

[All Contests](#) > [DAA\\_LAB](#) > Knapsack-greedy\_method

# Knapsack-greedy\_method

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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: 55.33333333333336 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: 55.33333333333336  
The objects picked up into knapsack are:  
1.0  
1.0  
1.0  
1.0  
0.6666666666666666  
0.0
```

[f](#) [t](#) [in](#)**Contest ends in 9 days****Submissions: 100****Max Score: 10****Difficulty: Medium****Rate This Challenge:**

```
1 import java.util.Scanner;
2 class GKnapsack
3 {
4     int n;
5     double c;
6     double p[];
7     double w[];
8     public GKnapsack(int n,double c,double[] p,double[] w)
9     {
10         super();
11         this.n=n;
12         this.c=c;
13         this.p=p;
14         this.w=w;
15     }
16     void compute()
17     {
18         int i;
19         double[] x=new double[n+1];
20         for(i=0;i<n;i++)
21         {
22             x[i]=0.0;
23
24         }
25         double rc=c;
26         for(i=0;i<n;i++)
27         {
28             if(w[i]>rc)break;
29             x[i]=1;
30             rc=rc-w[i];
31         }
32         if(i<=n)
33         {
34             x[i]=rc/w[i];
35         }
36         double netProfit=0.0;
37         for(i=0;i<n;i++)
38         {
39             if(x[i]>0.0)
40             {
41                 netProfit=netProfit+x[i]*p[i];
42             }
43         }
44         System.out.println("Net Profit: "+netProfit);
45         System.out.println("The objects picked up into knapsack are:");
46         for(i=0;i<n;i++)
47         {
48             System.out.println(x[i]+" ");
49         }
50     }
51 }
52 public class KpGreedy
53 {
54     public static void main(String[] args)
55     {
56         int n;
57         double c;
58         Scanner input=new Scanner(System.in);
59         //System.out.println("Enter number of objects");
60         n=input.nextInt();
61         double[] p=new double[n+1];
62         double[] w=new double[n+1];
63         int i;
64         //System.out.println("Enter capacity of Knapsack");
65         c=input.nextDouble();
66         //System.out.println("Enter profit for each "+n+" objects");
67         for(i=0;i<n;i++)
68             p[i]=input.nextDouble();
69         //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

Test against custom input

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: 55.33333333333336
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: 55.33333333333336
The objects picked up into knapsack are:
1.0
1.0
1.0
1.0
1.0
0.6666666666666666
0.0
```

[All Contests](#) > [DAA\\_LAB](#) > Inorder Traversal 5

# Inorder Traversal 5

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

Write a java program to perform Inorder tree traversal.

[f](#) [t](#) [in](#)**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

Contest ends in 9 days

Submissions: 109

Max Score: 10

Difficulty: Medium

Rate This Challenge:



[More](#)

Java 7

v



```
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) {
```

```

18     this.node = b;
19     this.next = null;
20 }
21
22 public class NonRecursiveInorder {
23     StackNode top;
24     Node root;
25     public void NonRecursiveInorder() {
26         top = null;
27         root = null;
28     }
29     boolean isEmpty() {
30         if(top == null) {
31             return true;
32         }
33         return false;
34     }
35     void push(Node b) {
36         StackNode temp;
37         temp = new StackNode();
38         if(temp == null) {
39             System.out.printf("Stack is overflow.\n");
40         } else {
41             temp.node = b;
42             temp.next = top;
43             top = temp;
44         }
45     }
46     Node peek() {
47         if (top == null) {
48             return null;
49         }
50         return top.node;
51     }
52     Node pop() {
53         StackNode temp;
54         Node b;
55         if(top == null) {
56             System.out.printf("Stack is underflow.\n");
57             return null;
58         } else {
59             temp = top;
60             top = top.next;
61             b = temp.node;
62             return b;
63         }
64     }
65     void inorderInBST(Node root) {
66         Node curr = root;
67         while(true) {
68             if(curr!= null) {
69                 push(curr);
70                 curr = curr.left;
71             } else {
72                 curr = pop();
73                 System.out.printf("%d ",curr.data);
74                 curr = curr.right;
75             }
76             if(isEmpty() && curr == null)
77                 break;
78         }
79     }
80     /* Insertion into binary search tree */
81     Node insertBinarySearchTree(Node root, int item) {
82
83         /* If the tree is empty new node became root */
84         if (root == null) {
85             root = new Node(item);
86             return root;
87         }
88
89         /* Otherwise, if item is less then root then recur left side */

```

