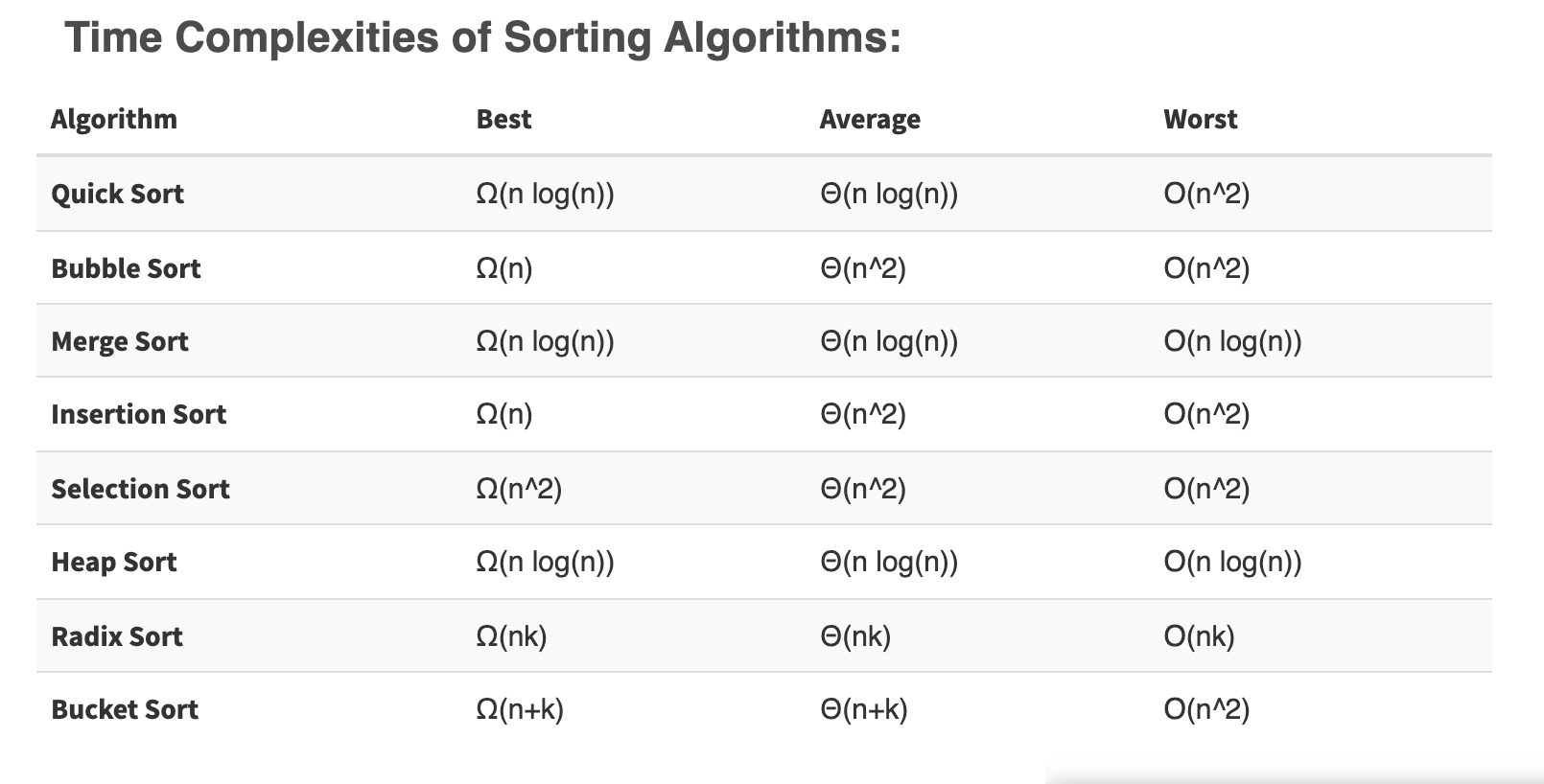
**Important Sorting Algorithms at one place.**



Quicksort example program in c++:

#include<iostream>

#include<cstdlib>

using namespace std;

// Swapping two values.

void swap(int \*a, int \*b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

// Partitioning the array on the basis of values at high as pivot value.

int Partition(int a[], int low, int high)

{

int pivot, index, i;

index = low;

pivot = high;

// Getting index of the pivot.

for(i=low; i < high; i++)

{

if(a[i] < a[pivot])

{

swap(&a[i], &a[index]);

index++;

}

}

// Swapping value at high and at the index obtained.

swap(&a[pivot], &a[index]);

return index;

}

// Random selection of pivot.

int RandomPivotPartition(int a[], int low, int high)

{

int pvt, n, temp;

n = rand();

// Randomizing the pivot value in the given subpart of array.

pvt = low + n%(high-low+1);

// Swapping pivot value from high, so pivot value will be taken as a pivot while partitioning.

swap(&a[high], &a[pvt]);

return Partition(a, low, high);

}

int QuickSort(int a[], int low, int high)

{

int pindex;

if(low < high)

{

// Partitioning array using randomized pivot.

pindex = RandomPivotPartition(a, low, high);

// Recursively implementing QuickSort.

