





Code (Merge Sort)

#include <iostream>

#include <bits/stdc++.h>

#include <chrono>

using namespace std;

using namespace std::chrono;

void merge(int arr[], int p, int q, int r)

{

    int n1, n2, i, j, k;

    n1 = q - p + 1;

    n2 = r - q;

    int left[n1 + 1], right[n2 + 1];

    for (i = 1; i < n1 + 1; i++)

    {

        left[i] = arr[p + i - 1];

    }

    for (i = 1; i < n2 + 1; i++)

    {

        right[i] = arr[q + i];

    }

    left[n1 + 1] = 1000000;

    right[n2 + 1] = 100000;

    i = 1;

    j = 1;

    for (k = p; k < r + 1; k++)

    {

        if (left[i] <= right[j])

        {

            arr[k] = left[i];

            i += 1;

        }

        else

        {

            arr[k] = right[j];

            j += 1;

        }

    }

}

void merge\_sort(int arr[], int p, int r)

{

    int q;

    if (p < r)

    {

        q = (p + r) / 2;

        merge\_sort(arr, p, q);

        merge\_sort(arr, q + 1, r);

        merge(arr, p, q, r);

    }

}

int main()

{

    int n, p, r, i, j;

    cout << "Enter the Number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter the elements: ";

    for (i = 0; i < n; i++)

    {

        cin >> arr[i];

    }

    auto start = steady\_clock::now();

    merge\_sort(arr, 0, n - 1);

    auto stop = steady\_clock::now();

    cout << "Sorted array: ";

    for (i = 0; i < n; i++)

    {

        cout << arr[i] << " ";

    }

    auto duration = duration\_cast<nanoseconds>(stop - start);

    cout << "\nTime taken by function: " << duration.count() << " nanoseconds" << endl;

}

Output (Merge Sort)

Best Case



Worst Case



Code (Quick Sort)

#include <iostream>

#include <bits/stdc++.h>

#include <chrono>

using namespace std;

using namespace std::chrono;

int partition(int arr[], int p, int r)

{

    int temp;

    int x = arr[r];

    int i = p - 1;

    for (int j = p; j < r; j++)

    {

        if (arr[j] <= x)

        {

            i = i + 1;

            temp = arr[i];

            arr[i] = arr[j];

            arr[j] = temp;

        }

    }

    temp = arr[i + 1];

    arr[i + 1] = arr[r];

    arr[r] = temp;

    return i + 1;

}

void quicksort(int arr[], int p, int r)

{

    if (p < r)

    {

        int q = partition(arr, p, r);

        quicksort(arr, p, q - 1);

        quicksort(arr, q + 1, r);

    }

}

int main()

{

    int n, i, j;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter the elements of array: ";

    for (i = 0; i < n; i++)

    {

        cin >> arr[i];

    }

    auto start = steady\_clock::now();

    quicksort(arr, 0, n - 1);

    auto stop = steady\_clock::now();

    cout << "Sorted Array is: ";

    for (i = 0; i < n; i++)

    {

        cout << arr[i] << " ";

    }

    auto duration = duration\_cast<nanoseconds>(stop - start);

    cout << "\nTime taken by function: " << duration.count() << " nanoseconds" << endl;

}

Output (Quick Sort)

Best Case



Worst Case