```

91     if (item < root.data)
92         root.left = insertBinarySearchTree(root.left, item);
93     else if (item > root.data)
94         root.right = insertBinarySearchTree(root.right, item);
95
96     /* return the root node pointer */
97     return root;
98 }
99
100 // Driver main method Code
101 public static void main(String[] args) {
102     NonRecursiveInorder tree = new NonRecursiveInorder();
103     Scanner sc = new Scanner(System.in);
104     int option;
105     int item;
106     //System.out.println("Enter 1 to insert\nEnter 2 to display BST in inorder\nEnter 3 to
107     //Exit");
108     while(true) {
109         //System.out.print("Enter your option: ");
110         option = sc.nextInt();
111         switch(option) {
112             default:
113                 System.out.println("Enter the right option");
114                 break;
115             case 1:
116                 //System.out.print("Enter the element to insert: ");
117                 item = sc.nextInt();
118                 tree.root = tree.insertBinarySearchTree(tree.root, item);
119                 break;
120             case 2:
121                 if(tree.root == null) {
122                     System.out.println("Tree is empty, root is null");
123                 } else {
124                     System.out.println("InorderTraversal is:");
125                     tree.inorderInBST(tree.root);
126                     System.out.println();
127                 }
128                 break;
129             case 3:
130                 return;
131         }
132     }
133 }

```

Line: 1 Col: 1

Test against custom input

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
```

[All Contests](#) > [DAA\\_LAB](#) > Dijkstra's Algorithm 4

# Dijkstra's Algorithm 4

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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
```

[f](#) [t](#) [in](#)**Contest ends in 9 days****Submissions: 98****Max Score: 10****Difficulty: Medium****Rate This Challenge:**[More](#)

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) {
```

```

7     Scanner scanner = new Scanner(System.in);
8     int n = scanner.nextInt(); // Number of vertices
9     int[][] graph = new int[n][n];
10
11    // Taking input for the weighted graph
12    for (int i = 0; i < n; i++) {
13        for (int j = 0; j < n; j++) {
14            graph[i][j] = scanner.nextInt();
15        }
16    }
17
18    int source = scanner.nextInt(); // Source vertex
19    dijkstra(graph, source, n);
20    scanner.close();
21}
22
23 private static void dijkstra(int[][] graph, int source, int n) {
24     boolean[] visited = new boolean[n];
25     int[] distance = new int[n];
26     Arrays.fill(distance, INF);
27     distance[source] = 0;
28
29     for (int i = 0; i < n - 1; i++) {
30         int u = minDistance(distance, visited, n);
31         visited[u] = true;
32
33         for (int v = 0; v < n; v++) {
34             if (!visited[v] && graph[u][v] != 0 && distance[u] != INF &&
35                 distance[u] + graph[u][v] < distance[v]) {
36                 distance[v] = distance[u] + graph[u][v];
37             }
38         }
39     }
40
41     // Printing the shortest paths
42     System.out.println("Shortest path from " + source + " to all other vertices");
43     for (int i = 0; i < n; i++) {
44         System.out.println("To " + i + " is " + distance[i]);
45     }
46 }
47
48 private static int minDistance(int[] distance, boolean[] visited, int n) {
49     int min = INF, minIndex = -1;
50     for (int i = 0; i < n; i++) {
51         if (!visited[i] && distance[i] <= min) {
52             min = distance[i];
53             minIndex = i;
54         }
55     }
56     return minIndex;
57 }
58 }
59

