Exp1. implement insertion sort

#include <math.h>

#include <stdio.h>

void insertionSort(int arr[], int n)

{

int i, key, j;

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

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] > key) {

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

j = j - 1;

}

arr[j + 1] = key;

}

}

{

int i;

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

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

printf("\n");

}

int main()

{

int arr[] = { 23, 69, 52, 74, 21 };

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

insertionSort(arr, n);

printArray(arr, n);

return 0;

}

Output:-

21

23

52

69

74

Exp2: selection sort

#include <stdio.h>

void swap(int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void selectionSort(int arr[], int n)

{

int i, j, min\_idx;

// One by one move boundary of unsorted subarray

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

{

// Find the minimum element in unsorted array

min\_idx = i;

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

if (arr[j] < arr[min\_idx])

min\_idx = j;

// Swap the found minimum element with the first element

if(min\_idx != i)

swap(&arr[min\_idx], &arr[i]);

}

}

/\* Function to print an array \*/

void printArray(int arr[], int size)

{

int i;

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

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

printf("\n");

}

// Driver program to test above functions

int main()

{

int arr[] = {34,23,45,16,13};

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

selectionSort(arr, n);

printf("Sorted array: \n");

printArray(arr, n);

return 0;

}

Output:



Exp4: Binary Search in C

#include <stdio.h>

int binarySearch(int array[], int x, int low, int high) {

// Repeat until the pointers low and high meet each other

while (low <= high) {

int mid = low + (high - low) / 2;

if (array[mid] == x)

return mid;

if (array[mid] < x)

low = mid + 1;

else

high = mid - 1;

}

return -1;

}

int main(void) {

int array[] = {3, 4, 5, 6, 7, 8, 9};

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

int x = 4;

int result = binarySearch(array, x, 0, n - 1);

if (result == -1)

printf("Not found");

else

printf("Element is found at index %d", result);

return 0;

}



Exp 5:quick sort

#include <stdio.h>

// Function to swap two elements

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

int temp = \*a;

\*a = \*b;

\*b = temp;

}

// Function to partition the array and return the pivot index

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

int pivot = arr[high]; // Choose the last element as the pivot

int i = (low - 1); // Index of smaller element

int j;

for (j = low; j <= high - 1; j++) {

// If current element is smaller than or equal to pivot

if (arr[j] <= pivot) {

i++; // Increment index of smaller element

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

}

}

swap(&arr[i + 1], &arr[high]);

return (i + 1); // Return the partition index

}

// Function to perform Quick Sort

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

if (low < high) {

// Get the partition index

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

// Recursively sort the left and right subarrays

quickSort(arr, low, partitionIndex - 1);

quickSort(arr, partitionIndex + 1, high);

}

}

// Driver code

int main() {

int i;

int arr[] = {12, 11, 13, 5, 6, 7};

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

printf("Original array: ");

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

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

printf("\n");

quickSort(arr, 0, n - 1);

printf("Sorted array: ");

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

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

printf("\n");

return 0;

}