Knapsack-greedy_method

Implement in Java, the 0/1 Knapsack problem using Greedy method

Input Format

7 15 6 10 18 15 3 5 7 1 2 4 5 1 3 7

Constraints

--

Output Format

Net Profit: 55.333333333333336 The objects picked up into knapsack are: 1.0 1.0 1.0 1.0 1.0 0.6666666666666666 0.0

Sample Input 0

```
7
15
6
10
18
15
3
5
7
1
2
4
5
1
3
7
```

Sample Output 0

```
Net Profit: 55.333333333333336
The objects picked up into knapsack are:
1.0
1.0
1.0
1.0
1.0
0.6666666666666666
0.0
```

Java 7

```
import java.util.Scanner;
class GKnapsack
{
    int n;
    double c;
    double p[];
    double w[];
    public GKnapsack(int n,double c,double[] p,double[] w)
    {
        super();
        this.n=n;
        this.c=c;
        this.p=p;
        this.w=w;
    }
    void compute()
    {
        int i;
        double[] x=new double[n+1];
        for(i=0;i<n;i++)
        {
            x[i]=0.0;

        }
        double rc=c;
        for(i=0;i<n;i++)
        {
            if(w[i]>rc)break;
            x[i]=1;
            rc=rc-w[i];
        }
        if(i<=n)
        {
            x[i]=rc/w[i];
        }
        double netProfit=0.0;
        for(i=0;i<n;i++)
        {
            if(x[i]>0.0)
            {
                netProfit=netProfit+x[i]*p[i];
            }
        }
        System.out.println("Net Profit: "+netProfit);
        System.out.println("The objects picked up into knapsack are:");
        for(i=0;i<n;i++)
        {
            System.out.println(x[i]+" ");
        }
    }
}
public class KpGreedy
{
    public static void main(String[] args)
    {
        int n;
        double c;
        Scanner input=new Scanner(System.in);
        //System.out.println("Enter number of objects");
        n=input.nextInt();
        double[] p=new double[n+1];
        double[] w=new double[n+1];
        int i;
        //System.out.println("Enter capacity of Knapsack");
        c=input.nextDouble();
        //System.out.println("Enter profit for each "+n+" objects");
        for(i=0;i<n;i++)
            p[i]=input.nextDouble();
        //System.out.println("Enter weight for each "+n+" objects");
```

```
70          for(i=0;i<n;i++)
71              w[i]=input.nextDouble();
72          GKnapsack gk=new GKnapsack(n,c,p,w);
73          gk.compute();
74      }
75  }
76
77
78
79
80
81
82
```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
7
15
6
10
18
15
3
5
7
1
2
4
5
1
3
7
```

Your Output (stdout)

```
Net Profit: 55.333333333333336
The objects picked up into knapsack are:
1.0
1.0
1.0
1.0
1.0
0.6666666666666666
0.0
```

Expected Output

```
Net Profit: 55.333333333333336
The objects picked up into knapsack are:
1.0
1.0
1.0
1.0
1.0
0.6666666666666666
0.0
```

Inorder Traversal 5

Problem Submissions Leaderboard Discussions

Write a java program to perform Inorder tree traversal.

Input Format

1 10 1 20 1 30 1 40 1 50 1 60 2 3

Constraints

No Constraints

Output Format

InorderTraversal is: 10 20 30 40 50 60

Sample Input 0

```
1
10
1
20
1
30
1
40
1
50
1
60
2
3
```

Sample Output 0

```
InorderTraversal is:
10 20 30 40 50 60
```


Java 7

```java
import java.util.*;
class Node {
    int data;
    Node left;
    Node right;
    public Node( int item) {
        this.data = item;
        this.left = null;
        this.right = null;
    }
}

class StackNode {
    Node node;
    StackNode next;
    public void StackNode(Node b) {
```

```
        this.node = b;
        this.next = null;
    }
}

public class NonRecursiveInorder {
    StackNode top;
    Node root;
    public void NonRecursiveInorder() {
        top = null;
        root = null;
    }
    boolean isEmpty() {
        if(top == null) {
            return true;
        }
        return false;
    }
    void push(Node b) {
        StackNode temp;
        temp = new StackNode();
        if(temp == null) {
            System.out.printf("Stack is overflow.\n");
        } else {
            temp.node = b;
            temp.next = top;
            top = temp;
        }
    }
    Node peek() {
        if (top == null) {
            return null;
        }
        return top.node;
    }
    Node pop() {
        StackNode temp;
        Node b;
        if(top == null) {
            System.out.printf("Stack is underflow.\n");
            return null;
        } else {
            temp = top;
            top = top.next;
            b = temp.node;
            return b;
        }
    }
    void inorderInBST(Node root) {
        Node curr = root;
        while(true) {
            if(curr!= null) {
                push(curr);
                curr = curr.left;
            } else {
                curr = pop();
                System.out.printf("%d ",curr.data);
                curr = curr.right;
            }
            if(isEmpty() && curr == null)
                break;
        }
    }
/* Insertion into binary search tree */
    Node insertBinarySearchTree(Node root, int item) {

        /* If the tree is empty new node became root */
        if (root == null) {
            root = new Node(item);
            return root;
        }

        /* Otherwise, if item is less then root then recur left side */
```

```
        if (item < root.data)
            root.left = insertBinarySearchTree(root.left, item);
        else if (item > root.data)
            root.right = insertBinarySearchTree(root.right, item);

        /* return the root node pointer */
        return root;
    }

    // Driver main method Code
    public static void main(String[] args) {
        NonRecursiveInorder tree = new NonRecursiveInorder();
        Scanner sc = new Scanner(System.in);
        int option;
        int item;
        //System.out.println("Enter 1 to insert\nEnter 2 to display BST in inorder\nEnter 3 to Exit");
        while(true) {
            //System.out.print("Enter your option: ");
            option = sc.nextInt();
            switch(option) {
                default:
                    System.out.println("Enter the right option");
                    break;
                case 1:
                    //System.out.print("Enter the element to insert: ");
                    item = sc.nextInt();
                    tree.root = tree.insertBinarySearchTree(tree.root, item);
                    break;
                case 2:
                    if(tree.root == null) {
                        System.out.println("Tree is empty, root is null");
                    }else {
                        System.out.println("InorderTraversal is:");
tree.inorderInBST(tree.root);
                        System.out.println();
                    }
                    break;
                case 3:
                    return;
            }
        }

    }
}

```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
1
10
1
20
1
30
1
40
1
50
1
60
```

```
2
3
```

Your Output (stdout)

```
InorderTraversal is:
10 20 30 40 50 60
```

Expected Output

```
InorderTraversal is:
10 20 30 40 50 60
```

Dijkstra's Algorithm 4

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm. Write the program in Java.