```

Line: 1 Col: 1

Test against custom input

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
```

[All Contests](#) > [DAA\\_LAB](#) > Postorder Traversal 1

# Postorder Traversal 1

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

Write a java program to perform postorder tree traversal.

[f](#) [t](#) [in](#)**Input Format**

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

**Constraints**

No Constraints

**Output Format**

Postorder Traversal 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

Contest ends in 9 days

Submissions: 111

Max Score: 10

Difficulty: Medium

Rate This Challenge:



[More](#)

Java 7

v



```
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) {
```

```

18     this.node = b;
19     this.next = null;
20 }
21
22 public class NonRecursivePostorder {
23     StackNode top;
24     Node root;
25     public void NonRecursivePostorder() {
26         top = null;
27         root = null;
28     }
29     boolean isEmpty() {
30         if(top == null) {
31             return true;
32         }
33         return false;
34     }
35     void push(Node b) {
36         StackNode temp;
37         temp = new StackNode();
38         if(temp == null) {
39             System.out.printf("Stack is overflow.\n");
40         } else {
41             temp.node = b;
42             temp.next = top;
43             top = temp;
44         }
45     }
46     Node peek() {
47         if (top == null) {
48             return null;
49         }
50         return top.node;
51     }
52     Node pop() {
53         StackNode temp;
54         Node b;
55         if(top == null) {
56             System.out.printf("Stack is underflow.\n");
57             return null;
58         } else {
59             temp = top;
60             top = top.next;
61             b = temp.node;
62             return b;
63         }
64     }
65     void postorderInBST(Node root) {
66     do {
67         while(root != null) {
68             if(root.right != null) {
69                 push(root.right);
70             }
71             push(root);
72             root = root.left;
73         }
74         root = pop();
75         if(root.right != null && peek() == root.right) {
76             pop();
77             push(root);
78             root = root.right;
79         } else {
80             System.out.printf("%d ",root.data);
81             root = null;
82         }
83     } while(!isEmpty());
84 }
85 /* Insertion into binary search tree */
86 Node insertBinarySearchTree(Node root, int item) {
87
88     /* If the tree is empty new node became root */
89     if (root == null) {

```

```

91         root = new Node(item);
92     }
93
94     /* Otherwise, if item is less than root then recur left side */
95     if (item < root.data)
96         root.left = insertBinarySearchTree(root.left, item);
97     else if (item > root.data)
98         root.right = insertBinarySearchTree(root.right, item);
99
100    /* return the root node pointer */
101   return root;
102 }
103
104 // Driver main method Code
105 public static void main(String[] args) {
106     NonRecursivePostorder tree = new NonRecursivePostorder();
107     Scanner sc = new Scanner(System.in);
108     int option;
109     int item;
110     //System.out.println("Enter 1 to insert\nEnter 2 to display BST in postorder\nEnter 3 to
111     Exit");
112     while(true) {
113         //System.out.print("Enter your option: ");
114         option = sc.nextInt();
115         switch(option) {
116             default:
117                 System.out.println("Enter the right option");
118                 break;
119             case 1:
120                 //System.out.print("Enter the element to insert: ");
121                 item = sc.nextInt();
122                 tree.root = tree.insertBinarySearchTree(tree.root, item);
123                 break;
124             case 2:
125                 if(tree.root == null) {
126                     System.out.println("Tree is empty, root is null");
127                 } else {
128                     System.out.println("Postorder Traversal is:");
129                     tree.postorderInBST(tree.root);
130                     System.out.println();
131                 }
132                 break;
133             case 3:
134                 return;
135             }
136         }
137     }
138
139
140
141
142
143
144
145
146
147
148
149

```

Line: 1 Col: 1

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 Contests](#) > [DAA\\_LAB](#) > [All\\_pair\\_Shortest\\_path](#)

# All\_pair\_Shortest\_path

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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

[f](#) [t](#) [in](#)**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

**Contest ends in 9 days****Constraints**

no constraints

**Submissions: 95****Max Score: 10****Difficulty: Medium****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

**Rate This Challenge:****Sample Input 0**[More](#)

