**INSERTION SORT**

#include <iostream>

using namespace std;

void insertionsort(int arr[], int n)

{

int count = 0;

for (int i = 1; i < n; i++)

{

count++;

int j = 0;

count++;

int key = arr[i];

count++;

for (j = i- 1; j >= 0 && arr[j] > key; j--)

{

count++;

arr[j + 1] = arr[j];

count++;

}

count++;

arr[j + 1] = key;

count++;

}

count++;

cout << "display the time complexity:" << count << endl;

}

int main()

{

int n, arr[] = {5,4,3,2,1};

n =sizeof(arr) / sizeof(arr[0]);

insertionsort(arr, n);

cout << "DISPLAY SORTED ARRAY:" << endl;

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

{

cout << arr[i] << " ";

}

}

**SELECTION SORT**

#include <iostream>

using namespace std;

void swap(int &a, int &b)

{

int temp = a;

a =b;

b =temp;

}

void selectionsort(int arr[], int n)

{

int count = 0, mid\_index;

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

{

count++;

mid\_index = i;

count++;

for (int j = i + 1; j < n; j++)

{

count++;

if (arr[j] < arr[mid\_index])

{

count++;

mid\_index = j;

count++;

}

count++;

}

count++;

swap(arr[mid\_index], arr[i]);

count++;

}

count++;

cout << "display the time complexity:" << count << endl;

}

int main()

{

int n, arr[] = {1,2,3,5,4};

n =sizeof(arr) / sizeof(arr[0]);

selectionsort(arr, n);

cout << "DISPLAY SORTED ARRAY:" << endl;

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

{

cout << arr[i] << " ";

}

}

**BUBBLE SORT**

#include <iostream>

using namespace std;

void swap(int &a, int &b)

{

int temp = a;

a =b;

b =temp;

}

void bubblesort(int arr[], int n)

{

int count = 0;

for (int i = 0; i < n- 1; i++)

{

count++;

for (int j = 0; j < n- i- 1; j++)

{

count++;

if (arr[j + 1] < arr[j])

{

count++;

swap(arr[j + 1], arr[j]);

count++;

}

count++;

}

count++;

}

count++;

cout << "display the time complexity:" << count << endl;

}

int main()

{

int n, arr[] = {1,2,3,4,5};

n =sizeof(arr) / sizeof(arr[0]);

bubblesort(arr, n);

cout << "DISPLAY SORTED ARRAY:" << endl;

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

{

cout << arr[i] << " ";

}

}

**QUICK SORT**

#include <iostream>

using namespace std;

void swap(int &a, int &b)

{

int temp = a;

a =b;

b =temp;

}

int partition(int arr[], int low, int high, int &count)

{

int pivot = arr[low];

int i = (low + 1);

int j = high;

do

{

count++;

while (arr[i] <= pivot)

{

i++;

count++;

}

while (arr[j] > pivot)

{

j--;

count++;

}

if (i < j)

{

count++;

swap(arr[i], arr[j]);

count++;

}

} while (i < j);

swap(arr[low], arr[j]);

count++;

return j;

}

int quickSort(int arr[], int low, int high, int &count)

{

count++;

if (low < high)

{

count++;

int pivotIndex = partition(arr, low, high, count);

count++;

quickSort(arr, low, pivotIndex- 1, count);

count++;

quickSort(arr, pivotIndex + 1, high, count);

count++;

}

return count;

}

int main()

{

int arr[] = {1,4,5,2,3};

int count = 0;

int n = sizeof(arr) / sizeof(arr[0]);

quickSort(arr, 0, n- 1, count);

cout << "Display time complexity:" << count << endl;

cout<<"DISPLAY SORTED ARRAY"<<endl;

cout << "Sorted Array: ";

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

{

cout << arr[i] << " ";

}

}

**MERGE SORT**

#include <iostream>

using namespace std;

void merge(int arr[], int mid, int low, int high, int &count)

{

int i, j, k;

int B[high- low + 1];

i = low;

j = mid + 1;

k = 0;

while (i <= mid && j <= high)

{

count++;

if (arr[i] < arr[j])

{

count++;

B[k] = arr[i];

count++;

i++;

count++;

k++;

count++;

}

else

{

count++;

B[k] = arr[j];

count++;

j++;

count++;

k++;

count++;

}

}

count++;

while (i <= mid)

{

count++;

B[k] = arr[i];

count++;

i++;

count++;

k++;

count++;

}

count++;

while (j <= high)

{

count++;

B[k] = arr[j];

count++;

j++;

count++;

k++;

count++;

}

count++;

for (i = low, k = 0; i <= high; i++)

{

count++;

arr[i] = B[k];

count++;

k++;

count++;

}

count++;

}

void mergesort(int arr[], int low, int high, int &count)

{

count++;

if (low < high)

{

count++;

int mid = (low + high) / 2;

mergesort(arr, low, mid, count);

mergesort(arr, mid + 1, high, count);

merge(arr, mid, low, high, count);

}

}

int main()

{

int arr[] = {5,4,1,2,3};

int n = sizeof(arr) / sizeof(arr[0]);

int count = 0;

mergesort(arr, 0, n- 1, count);

cout << "Display time complexity:" << count << endl;

cout<<"Display Sorted Array: "<<endl;

cout << "Sorted Array: ";

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

{

cout << arr[i] << " ";

}

cout << endl;

return 0;

}