Input Format

4 0 15 10 9999 9999 0 15 9999 20 9999 0 20 9999 10 9999 0 3

Constraints

--

Output Format

Shortest path from 3 to all other vertices To 0 is 45 To 1 is 10 To 2 is 25 To 3 is 0

Sample Input 0

```
4
0 15 10 9999
9999 0 15 9999
20 9999 0 20
9999 10 9999 0
3
```

Sample Output 0

```
Shortest path from 3 to all other vertices
To 0 is 45
To 1 is 10
To 2 is 25
To 3 is 0
```


Java 7

```
1  import java.util.*;
2
3  public class DijkstraAlgorithm {
4      private static final int INF = 9999;
5
6      public static void main(String[] args) {
```

```java
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt(); // Number of vertices
        int[][] graph = new int[n][n];

        // Taking input for the weighted graph
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                graph[i][j] = scanner.nextInt();
            }
        }

        int source = scanner.nextInt(); // Source vertex
        dijkstra(graph, source, n);
        scanner.close();
    }

    private static void dijkstra(int[][] graph, int source, int n) {
        boolean[] visited = new boolean[n];
        int[] distance = new int[n];
        Arrays.fill(distance, INF);
        distance[source] = 0;

        for (int i = 0; i < n - 1; i++) {
            int u = minDistance(distance, visited, n);
            visited[u] = true;

            for (int v = 0; v < n; v++) {
                if (!visited[v] && graph[u][v] != 0 && distance[u] != INF &&
                        distance[u] + graph[u][v] < distance[v]) {
                    distance[v] = distance[u] + graph[u][v];
                }
            }
        }

        // Printing the shortest paths
        System.out.println("Shortest path from " + source + " to all other vertices");
        for (int i = 0; i < n; i++) {
            System.out.println("To " + i + " is " + distance[i]);
        }
    }

    private static int minDistance(int[] distance, boolean[] visited, int n) {
        int min = INF, minIndex = -1;
        for (int i = 0; i < n; i++) {
            if (!visited[i] && distance[i] <= min) {
                min = distance[i];
                minIndex = i;
            }
        }
        return minIndex;
    }
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
4
0 15 10 9999
9999 0 15 9999
20 9999 0 20
```

```
9999 10 9999 0
3
```

Your Output (stdout)

```
Shortest path from 3 to all other vertices
To 0 is 45
To 1 is 10
To 2 is 25
To 3 is 0
```

Expected Output

```
Shortest path from 3 to all other vertices
To 0 is 45
To 1 is 10
To 2 is 25
To 3 is 0
```

Postorder Traversal 1

Problem Submissions Leaderboard Discussions

Write a java program to perform postorder tree traversal.

Input Format

1 10 1 20 1 30 1 40 1 60 1 50 2 3

Constraints

No Constraints

Output Format

Postorder Traveral is: 50 60 40 30 20 10

Sample Input 0

```
1
10
1
20
1
30
1
40
1
60
1
50
2
3
```

Sample Output 0

```
Postorder Traversal is:
50 60 40 30 20 10
```


Java 7

```
1  import java.util.*;
2  class Node {
3      int data;
4      Node left;
5      Node right;
6      public Node( int item) {
7          this.data = item;
8          this.left = null;
9          this.right = null;
10     }
11 }
12
13 class StackNode {
14     Node node;
15     StackNode next;
16     public void StackNode(Node b) {
```

```
        this.node = b;
        this.next = null;
    }
}

public class NonRecursivePostorder {
    StackNode top;
    Node root;
    public void NonRecursivePostorder() {
        top = null;
        root = null;
    }
    boolean isEmpty() {
        if(top == null) {
            return true;
        }
        return false;
    }
    void push(Node b) {
        StackNode temp;
        temp = new StackNode();
        if(temp == null) {
            System.out.printf("Stack is overflow.\n");
        } else {
            temp.node = b;
            temp.next = top;
            top = temp;
        }
    }
    Node peek() {
        if (top == null) {
            return null;
        }
        return top.node;
    }
    Node pop() {
        StackNode temp;
        Node b;
        if(top == null) {
            System.out.printf("Stack is underflow.\n");
            return null;
        } else {
            temp = top;
            top = top.next;
            b = temp.node;
            return b;
        }
    }
    void postorderInBST(Node root) {
    do {
        while(root != null) {
            if(root.right != null) {
                push(root.right);
            }
            push(root);
            root = root.left;
        }
        root = pop();
        if(root.right != null && peek() == root.right) {
            pop();
            push(root);
            root = root.right;
        } else {
            System.out.printf("%d ",root.data);
            root = null;
        }
    } while(!isEmpty());
}
/* Insertion into binary search tree */
    Node insertBinarySearchTree(Node root, int item) {

        /* If the tree is empty new node became root */
        if (root == null) {
```

```
            root = new Node(item);
            return root;
        }

        /* Otherwise, if item is less then root then recur left side */
        if (item < root.data)
            root.left = insertBinarySearchTree(root.left, item);
        else if (item > root.data)
            root.right = insertBinarySearchTree(root.right, item);

        /* return the root node pointer */
        return root;
    }

    // Driver main method Code
    public static void main(String[] args) {
        NonRecursivePostorder tree = new NonRecursivePostorder();
        Scanner sc = new Scanner(System.in);
        int option;
        int item;
        //System.out.println("Enter 1 to insert\nEnter 2 to display BST in postorder\nEnter 3 to Exit");
        while(true) {
            //System.out.print("Enter your option: ");
            option = sc.nextInt();
            switch(option) {
                default:
                    System.out.println("Enter the right option");
                    break;
                case 1:
                    //System.out.print("Enter the element to insert: ");
                    item = sc.nextInt();
                    tree.root = tree.insertBinarySearchTree(tree.root, item);
                    break;
                case 2:
                    if(tree.root == null) {
                        System.out.println("Tree is empty, root is null");
                    }else {
                        System.out.println("Postorder Traversal is:");
tree.postorderInBST(tree.root);
                        System.out.println();
                }
                    break;
                case 3:
                    return;
            }
        }

    }
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
1
10
1
20
1
30
1
40
1
60
1
50
2
3
```

