1. Linear Search [Solution]

class Linked {

public static int search(int arr[], int N, int x)

{

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

if (arr[i] == x)

return i;

}

return -1;

}

public static void main(String args[])

{

int arr[] = { 2, 3, 4, 10, 40 };

int x = 10;

int result = search(arr, arr.length, x);

if (result == -1)

System.out.print(

"Element is not present in array");

else

System.out.print("Element is present at index "

+ result);

}

} class Linked {

public static int search(int arr[], int N, int x)

{

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

if (arr[i] == x)

return i;

}

return -1;

}

public static void main(String args[])

{

int arr[] = { 2, 3, 4, 10, 40 };

int x = 10;

int result = search(arr, arr.length, x);

if (result == -1)

System.out.print(

"Element is not present in array");

else

System.out.print("Element is present at index "

+ result);

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

Element is present at index 3

2. Binary Search [Solution]

class Linked {

int Linked(int arr[], int x)

{

int l = 0, r = arr.length - 1;

while (l <= r) {

int m = l + (r - l) / 2;

if (arr[m] == x)

return m;

if (arr[m] < x)

l = m + 1;

else

r = m - 1;

}

return -1;

}

public static void main(String args[])

{

Linked ob = new Linked();

int arr[] = { 2, 3, 4, 10, 40 };

int n = arr.length;

int x = 10;

int result = ob.Linked(arr, x);

if (result == -1)

System.out.println(

"Element is not present in array");

else

System.out.println("Element is present at "

+ "index " + result);

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

Element is present at index 3

3. Sort elements by frequency

import java.util.\*;

class ele {

int count, index, val;

}

class mycomp implements Comparator<ele> {

public int compare(ele a, ele b)

{

return (a.val - b.val);

}

}

class mycomp2 implements Comparator<ele> {

public int compare(ele a, ele b)

{

if (a.count != b.count)

return (a.count - b.count);

return (b.index - a.index);

}

}

class Linked {

static void sortByFrequency(int[] arr, int n)

{

ArrayList<ele> element = new ArrayList<ele>();

// ele[] element = new ele[n];

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

element.add(new ele());

element.get(i).index = i;

element.get(i).count = 0;

element.get(i).val = arr[i];

}

Collections.sort(element, new mycomp());

element.get(0).count = 1;

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

if (element.get(i).val == element.get(i - 1).val) {

element.get(i).count += element.get(i - 1).count + 1;

element.get(i - 1).count = -1;

element.get(i).index = element.get(i - 1).index;

}

else

element.get(i).count = 1;

}

Collections.sort(element, new mycomp2());

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

if (element.get(i).count != -1)

{

for (int j = 0; j < element.get(i).count;j++)

arr[index++] = element.get(i).val;

}

}

}

public static void main(String[] args)

{

int[] arr = { 2, 5, 2, 6, -1, 9999999, 5, 8, 8, 8 };

int n = arr.length;

sortByFrequency(arr, n);

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

System.out.print(arr[i] + " ");

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

8 8 8 2 2 5 5 6 -1 9999999

4. Sort an array of 0s, 1s and 2s

class Linked

{

static void sort012(int a[], int arr\_size)

{

int lo = 0;

int hi = arr\_size - 1;

int mid = 0, temp = 0;

while (mid <= hi) {

switch (a[mid]) {

case 0: {

temp = a[lo];

a[lo] = a[mid];

a[mid] = temp;

lo++;

mid++;

break;

}

case 1:

mid++;

break;

case 2: {

temp = a[mid];

a[mid] = a[hi];

a[hi] = temp;

hi--;

break;

}

}

}

}

static void printArray(int arr[], int arr\_size)

{

int i;

for (i = 0; i < arr\_size; i++)

System.out.print(arr[i] + " ");

System.out.println("");

}

public static void main(String[] args)

{

int arr[] = { 0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1 };

int arr\_size = arr.length;

sort012(arr, arr\_size);

printArray(arr, arr\_size);

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

0 0 0 0 0 1 1 1 1 1 2 2

5. Java Program to Check for balanced parenthesis by using Stacks

import java.util.\*;

public class Linked {

static boolean areBracketsBalanced(String expr)

{

Deque<Character> stack

= new ArrayDeque<Character>();

for (int i = 0; i < expr.length(); i++) {

char x = expr.charAt(i);

if (x == '(' || x == '[' || x == '{') {

// Push the element in the stack

stack.push(x);

continue;

}

if (stack.isEmpty())

return false;

char check;

switch (x) {

case ')':

check = stack.pop();

if (check == '{' || check == '[')

return false;

break;

case '}':

check = stack.pop();

if (check == '(' || check == '[')

return false;

break;

case ']':

check = stack.pop();

if (check == '(' || check == '{')

return false;

break;

}

}

return (stack.isEmpty());

}

public static void main(String[] args)

