INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. | NAME OF PROGRAME | DATE | SIGN. |
| 1. | Linear Search |  |  |
| 2. | Binary Search |  |  |
| 3. | Bubble Sort |  |  |
| 14. | Insertion Sort |  |  |
| 5. | Heap Sort |  |  |
| 6. | Quick Sort |  |  |
| 7. | Merge Sort |  |  |
| 8. | Dictionary |  |  |

WAP FOR LINEAR SEARCH IN PYTHON

def search(arr, x):

    for i in range(len(arr)):

        if arr[i] == x:

            return i

    return -1

**Input:**

arr[] = {10, 20, 80, 30, 60, 50,110, 100, 130, 170}

x = 110;

**Output :** 6

Element x is present at index 6

WAP FOR BINARY SEARCH IN PYTHON

def binarySearch(arr, l, r, x):

    while l <= r:

        mid = l + (r - l)/2;

        if arr[mid] == x:

            return mid

        elif arr[mid] < x:

            l = mid + 1

        else:

            r = mid - 1

    return -1

**Input:**

arr[] = [2,3,4,10,40]

x = 10;

**Output :** 6

Element x is present at index 3

WAP FOR BUBBLE SORT IN PYTHON

def bubbleSort(arr):

    n = len(arr)

    # Traverse through all array elements

    for i in range(n):

        # Last i elements are already in place

        for j in range(0, n-i-1):

            # traverse the array from 0 to n-i-1

            # Swap if the element found is greater

            # than the next element

            if arr[j] > arr[j+1] :

                arr[j], arr[j+1] = arr[j+1], arr[j]

**Input:**

arr[] = [64,34,12,22,11,90]

**Output :**

Sorted array is: [11,12,22,34,64,90]

WAP FOR INSERTION SORT IN PYTHON

def insertionSort(arr):

    # Traverse through 1 to len(arr)

    for i in range(1, len(arr)):

        key = arr[i]

        # Move elements of arr[0..i-1], that are

        # greater than key, to one position ahead

        # of their current position

        j = i-1

        while j >=0 and key < arr[j] :

                arr[j+1] = arr[j]

                j -= 1

        arr[j+1] = key

insertionSort(arr)

print ("Sorted array is:")

for i in range(len(arr)):

    print ("%d" %arr[i])

**Input:**

arr[] = [12,11,13,5,6]

**Output :**

Sorted array is: [5,6,11,12,13]

WAP FOR HEAP SORT IN PYTHON

def heapify(arr, n, i):

    largest = i

    l = 2 \* i + 1

    r = 2 \* i + 2

if l < n and arr[i] < arr[l]:

        largest = l

if r < n and arr[largest] < arr[r]:

        largest = r

 if largest != i:

        arr[i],arr[largest] = arr[largest],arr[i]

heapify(arr, n, largest)

# The main function to sort an array of given size

def heapSort(arr):

    n = len(arr)

    # Build a maxheap.

    for i in range(n, -1, -1):

        heapify(arr, n, i)

    # One by one extract elements

    for i in range(n-1, 0, -1):

        arr[i], arr[0] = arr[0], arr[i] # swap

        heapify(arr, i, 0)

# Driver code to test above

arr = [ 12, 11, 13, 5, 6, 7]

heapSort(arr)

n = len(arr)

print ("Sorted array is")

for i in range(n):

    print ("%d" %arr[i])

**Input:**

arr[] = [12,11,13,5,6]

**Output :**

Sorted array is: [5,6,11,12,13]

WAP FOR QUICK SORT IN PYTHON

# Function to do Quick sort

def quickSort(arr,low,high):

    if low < high:

        # pi is partitioning index, arr[p] is now

        # at right place

        pi = partition(arr,low,high)

        # Separately sort elements before

        # partition and after partition

        quickSort(arr, low, pi-1)

        quickSort(arr, pi+1, high)