```
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++)
```

```

20     {
21         System.out.print(cost[i][j]+" ");
22     }
23     System.out.println();
24 }
25 }
26 public int min(int i,int j)
27 {
28     if(i<j)
29         return i;
30     else
31         return j;
32 }
33 public static void main(String[] args)
34 {
35     Scanner sc=new Scanner(System.in);
36     //System.out.println("Enter the no of vertices: ");
37     n=sc.nextInt();
38     int cost[][]=new int[n+1][n+1];
39     //System.out.println("Enter the cost matrix:");
40     for(i=1;i<=n;i++)
41         for(j=1;j<=n;j++)
42             cost[i][j]=sc.nextInt();
43     Floyds f = new Floyds();
44     f.floyd(n,cost);
45 }
46 }
47

```

Line: 1 Col: 1

Test against custom input

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

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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 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  
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
```

Contest ends in 9 days

Submissions: 122

Max Score: 10

Difficulty: Medium

Rate This Challenge:



[More](#)

Java 7



```

1 import java.io.*;
2 import java.util.*;
3 import java.text.*;
4 import java.math.*;
5 import java.util.regex.*;
6
7 public class Solution {
8     static int top = -1;
9     static int arr[] = new int[50];
10    public void push(int num, int ele) {
11        if (top == num - 1) {
12            System.out.println("Stack is overflow");
13        } else {
14            top++;
15            arr[top] = ele;
16            System.out.println("pushed element "+ele);
17        }
18    }
19    public void pop() {
20        if (top == -1) {
21            System.out.println("Stack is underflow");
22        } else {
23            System.out.println("Popped element " + arr[top]);
24            top--;
25        }
26    }
27    public void display(int[] arr2, int num) {
28        if (top < 0) {
29            System.out.print("Stack Empty");
30        } else {
31            System.out.print("ELEMENTS : ");
32            for (int i = top; i >= 0; i--) {
33                System.out.print(arr2[i] + " ");
34            }
35        }
36        System.out.println();
37    }
38    public static void main(String[] args) {
39        Solution su = new Solution();
40        Scanner sc = new Scanner(System.in);
41        int num, opt;
42        int ele;
43        num = sc.nextInt();
44        Boolean kl = true;
45        while (kl) {
46            opt = sc.nextInt();
47            switch (opt) {
48                case 1:
49                    ele = sc.nextInt();
50                    su.push(num, ele);
51                    break;
52                case 2:
53                    su.pop();
54                    break;
55                case 3:
56                    su.display(arr, num);
57                    break;
58                case 4:
59                    kl = false;
60                    break;
61                default:
62                    break;
63            }
64        }
65    }
66 }

```

Line: 1 Col: 1

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

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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
```

Contest ends in 9 days

Submissions: 103

Max Score: 10

Difficulty: Medium

Rate This Challenge:



[More](#)