Your Output (stdout)

```
Postorder Traversal is:
50 60 40 30 20 10
```

Expected Output

```
Postorder Traversal is:
50 60 40 30 20 10
```

All_pair_Shortest_path

Write Java program to Implement All-Pairs Shortest Paths problem using Floyd's algorithm

Input Format

5 0 5 999 2 999 999 0 2 999 999 3 999 0 999 7 999 999 4 0 1 1 3 999 999 0

Constraints

no constraints

Output Format

all pair shortest paths matrix. 0 5 6 2 3 5 0 2 7 8 3 8 0 5 6 2 4 4 0 1 1 3 5 3 0

Sample Input 0

```
5
0 5 999 2 999
999 0 2 999 999
3 999 0 999 7
999 999 4 0 1
1 3 999 999 0
```

Sample Output 0

```
all pair shortest paths matrix.
0 5 6 2 3
5 0 2 7 8
3 8 0 5 6
2 4 4 0 1
1 3 5 3 0
```


Java 7

```
1  import java.util.*;
2  public class Floyds {
3   static int n,i,j,k;
4   public void floyd(int n , int[][] cost)
5   {
6     for(k=1;k<=n;k++)
7     {
8         for(i=1;i<=n;i++)
9         {
10            for(j=1;j<=n;j++)
11            {
12                cost[i][j]=min(cost[i][j],cost[i][k]+cost[k][j]);
13            }
14        }
15    }
16    System.out.println("all pair shortest paths matrix.");
17    for(i=1;i<=n;i++)
18    {
19        for(j=1;j<=n;j++)
```

```
      {
         System.out.print(cost[i][j]+" ");
      }
     System.out.println();
  }
 }
 public int min(int i,int j)
 {
    if(i<j)
        return i;
    else
        return j;
 }
 public static void main(String[] args)
 {
    Scanner sc=new Scanner(System.in);
    //System.out.println("Enter the no of vertices: ");
    n=sc.nextInt();
    int cost[][]=new int[n+1][n+1];
    //System.out.println("Enter the cost matrix:");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            cost[i][j]=sc.nextInt();
    Floyds f = new Floyds();
          f.floyd(n,cost);
  }
}
```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
0 5 999 2 999
999 0 2 999 999
3 999 0 999 7
999 999 4 0 1
1 3 999 999 0
```

Your Output (stdout)

```
all pair shortest paths matrix.
0 5 6 2 3
5 0 2 7 8
3 8 0 5 6
2 4 4 0 1
1 3 5 3 0
```

Expected Output

```
all pair shortest paths matrix.
0 5 6 2 3
5 0 2 7 8
3 8 0 5 6
2 4 4 0 1
1 3 5 3 0
```

Stack using arrays

Write a Java program to implement the Stack using arrays. Write push(), pop(), and display() methods to demonstrate its working.

Input Format

3 1 53 1 68 1 20 2 2 2 3 4

Constraints

size of stack should be positive

Output Format

pushed element 53 pushed element 68 pushed element 20 Popped element 20 Popped element 68 Popped element 53 Stack Empty

Sample Input 0

```
3
1
53
1
68
1
20
2
2
2
3
4
```

Sample Output 0

```
pushed element 53
pushed element 68
pushed element 20
Popped element 20
Popped element 68
Popped element 53
Stack Empty
```

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
 static int top = -1;
 static int arr[] = new int[50];
 public void push(int num, int ele) {
   if (top == num - 1) {
     System.out.println("Stack is overflow");
   } else {
     top++;
     arr[top] = ele;
     System.out.println("pushed element "+ele);
   }
 }
 public void pop() {
   if (top == -1) {
      System.out.println("Stack is underflow");
   } else {
      System.out.println("Popped element " + arr[top]);
      top--;
   }
 }
 public void display(int[] arr2, int num) {
   if (top < 0) {
     System.out.print("Stack Empty");
 } else {
     System.out.print("ELEMENTS : ");
     for (int i = top; i >= 0; i--) {
       System.out.print(arr2[i] + " ");
     }
 }
 System.out.println();
}
public static void main(String[] args) {
 Solution su = new Solution();
 Scanner sc = new Scanner(System.in);
 int num, opt;
 int ele;
 num = sc.nextInt();
 Boolean kl = true;
 while (kl) {
  opt = sc.nextInt();
  switch (opt) {
  case 1:
   ele = sc.nextInt();
   su.push(num, ele);
   break;
 case 2:
   su.pop();
   break;
 case 3:
   su.display(arr, num);
   break;
 case 4:
   kl = false;
   break;
 default:
 break;
 }
}
}
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
3
1
53
1
68
1
20
2
2
2
3
4
```

Your Output (stdout)

```
pushed element 53
pushed element 68
pushed element 20
Popped element 20
Popped element 68
Popped element 53
Stack Empty
```

Expected Output

```
pushed element 53
pushed element 68
pushed element 20
Popped element 20
Popped element 68
Popped element 53
Stack Empty
```

Quick-sort

Sort a given set of n integer elements using Quick Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus non graph sheet.The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide - and - conquer method works along with its time complexity analysis: worst case, average case and best case.