{

String expr = "([{}])";

if (areBracketsBalanced(expr))

System.out.println("Balanced ");

else

System.out.println("Not Balanced ");

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

Balanced

6. Java Program to Implement Stack

import java.util.\*;

public class Linked

{

static void stack\_push(Stack<Integer> stack)

{

for(int i = 0; i < 5; i++)

{

stack.push(i);

}

}

static void stack\_pop(Stack<Integer> stack)

{

System.out.println("Pop Operation:");

for(int i = 0; i < 5; i++)

{

Integer y = (Integer) stack.pop();

System.out.println(y);

}

}

static void stack\_peek(Stack<Integer> stack)

{

Integer element = (Integer) stack.peek();

System.out.println("Element on stack top: " + element);

}

static void stack\_search(Stack<Integer> stack, int element)

{

Integer pos = (Integer) stack.search(element);

if(pos == -1)

System.out.println("Element not found");

else

System.out.println("Element is found at position: " + pos);

}

public static void main (String[] args)

{

Stack<Integer> stack = new Stack<Integer>();

stack\_push(stack);

stack\_pop(stack);

stack\_push(stack);

stack\_peek(stack);

stack\_search(stack, 2);

stack\_search(stack, 6);

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

Pop Operation:

4

3

2

1

0

Element on stack top: 4

Element is found at position: 3

Element not found

7. Java Program to Implement Queue

import java.util.\*;

public class Linked

{

private int[] arr;

private int front;

private int rear;

private int size;

public Linked(int capacity) {

arr = new int[capacity];

front = 0;

rear = -1;

size = 0;

}

public void enqueue(int item) {

if (isFull()) {

System.out.println("Queue is full");

return;

}

rear++;

arr[rear] = item;

size++;

}

public int dequeue() {

if (isEmpty()) {

System.out.println("Queue is empty");

return -1;

}

int item = arr[front];

front++;

size--;

return item;

}

public int peek() {

if (isEmpty()) {

System.out.println("Queue is empty");

return -1;

}

return arr[front];

}

public boolean isEmpty() {

return size == 0;

}

public boolean isFull() {

return rear == arr.length - 1;

}

public int size() {

return size;

}

public static void main(String[] args)

{

Linked queue = new Linked(5);

queue.enqueue(7);

queue.enqueue(9);

System.out.println("Peek element: "+queue.peek());

System.out.println("Deleted element: " + queue.dequeue());

System.out.println("Deleted element: " + queue.dequeue());

queue.enqueue(11);

System.out.println("Deleted element: " + queue.dequeue());

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

Peek element: 7

Deleted element: 7

Deleted element: 9

Deleted element: 11

8. Java Program to Implement Dequeue.

import java.util.\*;

public class Linked

{

private int[] arr;

private int front;

private int rear;

private int size;

public Linked(int capacity) {

arr = new int[capacity];

front = 0;

rear = -1;

size = 0;

}

public void enqueue(int item) {

if (isFull()) {

System.out.println("Queue is full");

return;

}

rear++;

arr[rear] = item;

size++;

}

public int dequeue() {

if (isEmpty()) {

System.out.println("Queue is empty");

return -1;

}

int item = arr[front];

front++;

size--;

return item;

}

public int peek() {

if (isEmpty()) {

System.out.println("Queue is empty");

return -1;

}

return arr[front];

}

public boolean isEmpty() {

return size == 0;

}

public boolean isFull() {

return rear == arr.length - 1;

}

public int size() {

return size;

}

public static void main(String[] args)

{

Linked queue = new Linked(5);

queue.enqueue(7);

queue.enqueue(9);

System.out.println("Peek element: "+queue.peek());

System.out.println("Deleted element: " + queue.dequeue());

System.out.println("Deleted element: " + queue.dequeue());

queue.enqueue(11);

System.out.println("Deleted element: " + queue.dequeue());

}

}

10. Java Program to Implement Queue Using Two Stacks

import java.util.\*;

public class Linked

{

static class Queue

{

static Stack<Integer> s1 = new Stack<Integer>();

static Stack<Integer> s2 = new Stack<Integer>();

static void enQueue(int x)

{

while (!s1.isEmpty())

{

s2.push(s1.pop());

}

s1.push(x);

while (!s2.isEmpty())

{

s1.push(s2.pop());

}

}

static int deQueue()

{

if (s1.isEmpty())

{

return -1;

}

// Return top of s1

int x = s1.peek();

s1.pop();

return x;

}

}

public static void main(String[] args)

{

Queue q = new Queue();

q.enQueue(1);

q.enQueue(2);

q.enQueue(3);

System.out.println(q.deQueue());

System.out.println(q.deQueue());

System.out.println(q.deQueue());

}

}

C:\Users\mujah\Downloads\Algorithm and Data Structure>java Linked

1

2

3