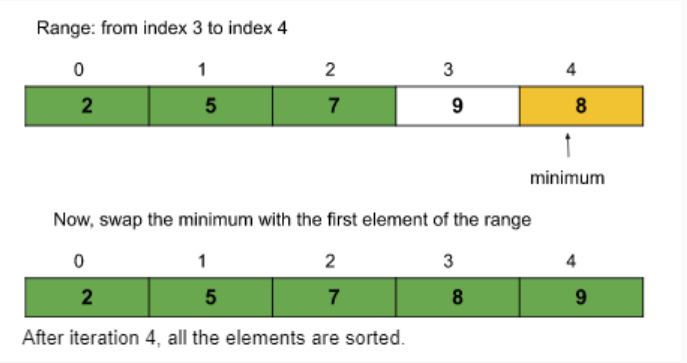
SORTING

1. Selection sort

#include<bits/stdc++.h>

using namespace std;

void selectionsort(int arr[],int n){

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

int mini=i;

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

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

mini=j;

}

int temp=arr[mini];

arr[mini]=arr[i];

arr[i]=temp;

}

cout<<"\nAfter selection sort : ";

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

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

}

}

int main(){

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

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

cout<<"Before selection sort : ";

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

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

}

selectionsort(arr,n);

return 0;

}

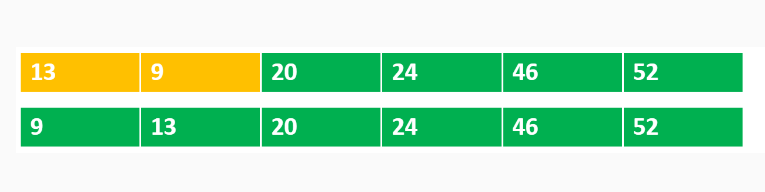
Output

Before selection sort : 1 4 3 2 5

After selection sort : 1 2 3 4 5

1. Bubble sort

#include<bits/stdc++.h>

using namespace std;

void bubblesort(int arr[],int n){

for(int i=n-1;i>=0;i--){

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

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

int temp=arr[j+1];

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

arr[j]=temp;

}

}

}

cout<<"\nAfter sorting : ";

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

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

}

}

int main(){

int arr[]={1,2,4,3,6,1};

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

cout<<"Before sorting : ";

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

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

}

bubblesort(arr,n);

return 0;

}

Output

Before sorting : 1 2 4 3 6 1

After sorting : 1 1 2 3 4 6

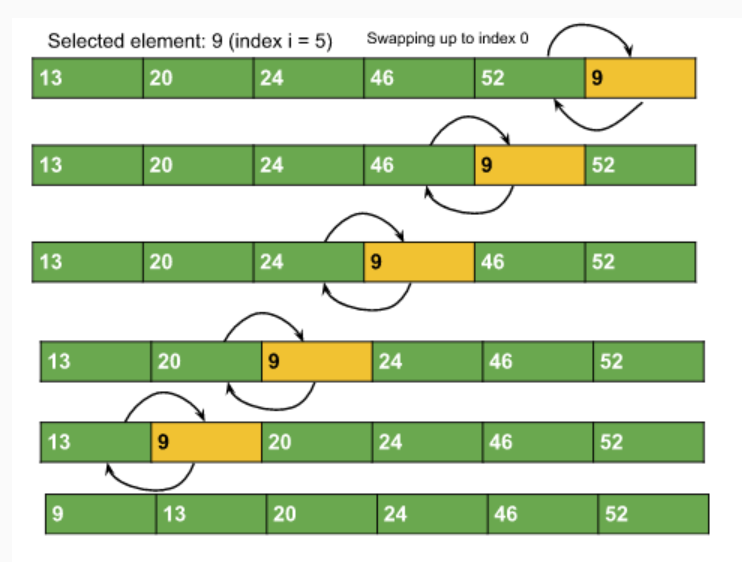
1. Insertion sort

#include<bits/stdc++.h>

using namespace std;

void insertionsort(int arr[],int n){

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

 int j=i;

while(j>0 && arr[j-1]>arr[j]){

int temp=arr[j-1];

arr[j-1]=arr[j];

arr[j]=temp;

j--;

}

}

cout<<"\nAfter sorting : ";

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

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

}

}

int main(){

int arr[] = {10,3,2,4};

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

cout<<"Before sorting : ";

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

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

}

insertionsort(arr,n);

return 0;

}

Output

Before sorting : 10 3 2 4

After sorting : 2 3 4 10

1. Merge sort

#include<bits/stdc++.h>

using namespace std;

void merge(vector<int>&arr,int low,int mid,int high){

vector<int>temp;

int left = low;

int right = mid+1;

while(left<=mid && right<=high){

if(arr[left]<=arr[right]){

temp.push\_back(arr[left]);

left++;

}

else{

temp.push\_back(arr[right]);

right++;

}

}

while(left<=mid){

temp.push\_back(arr[left]);

left++;

}

while(right<=high){

temp.push\_back(arr[right]);

right++;

}

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

arr[i]=temp[i-low];

}

}

void mergesort(vector<int>&arr,int low,int high){

if(low>=high)return;

int mid = (low+high)/2;

mergesort(arr,low,mid);

mergesort(arr,mid+1,high);

merge(arr,low,mid,high);

}

int main(){

vector<int>arr={3,5,4,2,1};

int n=5;

cout<<"Before sorting : ";

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

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

}

mergesort(arr,0,n-1);

cout<<"\nAfter sorting : ";

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

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

}

return 0;

}

Output

Before sorting : 3 5 4 2 1

After sorting : 1 2 3 4 5

1. Quick sort

#include<bits/stdc++.h>

using namespace std;

int partition(vector<int>&arr,int low,int high){

int pivot=arr[low];

int i=low;

int j=high;

while(i<j){

while(arr[i]<=pivot && i<=high-1){

i++;

}

while(arr[j]> pivot && j>=low+1){

j--;

}

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

}

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

return j;

}

void qs(vector<int>&arr,int low,int high){

if(low<high){

int pIndex=partition(arr,low,high);

qs(arr,low,pIndex-1);

qs(arr,pIndex+1,high);

}

}

vector<int>quicksort(vector<int>arr){

qs(arr,0,arr.size()-1);

return arr;

}

int main(){

vector<int>arr = {2,3,4,5,2,12};

int n = arr.size();

cout<<"Before sorting : ";

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

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

}

arr = quicksort(arr);

cout<<"\nAfter sorting : ";

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

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

}

return 0;

}

Output

Before sorting : 2 3 4 5 2 12

After sorting : 2 2 3 4 5 12