Input Format

5 0 0 4 3 1

Constraints

Size of the array should be always positive

Output Format

Before Sort: 0 0 4 3 1 After sort: 0 0 1 3 4

Sample Input 0

```
5
0
0
4
3
1
```

Sample Output 0

```
Before Sort:
0
0
4
3
1
After sort:
0
0
1
3
4
```

Java 7

```java
import java.util.Scanner;
class QuickSort {
    private int a[];
    public QuickSort(int[] a)
    {
        this.a = a;
    }
    public int partition ( int a[], int m, int p ) {
        int v = a[m];
        int i = m;
        int j = p;
        do {
            while (a[++ i] <v);
            while ( a[-- j] > v );
            if ( i < j )
                interchange ( a, i, j );

        }
        while ( i <= j );
        a[m] = a[j]; a[j] = v;
        return j;
    }
    public void qSort ( int p, int q ) {
        int j;
        if ( p < q ) {
            j = partition ( a, p, q + 1 );
            qSort ( p, j - 1 );
            qSort ( j + 1, q );

        }
    }
    public void interchange ( int a[], int i, int j ) {
        int t;
        t = a[i];
        a[i] = a[j];
        a[j] = t;
    }
}
public class QuickSortDemo {
    public static void main(String[] args) {
        int n, a[], i;
        Scanner input = new Scanner(System.in);
        //System.out.print("Enter the Size of an Array: ");
        n = input.nextInt();
        a = new int[n + 1];
        //System.out.println("System automatically generates numbers ");
        for ( i = 0; i < n; ++ i ){
            a[i] = input.nextInt(n);
        }
        a[i] = 100000; //Sentinel value
        QuickSort qSort = new QuickSort(a);
        System.out.println("Before Sort: ");
        for ( i = 0; i < n; ++ i ) {
            System.out.print(a[i] + "\n");
        }
        int p = 0;
        int q = n - 1;
        qSort.qSort(p, q);
        System.out.println("After sort: ");
        for ( i = 0; i < n; ++ i ) {
            System.out.print(a[i] + "\n");
        }
    }
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
0
0
4
3
1
```

Your Output (stdout)

```
Before Sort:
0
0
4
3
1
After sort:
0
0
1
3
4
```

Expected Output

```
Before Sort:
0
0
4
3
1
After sort:
0
0
1
3
4
```

Knapsack-Dynamic_programming

Problem Submissions Leaderboard Discussions

Implement in Java, the 0/1 Knapsack problem using Dynamic Programming method

Input Format

5 20 3 4 5 6 8 2 3 4 5 6

Constraints

--

Output Format

Optimal Solution: 26 The objects picked up into knapsack are: 5 4 3 2 1

Sample Input 0

```
5
20
3
4
5
6
8
2
3
4
5
6
```

Sample Output 0

```
Optimal Solution: 26
The objects picked up into knapsack are:
5 4 3 2 1
```


Java 7

```java
import java.util.Scanner;
class DKnapsack {
    int c,n,p[],w[],v[][];
    public DKnapsack(int n,int c,int[] p,int[] w)
    {
        super();
        this.n=n;
        this.c=c;
        this.w=w;
        this.p=p;
        this.v=new int[n+1][c+1];
    }
    void compute()
    {
        for(int i=0;i<=n;++i){
            for(int j=0;j<=c;++j){
                if(i==0||j==0){
```

```
                    v[i][j]=0;
                }
                else if(j-w[i]>=0)
                {
                    v[i][j]=max(v[i-1][j],p[i]+v[i-1][j-w[i]]);
                }
                else if(j-w[i]<0)
                {
                    v[i][j]=v[i-1][j];
                }
            }
        }
        System.out.println("Optimal Solution: "+v[n][c]);
        traceback();
    }
    void traceback(){
        System.out.println("The objects picked up into knapsack are:");
        int i=n;
        int j=c;
        while(i>0)
        {
            if(v[i][j]!=v[i-1][j])
            {
                System.out.print(i+" ");
                j=j-w[i];
                i--;
            }
            else {
                i--;
            }
        }
    }
    private int max(int i,int j){
        if(i>j)return i;
        else return j;
    }
}
public class KpDynamic{
    public static void main(String[] args){
        int c,n;
        Scanner input=new Scanner(System.in);
        //System.out.println("Enter number of objects");
        n=input.nextInt();
        int[] p=new int[n+1];
        int[] w=new int[n+1];
        int i;
        //System.out.println("Enter capacity of Knapsack");
        c=input.nextInt();
        //System.out.println("Enter profit for each "+n+" objects");
        for(i=1;i<=n;i++)
            p[i]=input.nextInt();
        //System.out.println("Enter weight for each "+n+" objects");
        for(i=1;i<=n;i++)
            w[i]=input.nextInt();
        DKnapsack dk=new DKnapsack(n,c,p,w);
        dk.compute();
    }
}

```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
20
3
4
5
6
8
2
3
4
5
6
```

Your Output (stdout)

```
Optimal Solution: 26
The objects picked up into knapsack are:
5 4 3 2 1
```

Expected Output

```
Optimal Solution: 26
The objects picked up into knapsack are:
5 4 3 2 1
```

Breadth first search(BFS) 1

Problem Submissions Leaderboard Discussions

Write a Java Program to print all nodes reachable from a given starting node in a digraph using BFS method

Input Format

4 5 1 3 0 2 2 3 3 0 0 3 2

Constraints

positive vertex values only

Output Format

Enter the number of vertices : 4 Enter the number of edges : 5 Enter source : 1 Enter destination : 3 Enter source : 0 Enter destination : 2 Enter source : 2 Enter destination : 3 Enter source : 3 Enter destination : 0 Enter source : 0 Enter destination : 3 Enter Start Vertex for BFS : 2 BFS of graph : 2 3 0

Sample Input 0