# Function to do partition

def partition(arr,low,high):

    i = ( low-1 )         # index of smaller element

    pivot = arr[high]     # pivot

    for j in range(low , high):

        # If current element is smaller than or

        # equal to pivot

        if   arr[j] <= pivot:

            # increment index of smaller element

            i = i+1

            arr[i],arr[j] = arr[j],arr[i]

    arr[i+1],arr[high] = arr[high],arr[i+1]

    return ( i+1 )

# Driver code to test above

arr = [10, 7, 8, 9, 1, 5]

n = len(arr)

quickSort(arr,0,n-1)

print ("Sorted array is:")

for i in range(n):

    print ("%d" %arr[i])

**Input:**

arr[] = [10,7,8,9,1,5]

**Output :**

Sorted array is: [1,5,7,8,9,10]

WAP FOR MERGE SORT IN PYTHON

def merge(arr, l, m, r):

    n1 = m - l + 1

    n2 = r- m

    # create temp arrays

    L = [0] \* (n1)

    R = [0] \* (n2)

    # Copy data to temp arrays L[] and R[]

    for i in range(0 , n1):

        L[i] = arr[l + i]

    for j in range(0 , n2):

        R[j] = arr[m + 1 + j]

    # Merge the temp arrays back into arr[l..r]

    i = 0     # Initial index of first subarray

    j = 0     # Initial index of second subarray

    k = l     # Initial index of merged subarray

    while i < n1 and j < n2 :

        if L[i] <= R[j]:

            arr[k] = L[i]

            i += 1

        else:

            arr[k] = R[j]

            j += 1

        k += 1

    # Copy the remaining elements of L[], if there

    # are any

    while i < n1:

        arr[k] = L[i]

        i += 1

        k += 1

    # Copy the remaining elements of R[], if there

    # are any

    while j < n2:

        arr[k] = R[j]

        j += 1

        k += 1

# l is for left index and r is right index of the

# sub-array of arr to be sorted

def mergeSort(arr,l,r):

    if l < r:

        # Same as (l+r)/2, but avoids overflow for

        # large l and h

        m = (l+(r-1))/2

        # Sort first and second halves

        mergeSort(arr, l, m)

        mergeSort(arr, m+1, r)

        merge(arr, l, m, r)

# Driver code to test above

arr = [12, 11, 13, 5, 6, 7]

n = len(arr)

print ("Given array is")

for i in range(n):

    print ("%d" %arr[i]),

mergeSort(arr,0,n-1)

print ("\n\nSorted array is")

for i in range(n):

    print ("%d" %arr[i])

Input:

arr[]=12 11 13 5 6 7

Output:

Sorted array is [5 6 7 11 12 13]

WAP FOR DICTIONARY IN PYTHON

# Creating an empty Dictionary

Dict = {}

print("Empty Dictionary: ")

print(Dict)

# Creating a Dictionary

# with Integer Keys

Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}

print("\nDictionary with the use of Integer Keys: ")

print(Dict)

# Creating a Dictionary

# with Mixed keys

Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]}

print("\nDictionary with the use of Mixed Keys: ")

print(Dict)

# Creating a Dictionary

# with dict() method

Dict = dict({1: 'Geeks', 2: 'For', 3:'Geeks'})

print("\nDictionary with the use of dict(): ")

print(Dict)

# Creating a Dictionary

# with each item as a Pair

Dict = dict([(1, 'Geeks'), (2, 'For')])

print("\nDictionary with each item as a pair: ")

print(Dict)

Output:

Empty Dictionary:

{}

Dictionary with the use of Integer Keys:

{1: 'Geeks', 2: 'For', 3: 'Geeks'}

Dictionary with the use of Mixed Keys:

{1: [1, 2, 3, 4], 'Name': 'Geeks'}

Dictionary with the use of dict():

{1: 'Geeks', 2: 'For', 3: 'Geeks'}

Dictionary with each item as a pair:

{1: 'Geeks', 2: 'For'}