QuickSort(a, low, pindex-1);

QuickSort(a, pindex+1, high);

}

return 0;

}

int main()

{

int n, i;

cout<<"\nEnter the number of data elements to be sorted: ";

cin>>n;

int arr[n];

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

{

cout<<"Enter element "<<i+1<<": ";

cin>>arr[i];

}

QuickSort(arr, 0, n-1);

// Printing the sorted data.

cout<<"\nSorted Data ";

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

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

return 0;

}

Input: 50, 23, 9, 18, 61, 32

Output: 9, 18, 23, 32, 50, 61

Implementation of bubble sort in C

*#include<stdio.h>*

**int** main**()**

**{**

**int** arr**[]** **=** **{5,** **6,** **2** **,6,** **9,** **0,** **-1},** n **=** **7,** i**,** j**;**

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

**{**

**int** swapped **=** **0;**

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

**if(**arr**[**j**]** **>** arr**[**j**+1])**

**{**

**int** temp **=** arr**[**j**];**

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

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

swapped **=** **1;**

**}**

**if(!**swapped**)**

**break;**

**}**

printf**(**"\nArray after sorting: "**);**

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

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

**return** **0;**

**}**

Input: 50, 23, 9, 18, 61, 32

Output: 9, 18, 23, 32, 50, 61

*// example of merge sort in C/C++*

*// merge function take two intervals*

*// one from start to mid*

*// second from mid+1, to end*

*// and merge them in sorted order*

**void** **merge**(**int** **\***Arr, **int** start, **int** mid, **int** end) {

*// create a temp array*

**int** temp[end **-** start **+** 1];

*// crawlers for both intervals and for temp*

**int** i **=** start, j **=** mid**+**1, k **=** 0;

*// traverse both arrays and in each iteration add smaller of both elements in temp*

**while**(i **<=** mid **&&** j **<=** end) {

**if**(Arr[i] **<=** Arr[j]) {

temp[k] **=** Arr[i];

k **+=** 1; i **+=** 1;

}

**else** {

temp[k] **=** Arr[j];

k **+=** 1; j **+=** 1;

}

}

*// add elements left in the first interval*

**while**(i **<=** mid) {

temp[k] **=** Arr[i];

k **+=** 1; i **+=** 1;

}

*// add elements left in the second interval*

**while**(j **<=** end) {

temp[k] **=** Arr[j];

k **+=** 1; j **+=** 1;

}

*// copy temp to original interval*

**for**(i **=** start; i **<=** end; i **+=** 1) {

Arr[i] **=** temp[i **-** start]

}

}

*// Arr is an array of integer type*

*// start and end are the starting and ending index of current interval of Arr*

**void** **mergeSort**(**int** **\***Arr, **int** start, **int** end) {

**if**(start **<** end) {

**int** mid **=** (start **+** end) **/** 2;

mergeSort(Arr, start, mid);

mergeSort(Arr, mid**+**1, end);

merge(Arr, start, mid, end);

}

}

Input: 50, 23, 9, 18, 61, 32

Output: 9, 18, 23, 32, 50, 61

Insertion sort Implementation in C:

#include <stdio.h>

#include <stdbool.h>

#define MAX 7 //defining size of our array

int intArray[MAX] = {4,6,3,2,1,9,7};

void printline(int count) {

int i;

for(i = 0;i < count-1;i++) {

printf("=");

}

printf("=\n");

}

void display() {

int i;

printf("[");

// navigate through all items

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

printf("%d ",intArray[i]);

}

printf("]\n");

}

void insertionSort() {

int valueToInsert;

int holePosition;

int i;

// loop through all numbers

for(i = 1; i < MAX; i++) {

// select a value to be inserted.

valueToInsert = intArray[i];

// select the hole position where number is to be inserted

holePosition = i;

// check if previous no. is larger than value to be inserted

while (holePosition > 0 && intArray[holePosition-1] > valueToInsert) {

intArray[holePosition] = intArray[holePosition-1];

holePosition--;

printf(" item moved : %d\n" , intArray[holePosition]);

}

if(holePosition != i) {

printf(" item inserted : %d, at position : %d\n" , valueToInsert,holePosition);

// insert the number at current hole

intArray[holePosition] = valueToInsert;

}

printf("Iteration %d#:",i);

display();

}

}

void main() {

printf("Input Array: ");

display();

printline(50);

insertionSort();

printf("Output Array: ");

display();

printline(50);

}

Input: 50, 23, 9, 18, 61, 32

Output: 9, 18, 23, 32, 50, 61

Selection sort program in C++:

#include <iostream>

#include <vector>

using namespace std;

int findMinIndex(vector<int> &A, int start) {

int min\_index = start;

++start;

while(start < (int)A.size()) {

if(A[start] < A[min\_index])

min\_index = start;

++start;

}

return min\_index;

}

void selectionSort(vector<int> &A) {

for(int i = 0; i < (int)A.size(); ++i) {

int min\_index = findMinIndex(A, i);

if(i != min\_index)

swap(A[i], A[min\_index]);

}

}

int main() {

vector<int> A = {5, 2, 6, 7, 2, 1, 0, 3};

selectionSort(A);

for(int num : A)

cout << num << ' ';

return 0;

}

Input: 50, 23, 9, 18, 61, 32

Output: 9, 18, 23, 32, 50, 61