```
4
5
1
3
0
2
2
3
3
0
0
3
2
```

Sample Output 0

```
BFS of graph : 2 3 0
```

```java
import java.text.*;
import java.math.*;
import java.util.regex.*;
class Gnode {
    Gnode next;
    int vertex;
}
public class Solution {
    Gnode graph[ ];
    boolean visited[ ];
    int queue[];
    int numVertices;
    int front;
    int rear;
    public Solution(int n) {
        graph = new Gnode[n];
        visited = new boolean[n];
        queue = new int[n];
        numVertices = n;
        front = -1;
        rear = -1;

    }
    void insertQueue(int vertex) {
        if(rear == numVertices - 1){}
        //System.out.printf("Queue Overflow.\n");
        else {
            if(front == -1)
                front = 0;
            rear = rear + 1;
            queue[rear] = vertex ;
        }
    }

    boolean isEmptyQueue() {
        if(front == -1 || front > rear)
            return true;
        else
            return false;
    }

    int deleteQueue() {
        int deleteItem;
        if(isEmptyQueue()) {
            //System.out.printf("Queue Underflow\n");
            return -1;
        }
        deleteItem = queue[front];
        front = front + 1;
        return deleteItem;
    }
    void Bfs(int v) {
        int w;
        insertQueue(v);
        while(!isEmptyQueue()) {
            v = deleteQueue( );
            System.out.printf(" %d",v);
            visited[v] = true;
            Gnode g = graph[v];
            for( ; g != null; g = g.next) {
                w = g.vertex;
                if(visited[w] == false) {
                    insertQueue(w);
                    visited[w] = true;
                }
            }
        }
    }
    public static void main(String []args) {
        int n, e, i, s, d, v;
        Gnode q, p;
        Scanner sc = new Scanner(System.in);

```

```
        //System.out.printf("Enter the number of vertices : ");
        n = sc.nextInt();
        //System.out.printf("Enter the number of edges : ");
        e = sc.nextInt();
        Solution g = new Solution(n);
        for(i=1;i<=e;i++) {
            //System.out.printf("Enter source : ");
            s = sc.nextInt();
            //System.out.printf("Enter destination : ");
            d = sc.nextInt();
            q = new Gnode();
            q.vertex = d;
            q.next = null;
            if(g.graph[s] == null)
                g.graph[s]=q;
            else {
                p=g.graph[s];
                while(p.next != null)
                    p = p.next;
                p.next = q;
            }
        }
        for(i = 0;i < n;i++)
            g.visited[i] = false;
        //System.out.printf("Enter Start Vertex for BFS : ");
        v = sc.nextInt();
        System.out.printf("BFS of graph :");
        g.Bfs(v);
        System.out.printf("\n");
    }
}
```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
4
5
1
3
0
2
2
3
3
0
0
3
2
```

Your Output (stdout)

```
BFS of graph : 2 3 0
```

Expected Output

```
BFS of graph : 2 3 0
```

Preorder Traversal 4

Problem Submissions Leaderboard Discussions

Write a java program to perform preorder tree order tree traversal

Input Format

1 10 1 20 1 30 1 40 1 50 2 3

Constraints

No constraints

Output Format

Preorder Traversal is: 10 20 30 40 50

Sample Input 0

```
1
10
1
20
1
30
1
40
1
50
2
3
```

Sample Output 0

```
Preorder Traversal is:
10 20 30 40 50
```


Java 7

```
1  import java.util.*;
2  class Node {
3      int data;
4      Node left;
5      Node right;
6      public Node( int item) {
7          this.data = item;
8          this.left = null;
9          this.right = null;
10     }
11 }
12
13 class StackNode {
14     Node node;
15     StackNode next;
16     public void StackNode(Node b) {
17         this.node = b;
18         this.next = null;
```

```
    }
}

public class NonRecursivePreorder {
    StackNode top;
    Node root;
    public void NonRecursivePreorder() {
        top = null;
        root = null;
    }
    boolean isEmpty() {
        if(top == null) {
            return true;
        }
        return false;
    }
    void push(Node b) {
        StackNode temp;
        temp = new StackNode();
        if(temp == null) {
            System.out.printf("Stack is overflow.\n");
        } else {
            temp.node = b;
            temp.next = top;
            top = temp;
        }
    }
    Node peek() {
        if (top == null) {
            return null;
        }
        return top.node;
    }
    Node pop() {
        StackNode temp;
        Node b;
        if(top == null) {
            System.out.printf("Stack is underflow.\n");
            return null;
        } else {
            temp = top;
            top = top.next;
            b = temp.node;
            return b;
        }
    }
    void preorderInBST(Node root) {
        Node curr = root;
        push(root);
        while(true) {
            curr = pop();
            System.out.printf("%d ",curr.data);
            if(curr.right != null) {
                push(curr.right);
            }
            if(curr.left != null) {
                push(curr.left);
            }
            if(isEmpty())
                break;
        }
    }
/* Insertion into binary search tree */
    Node insertBinarySearchTree(Node root, int item) {

        /* If the tree is empty new node became root */
        if (root == null) {
            root = new Node(item);
            return root;
        }

        /* Otherwise, if item is less then root then recur left side */
        if (item < root.data)
```

```
            root.left = insertBinarySearchTree(root.left, item);
        else if (item > root.data)
            root.right = insertBinarySearchTree(root.right, item);

        /* return the root node pointer */
        return root;
    }

    // Driver main method Code
    public static void main(String[] args) {
        NonRecursivePreorder tree = new NonRecursivePreorder();
        Scanner sc = new Scanner(System.in);
        int option;         int item;
        //System.out.println("Enter 1 to insert\nEnter 2 to display BST in preorder\nEnter 3 to Exit");
        while(true) {
            //System.out.print("Enter your option: ");
            option = sc.nextInt();
            switch(option) {
                default:
                    System.out.println("Enter the right option");
                    break;
                case 1:
                    //System.out.print("Enter the element to insert: ");
                    item= sc.nextInt();
                    tree.root = tree.insertBinarySearchTree(tree.root,item);
                    break;
                case 2:
                    if(tree.root == null) {
                        System.out.println("Tree is empty, root is null");
                    }else {

                        System.out.println("Preorder Traversal is:");
tree.preorderInBST(tree.root);
                        System.out.println();

                    }
                    break;
                case 3:
                    return;
            }
        }

    }
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
1
10
1
20
1
30
1
40
```

```
1
50
2
3
```