```
1 import java.util.Scanner;
2 class QuickSort {
3     private int a[];
4     public QuickSort(int[] a)
5     {
6         this.a = a;
7     }
8     public int partition ( int a[], int m, int p ) {
9         int v = a[m];
10        int i = m;
11        int j = p;
12        do {
13            while ( a[++ i] < v);
14            while ( a[-- j] > v );
15            if ( i < j )
16                interchange ( a, i, j );
17
18        }
19        while ( i <= j );
20        a[m] = a[j]; a[j] = v;
21        return j;
22    }
23    public void qSort ( int p, int q ) {
24        int j;
25        if ( p < q ) {
26            j = partition ( a, p, q + 1 );
27            qSort ( p, j - 1 );
28            qSort ( j + 1, q );
29
30        }
31    }
32    public void interchange ( int a[], int i, int j ) {
33        int t;
34        t = a[i];
35        a[i] = a[j];
36        a[j] = t;
37    }
38 }
39 public class QuickSortDemo {
40     public static void main(String[] args) {
41         int n, a[], i;
42         Scanner input = new Scanner(System.in);
43         //System.out.print("Enter the Size of an Array: ");
44         n = input.nextInt();
45         a = new int[n + 1];
46         //System.out.println("System automatically generates numbers ");
47         for ( i = 0; i < n; ++ i ){
48             a[i] = input.nextInt();
49         }
50         a[i] = 100000; //Sentinel value
51         QuickSort qSort = new QuickSort(a);
52         System.out.println("Before Sort: ");
53         for ( i = 0; i < n; ++ i ) {
54             System.out.print(a[i] + "\n");
55         }
56         int p = 0;
57         int q = n - 1;
58         qSort.qSort(p, q);
59         System.out.println("After sort: ");
60         for ( i = 0; i < n; ++ i ) {
61             System.out.print(a[i] + "\n");
62         }
63     }
64 }
```

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
```

[All Contests](#) > [DAA\\_LAB](#) > [Knapsack-Dynamic\\_programming](#)

# Knapsack-Dynamic\_programming

<a href="#">Problem</a>	<a href="#">Submissions</a>	<a href="#">Leaderboard</a>	<a href="#">Discussions</a>
-------------------------	-----------------------------	-----------------------------	-----------------------------

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

```
1 import java.util.Scanner;  
2 class DKnapsack {  
3     int c,n,p[],w[],v[][];  
4     public DKnapsack(int n,int c,int[] p,int[] w)  
5     {  
6         super();  
7         this.n=n;  
8         this.c=c;  
9         this.w=w;  
10        this.p=p;  
11        this.v=new int[n+1][c+1];  
12    }  
13    void compute()  
14    {  
15        for(int i=0;i<=n;++i){  
16            for(int j=0;j<=c;++j){  
17                if(i==0||j==0){
```

Contest ends in 9 days

Submissions: 99

Max Score: 10

Difficulty: Medium

Rate This Challenge:



[More](#)

```

18         v[i][j]=0;
19     }
20     else if(j-w[i]>=0)
21     {
22         v[i][j]=max(v[i-1][j],p[i]+v[i-1][j-w[i]]);
23     }
24     else if(j-w[i]<0)
25     {
26         v[i][j]=v[i-1][j];
27     }
28 }
29 System.out.println("Optimal Solution: "+v[n][c]);
30 traceback();
31 }
32 void traceback(){
33     System.out.println("The objects picked up into knapsack are:");
34     int i=n;
35     int j=c;
36     while(i>0)
37     {
38         if(v[i][j]!=v[i-1][j])
39         {
40             System.out.print(i+" ");
41             j=j-w[i];
42             i--;
43         }
44         else {
45             i--;
46         }
47     }
48 }
49 private int max(int i,int j){
50     if(i>j) return i;
51     else return j;
52 }
53 }
54 }
55 public class KpDynamic{
56     public static void main(String[] args){
57         int c,n;
58         Scanner input=new Scanner(System.in);
59         //System.out.println("Enter number of objects");
60         n=input.nextInt();
61         int[] p=new int[n+1];
62         int[] w=new int[n+1];
63         int i;
64         //System.out.println("Enter capacity of Knapsack");
65         c=input.nextInt();
66         //System.out.println("Enter profit for each "+n+" objects");
67         for(i=1;i<=n;i++)
68         {
69             p[i]=input.nextInt();
70             //System.out.println("Enter weight for each "+n+" objects");
71             for(i=1;i<=n;i++)
72                 w[i]=input.nextInt();
73             DKnapsack dk=new DKnapsack(n,c,p,w);
74             dk.compute();
75         }
76     }

