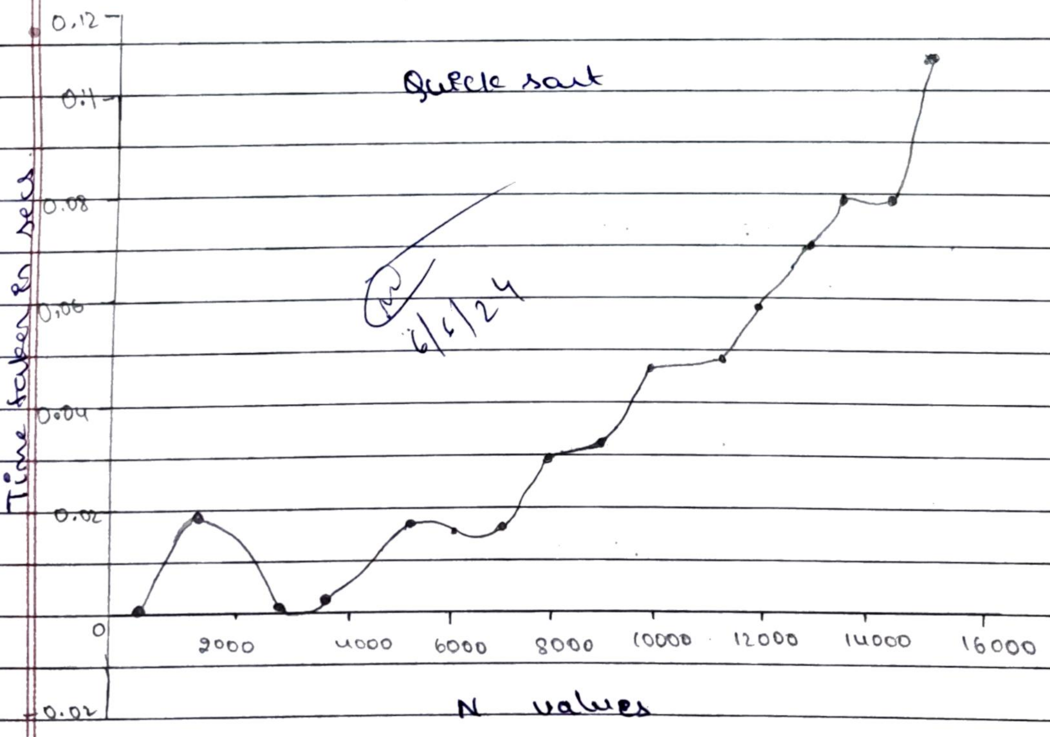
```
            printf("%d\t", a[i]);
          }
      printf("\nTime taken to sort %d numbers is %f secs",
                 n,(((double)(end-start))/clocks_per_sec));
      break;
   case 2:
        n=500;
        while(n<=14500) {
           for(i=0; i<n; i++) {
               a[i]=n-1;
           }
          start=clock();
          quick_sort(a, 0, n-1);
          for(j=0; j<500000; j++) {
              temp= 38/600;
          }
          end=clock();
          printf("\n Time taken to sort %d numbers is %f
                  secs", n,(((double)(end-start))/clocks_perSec))
          n=n+1000;
          }
        break;
      case 3:
          exit(0);
      }
   Getchar();
  }
}
void quick_sort(int a[], int low, int high) {
   if(low<high) {
      int pi=partition(a, low, high);
      quick_sort (a, low, pi-1);
```

```
        quick_sort(a, pi+1, ~~high~~high);
    }
  }
int partition(int a[], int low, int high) {
        int pivot = a[high];
        int i = ((low - 1);
        for(int j = low; j<=high-1; j++) {
            if (a[j] < pivot) {
                i++;
                int temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
        int temp = a[i+1];
        a[i+1] = a[high];
        a[high] = temp;
        return (i+1);
    }
```

output:

```
1. For manual entry of N value and array elements.
2. To display time taken for sorting number of elements N in
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 is: 1, 2, 3, 4, 5, ~~6~~7, 8, 9.
Time taken to sort 8 numbers is 0.00 secs.
1. For manual entry of N value & array elements
2. To display time taken for sorting number of elements N in
range 500 to 14500.
```

```
3: To exit
Enter your choice: 2.
Time taken to sort 500 numbers is 0.00 secs
Time taken to sort 1500 numbers is 0.150 secs
Time taken to sort 2500 numbers is 0.00 secs
Time taken to sort 3500 numbers is 0.00 secs
Time taken to sort 4500 numbers is 0.016 secs
Time taken to sort 5500 numbers is 0.015 secs
Time taken to sort 6500 numbers is 0.016 secs
Time taken to sort 7500 numbers is 0.031 secs
Time taken to sort 8500 numbers is 0.032 secs
Time taken to sort 9500 numbers is 0.046 secs
Time taken to sort 10500 numbers is 0.047 secs
Time taken to sort 11500 numbers is 0.63 secs
Time taken to sort 12500 numbers is 0.078 secs
Time taken to sort 13500 numbers is 0.078 secs
Time taken to sort 14500 numbers is 0.1090 secs.
```



6/6/24