Your Output (stdout)

```
Preorder Traversal is:
10 20 30 40 50
```

Expected Output

```
Preorder Traversal is:
10 20 30 40 50
```

Krushkals_algorithm

Find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm. Use Union-Find algorithms in your program

Input Format

7 0 28 999 999 999 10 999 28 0 16 999 999 999 14 999 16 0 12 999 999 999 999 999 12 0 22 999 18 999 999 999 22 0 25 24 10 999 999 999 25 999 999 999 14 999 18 24 999 999

Constraints

No Constraints

Output Format

1edge(1,6)=10 2edge(3,4)=12 3edge(2,7)=14 4edge(2,3)=16 5edge(4,5)=22 6edge(5,6)=25 The minimum cost of spanning tree is 99

Sample Input 0

```
7
0 28 999 999 999 10 999
28 0 16 999 999 999 14
999 16 0 12 999 999 999
999 999 12 0 22 999 18
999 999 999 22 0 25 24
10 999 999 999 25 999 999
999 14 999 18 24 999 999
```

Sample Output 0

```
1edge(1,6)=10
2edge(3,4)=12
3edge(2,7)=14
4edge(2,3)=16
5edge(4,5)=22
6edge(5,6)=25
The minimum cost of spanning tree is 99
```

```
public class Kruskals
{
    static int parent[],cost[][], mincost,n,i,j,ne,a,b,min,u,v;
    public void kruskal(int n,int[][] cost)
    {
        ne=1;
        while(ne<n)
        {
            min=999;
            for(i=1;i<=n;i++)
            {
                for(j=1;j<=n;j++)
                    if(cost[i][j]<min)
                    {
                        min=cost[i][j];
                        a=u=i;
                        b=v=j;
                    }
            }
            u=find(u);
            v=find(v);
            if(v!=u)
            {
                System.out.println( ne+"edge("+a+","+b+")="+min);
                ne=ne+1;
                mincost=mincost+min;
                uni(u,v);
            }
            cost[a][b]=cost[b][a]=999;
        }
        System.out.println("The minimum cost of spanning tree is "+mincost);
    }
    public int find (int i)
    {
        while (parent[i] != 0)
        i=parent[i];
        return i;
    }
    public void uni(int i,int j)
    {
        parent[j]=i;
    }
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        //System.out.println("Enter the number of vertices: ");
        n=sc.nextInt();
        int cost[][]= new int [n+1][n+1];
        parent=new int[n+1];
        //System.out.println("Enter the cost matrix:");
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                cost[i][j]=sc.nextInt();
                if(cost[i][j]==0)
                    cost[i][j]=999;
            }
        }
        Kruskals k = new Kruskals();
        k.kruskal(n,cost);
    }
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
7
0 28 999 999 999 10 999
28 0 16 999 999 999 14
999 16 0 12 999 999 999
999 999 12 0 22 999 18
999 999 999 22 0 25 24
10 999 999 999 25 999 999
999 14 999 18 24 999 999
```

Your Output (stdout)

```
1edge(1,6)=10
2edge(3,4)=12
3edge(2,7)=14
4edge(2,3)=16
5edge(4,5)=22
6edge(5,6)=25
The minimum cost of spanning tree is 99
```

Expected Output

```
1edge(1,6)=10
2edge(3,4)=12
3edge(2,7)=14
4edge(2,3)=16
5edge(4,5)=22
6edge(5,6)=25
The minimum cost of spanning tree is 99
```

Depth First Search(DFS)

Write a Java program to print all the nodes reachable from a given starting node in a digraph using the DFS method.

Input Format

4 6 0 3 2 3 1 2 3 1 2 0 3 0 1

Constraints

number of vertices should be positive

Output Format

DFS of graph : 1 2 3 0

Sample Input 0

```
4
6
0
3
2
3
1
2
3
1
2
0
3
0
1
```

Sample Output 0

```
DFS of graph : 1 2 3 0
```

```java
import java.math.*;
import java.util.regex.*;
class Gnode {
    Gnode next;
    int vertex;

}
public class Solution {
    Gnode graph[];
    boolean visited[];
    int numVertices;
    public Solution(int n) {
        this.numVertices = n;
        visited = new boolean[n];
        graph = new Gnode[n];

    }
    void Dfs(int i) {
        Gnode p;
        System.out.printf(" %d",i);
        p = graph[i];
        visited[i] = true;
        while(p != null) {
            i = p.vertex;
            if(visited[i] == false)
                Dfs(i);
            p = p.next;

        }

    }
    public static void main(String []args) {
        int n, e, i, s, d, v;
        Gnode q, p;
        Scanner sc = new Scanner(System.in);
        //System.out.printf("Enter the number of vertices : ");
        n = sc.nextInt();
        //System.out.printf("Enter the number of edges : ");
        e = sc.nextInt();
        Solution g = new Solution(n);
        for(i=1;i<=e;i++) {
            //System.out.printf("Enter source : ");
            s = sc.nextInt();
            //System.out.printf("Enter destination : ");
            d = sc.nextInt();
            q = new Gnode();
            q.vertex=d;
            q.next=null;
            if(g.graph[s]==null)
                g.graph[s]=q;
            else {
                p=g.graph[s];
                while(p.next!=null)
                    p=p.next;
                p.next=q;
            }
        }
        for(i=0;i<n;i++)
            g.visited[i] = false;
        //System.out.printf("Enter Start Vertex for DFS : ");
        v = sc.nextInt();
        System.out.printf("DFS of graph :");
        g.Dfs(v);
        System.out.printf("\n");

    }
}

