

Quick sort

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

```
void quick_sort(int arr[], int low, int high);
int partition(int arr[], int low, int high);
void main()
```

{

int arr[5000], n, i, j, ch, temp;

clock_t start, end;

while(1){

 printf("In 1. For manual entry of N value and
 array elements"); printf("In 2. To display time taken for sorting
 number of elements N in the range of 500 to 14500");

printf("In 3. To exit");

scanf("In Enter your choice");

scanf("%d", &ch);

switch(ch){

case:

printf("Enter the number of elements");

scanf("%d", &n);

printf("Enter array elements");

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

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

}

start = clock();

quick_sort(arr, 0, n-1);

end = clock();

printf("In sorted array is:");

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

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

}

 printf("In Time taken to sort %d numbers is %.f
secs", n, ((double)(end - start))/CLOCKS_PER_SEC);

break;

calc3:

n = 500;

```
while (n <= 14500) {  
    for (i = 0; i < n; i++) {  
        if (i == n - 1)  
            break;  
    }
```

start = clock();

```
quickSort(a, 0, n - 1);  
for (j = 0; j < 5000000; j++) {  
    sleep(381600);  
}
```

end = clock();

fixing ("Implementation of sort of numbers is
of class", n, ((double)(end - start)) / CLOCKS_PER_SEC)

n = n + 1000;

}

break;

calc3: calc3();

j

getchar();

}

void quickSort(int arr[], int low, int high) {

if (low > high)

int pi = partition(a, low, high);

quickSort(a, low, pi - 1);

quickSort(a, pi + 1, high);

}

int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = (low - 1);

```

for (int i = 0; i < high - 1; i++) {
    if (a[i] > a[i + 1]) {
        int temp = a[i];
        a[i] = a[i + 1];
        a[i + 1] = temp;
    }
}
    
```

Output:

- For manual entry of N value and array elements
- To display time taken for sorting number of elements N in the range of 500 to 14500
- To exit

Enter your choice: 1

Enter the number of elements: 5

Enter array elements: 14 17 10 13 12

Sorted array is: 10 12 13 14 17

Time taken to sort 5 numbers is 0.0000 secs

- For manual entry of N value and array elements
- To display time taken for sorting number of elements N in the range of 500 to 14500
- To exit

Enter your choice: 2

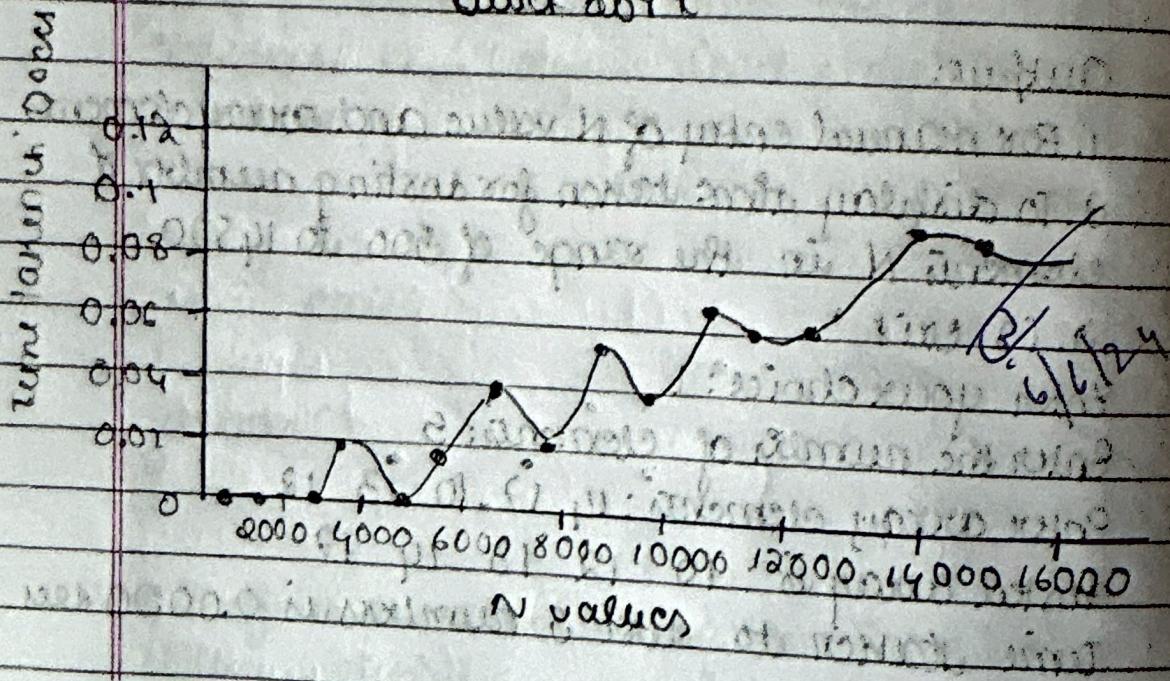
Time taken to sort 500 numbers is 0 secs

Time taken to sort 1500 numbers is 0 secs

Time taken to sort 2500 numbers is 0 secs

Time taken to sort 3600 numbers is 0.016 sec
Time taken to sort 4800 numbers is 0.020 sec
Time taken to sort 5800 numbers is 0.018 sec
Time taken to sort 6800 numbers is 0.032 sec
Time taken to sort 7800 numbers is 0.015 sec
Time taken to sort 8800 numbers is 0.043 sec
Time taken to sort 9800 numbers is 0.031 sec
Time taken to sort 10800 numbers is 0.053 sec
Time taken to sort 11800 numbers is 0.062 sec
Time taken to sort 12800 numbers is 0.063 sec
Time taken to sort 13800 numbers is 0.094 sec
Time taken to sort 14800 numbers is 0.093 sec

Quick sort



Quicksort is a divide and conquer algorithm. It has two main phases: partitioning and recursive sorting. In the partition phase, it selects a pivot element and rearranges the array such that all elements less than or equal to the pivot are on the left, and all elements greater than the pivot are on the right. This step is repeated for each subarray until the entire array is sorted. The time complexity of Quicksort is $O(N \log N)$ on average, but it can degenerate to $O(N^2)$ in the worst case if the pivot selection is poor.