```

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
4
6
0
3
2
3
1
2
3
1
2
0
3
0
1
```

Your Output (stdout)

```
DFS of graph : 1 2 3 0
```

Expected Output

```
DFS of graph : 1 2 3 0
```

Travelling_Sales_person

Problem Submissions Leaderboard Discussions

Write Java program to Implement Travelling Sales Person problem using Dynamic programming

Input Format

4 999 1 3 6 1 999 2 3 3 2 999 1 6 3 1 999

Constraints

No Constraints

Output Format

tsp tour 1--->2--->4--->3--->1 Tourcost=8

Sample Input 0

```
4
999 1 3 6
1 999 2 3
3 2 999 1
6 3 1 999
```

Sample Output 0

```
tsp tour
1--->2--->4--->3--->1
Tourcost=8
```


Java 7

```
1  import java.util.Scanner;
2  public class Tsp
3  {
4       static int cost[][];
5       public int tsp(int[] path,int start,int n)
6       {
7           int i,j,k,ccost;
8           int[] mintour=new int[n+1];
9           int[] temp=new int[n+1];
10          if(start==n-1)
11          return cost[path[n-1]][path[n]]+cost[path[n]][1];
12          int mincost=999;
13          for(i=start+1;i<=n;i++)
14          {
15             for(j=1;j<=n;j++)
16             temp[j]=path[j];
17             temp[start+1]=path[i];
18             temp[i]=path[start+1];
19             if(cost[path[start]][path[i]]+(ccost=tsp(temp,start+1,n))<mincost)
20             {
21                 mincost=cost[path[start]][path[i]]+ccost;
22                 for(k=1;k<=n;k++)
23                 mintour[k]=temp[k];
```

```
            }
        }
        for(i=1;i<=n;i++)
        path[i]=mintour[i];
        return mincost;
    }
    public static void main(String[] args)
    {
        int mincost,n,i,j;
        Scanner s = new Scanner(System.in);
        //System.out.println("enter the no of cities: ");
        n=s.nextInt();
        int path[] =new int[n+1];
        cost = new int[n+1][n+1];
        //System.out.println("Enter the cost matrix:");
        for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
              cost[i][j]=s.nextInt();
        for(i=1;i<=n;i++)
        path[i]=i;
        Tsp obj = new Tsp();
        mincost=obj.tsp(path,1,n);
        System.out.println("tsp tour");
        for(i=1;i<=n;i++)
            System.out.print(path[i] + "--->");
        System.out.println("1");
        System.out.println("Tourcost=" + mincost);
    }
}
```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
4
999 1 3 6
1 999 2 3
3 2 999 1
6 3 1 999
```

Your Output (stdout)

```
tsp tour
1--->2--->4--->3--->1
Tourcost=8
```

Expected Output

```
tsp tour
1--->2--->4--->3--->1
Tourcost=8
```

Queue using Arrays

Write a Java program to implement the Queue using arrays. Write insert(), delete(), and display() methods to demonstrate its working.

Input Format

3 1 53 1 68 1 20 2 2 2 3 4

Constraints

Size of Queue should be always positive

Output Format

Inserted Element is 53 Inserted Element is 68 Inserted Element is 20 Dequeued Element is 53 Dequeued Element is 68 Dequeued Element is 20 Queue is Empty

Sample Input 0

```
3
1
53
1
68
1
20
2
2
2
3
4
```

Sample Output 0

```
Inserted Element is 53
Inserted Element is 68
Inserted Element is 20
Dequeued Element is 53
Dequeued Element is 68
Dequeued Element is 20
Queue is Empty
```

Java 7

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
 static int front = -1, rear = -1;
 static int arr[] = new int[50];
 public void insert(int num, int ele) {
 if (rear == num - 1) {
 System.out.println("Queue is overflow");
 } else {
 rear++;
 arr[rear] = ele;
 System.out.println("Inserted Element is "+ele);
 }
 if (front == -1) {
 front++;
 }
 }
 public void delete() {
 if (front == -1) {
 System.out.println("Queue is underflow");
 } else {
 System.out.println("Dequeued Element is "+ arr[front]);
 if (front == rear) {
 front = rear = -1;
 } else {
 front++;
 }
 }
 }
 public void display(int[] arr, int num) {
 if (rear == -1 && front == -1) {
 System.out.print("Queue is Empty");
 } else {
 System.out.print("ELEMENTS : ");
 for (int i = front; i <= rear; i++) {
 System.out.print(arr[i] + " ");
 }
 }
 System.out.println();
 }
 public static void main(String[] args) {
 Solution qe = new Solution();
 Scanner sc = new Scanner(System.in);
 int num, opt;
 int ele;
 num = sc.nextInt();
 Boolean kl = true;
 while (kl) {
 opt = sc.nextInt();
 switch (opt) {
 case 1:
 ele = sc.nextInt();
 qe.insert(num, ele);
 break;
 case 2:
 qe.delete();
 break;
 case 3:
 qe.display(arr, num);
 break;
 case 4:
 kl = false;
 break;
 default :
 break;
 }
```

```
    }
72  }
73 }
```

Line: 1 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
3
1
53
1
68
1
20
2
2
2
3
4
```

Your Output (stdout)

```
Inserted Element is 53
Inserted Element is 68
Inserted Element is 20
Dequeued Element is 53
Dequeued Element is 68
Dequeued Element is 20
Queue is Empty
```

Expected Output

```
Inserted Element is 53
Inserted Element is 68
Inserted Element is 20
Dequeued Element is 53
Dequeued Element is 68
Dequeued Element is 20
Queue is Empty
```

Merge-sort 2

Sort a given set of n integer elements using Merge Sort method and compute its time complexity. Run the program for varied values of n> 5000 and record the time taken to sort. Plot a graph of the time taken versus non graph sheet.The elements can be read from a file or can be generated using the random number generator. Demonstrate using Java how the divide - and - conquer method works along with its time complexity analysis: worst case, average case and best case.

Input Format

5 0 0 4 3 1

Constraints

Size of the array should be always positive

Output Format

Before Sort: 0 0 4 3 1 After sort: 0 0 1 3 4

Sample Input 0

```
5
0
0
4
3
1
```

Sample Output 0

```
Before Sort:
0
0
4
3
1
After sort:
0
0
1
3
4
```

Java 7

```java
import java.util.Scanner;
class MergeSort {
    private int a[];
    public MergeSort(int[] a) {
        this.a = a;
    }
    void merge ( int low, int mid, int high ) {
        int b[] = new int[high + 1];
        int h = low;
        int i = low;
        int j = mid + 1;
        int k;
        while ( ( h <= mid ) && ( j <= high ) ) {
            if ( a[h] <= a[j] ) b[i ++] = a[h ++];
            else b[i ++] = a[j ++];
        }
        if (h > mid) {
            for ( k = j; k <= high; ++ k )
                b[i ++] = a[k];
        }
        else {
            for ( k = h; k <= mid; ++ k )
                b[i ++] = a[k];
        }
        for (k=low; k<= high; ++ k)
            a[k] =b[k];
    }
    void mergeSort ( int low, int high ) {
        int mid;
        if ( low < high ) {
            mid = ( low + high ) / 2;
            mergeSort ( low, mid );
            mergeSort ( mid + 1, high );
            merge ( low, mid, high );
        }
    }
}
public class MergeSortDemo {
    public static void main(String[] args) {
        int n, a[], i;
        Scanner input = new Scanner(System.in);
        //System.out.println("Enter the Size of an Array: ");
        n = input.nextInt();
        a = new int[n + 1];
        //System.out.println("System automatically generates numbers ");
        for ( i = 0; i < n; ++ i ) {
            a[i] = input.nextInt(n);
        }
        a[i] = 100000;
        MergeSort mSort = new MergeSort(a);
        System.out.println("Before Sort: ");
        for ( i = 0; i < n; ++ i ) {
            System.out.print(a[i] + "\n");
        }
        int low = 0;
        int high = n - 1;
        mSort.mergeSort(low, high);
        System.out.println("After sort: ");
        for ( i = 0; i < n; ++ i ) {
            System.out.print(a[i] + "\n");
        }
    }
}

```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
0
0
4
3
1
```

Your Output (stdout)

```
Before Sort:
0
0
4
3
1
After sort:
0
0
1
3
4
```

Expected Output

```
Before Sort:
0
0
4
3
1
After sort:
0
0
1
3
4
```

N Queen's problem

Problem Submissions Leaderboard Discussions

Write a Java program to Implement N Queen's problem using Back Tracking.

Input Format

4

Constraints

No Constraints

Output Format

0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0

Sample Input 0

```
4
```

Sample Output 0

```
0  0  1  0
1  0  0  0
0  0  0  1
0  1  0  0
```


Java 7

```java
import java.util.*;
public class NQueenBacktracking
{
    int n;
    NQueenBacktracking(int n)
    {
        this.n = n;
    }

    /* Display solution*/
    void displaySolution(int queenBoard[][])
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
                System.out.print(" " + queenBoard[i][j] + " ");
                System.out.println();
        }
    }

    /* isSafe() function check if a queen can be placed on queenBoard[row][col]. */
    boolean isSafe(int queenBoard[][], int row, int col)
    {
        int i, j;
        /* for row on left side */
```

```
        for (i = 0; i < col; i++)
            if (queenBoard[row][i] == 1)
                return false;
         /* for upper diagonal on left side */
        for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
            if (queenBoard[i][j] == 1)
                return false;
         /* for lower diagonal on left side */
        for (i = row, j = col; j >= 0 && i < n; i++, j--)
            if (queenBoard[i][j] == 1)
                return false;
        return true;
    }

    /*  Utility function for N Queen problem solution */
    boolean utilityFunctionNQueen(int queenBoard[][], int col)
    {
    /* base case when all queens are placed */
       if (col >= n)
           return true;
        /* for this column try placing this queen in all rows one by one */
        for (int i = 0; i < n; i++)
        {
        /* Check is it safe at queenBoard[i][col] */
            if (isSafe(queenBoard, i, col))
            {
                /* Place this queen in board[i][col] */
                queenBoard[i][col] = 1;

                /* recurence to place rest of the queens */
                if (utilityFunctionNQueen(queenBoard, col + 1) == true)
                    return true;

                /* Backtrack: If solution doesn't achieved then remove queen from queenBoard[i]
[col] */             queenBoard[i][col] = 0;

            }

        }

        /* If we cannot place queen in any row in this column col, then return false */
          return false;
    }

    /* uses solveNQUtil () to solve the problem. Note that there may be more than one
    /* solutions, this function prints one of the feasible solutions.*/
    boolean mainSolutionNQueen()
    {
        int queenBoard[][] = new int[n][n];
        if (utilityFunctionNQueen(queenBoard, 0) == false)
        {
            System.out.print("Solution does not exist");
            return false;
        }
        displaySolution(queenBoard);
        return true;
    }

    // Driver main method
    public static void main(String args[])
    {
        int n;
        //System.out.print("Enter size of queen board i.e. N: ");
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        NQueenBacktracking queen = new NQueenBacktracking(n);
        queen.mainSolutionNQueen();
    }
}
```

Upload Code as File | Test against custom input | Run Code | Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
4
```

Your Output (stdout)

```
0  0  1  0
1  0  0  0
0  0  0  1
0  1  0  0
```

Expected Output

```
0  0  1  0
1  0  0  0
0  0  0  1
0  1  0  0
```