```

Line: 1 Col: 1

Test against custom input

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
```

[All Contests](#) > [DAA\\_LAB](#) > [Breadth first search\(BFS\) 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 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

[f](#) [t](#) [in](#)**Contest ends in 9 days****Submissions: 117****Max Score: 10****Difficulty: Medium****Rate This Challenge:**[More](#)

Java 7



```
1 import java.io.*;  
2 import java.util.*;
```

```

3 import java.text.*;
4 import java.math.*;
5 import java.util.regex.*;
6 class Gnode {
7     Gnode next;
8     int vertex;
9 }
10 public class Solution {
11     Gnode graph[ ];
12     boolean visited[ ];
13     int queue[];
14     int numVertices;
15     int front;
16     int rear;
17     public Solution(int n) {
18         graph = new Gnode[n];
19         visited = new boolean[n];
20         queue = new int[n];
21         numVertices = n;
22         front = -1;
23         rear = -1;
24     }
25     void insertQueue(int vertex) {
26         if(rear == numVertices - 1){}
27         //System.out.printf("Queue Overflow.\n");
28         else {
29             if(front == -1)
30                 front = 0;
31             rear = rear + 1;
32             queue[rear] = vertex ;
33         }
34     }
35 }
36
37     boolean isEmptyQueue() {
38         if(front == -1 || front > rear)
39             return true;
40         else
41             return false;
42     }
43
44     int deleteQueue() {
45         int deleteItem;
46         if(isEmptyQueue()) {
47             //System.out.printf("Queue Underflow\n");
48             return -1;
49         }
50         deleteItem = queue[front];
51         front = front + 1;
52         return deleteItem;
53     }
54     void Bfs(int v) {
55         int w;
56         insertQueue(v);
57         while(!isEmptyQueue()) {
58             v = deleteQueue( );
59             System.out.printf(" %d",v);
60             visited[v] = true;
61             Gnode g = graph[v];
62             for( ; g != null; g = g.next) {
63                 w = g.vertex;
64                 if(visited[w] == false) {
65                     insertQueue(w);
66                     visited[w] = true;
67                 }
68             }
69         }
70     }
71     public static void main(String []args) {
72         int n, e, i, s, d, v;
73         Gnode q, p;
74         Scanner sc = new Scanner(System.in);
75

```

```

76     //System.out.printf("Enter the number of vertices : ");
77     n = sc.nextInt();
78     //System.out.printf("Enter the number of edges : ");
79     e = sc.nextInt();
80     Solution g = new Solution(n);
81     for(i=1;i<=e;i++) {
82         //System.out.printf("Enter source : ");
83         s = sc.nextInt();
84         //System.out.printf("Enter destination : ");
85         d = sc.nextInt();
86         q = new Gnode();
87         q.vertex = d;
88         q.next = null;
89         if(g.graph[s] == null)
90             g.graph[s]=q;
91         else {
92             p=g.graph[s];
93             while(p.next != null)
94                 p = p.next;
95             p.next = q;
96         }
97     }
98     for(i = 0;i < n;i++)
99         g.visited[i] = false;
100    //System.out.printf("Enter Start Vertex for BFS : ");
101    v = sc.nextInt();
102    System.out.printf("BFS of graph :");
103    g.Bfs(v);
104    System.out.printf("\n");
105}
106}
107
108
109
110
111
112
113
114

```

Line: 1 Col: 1

Test against custom input

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

[All Contests](#) > [DAA\\_LAB](#) > Preorder Traversal 4

# Preorder Traversal 4

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

Write a java program to perform preorder tree order tree traversal

[f](#) [t](#) [in](#)**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

Contest ends in 9 days

Submissions: 113

Max Score: 10

Difficulty: Medium

Rate This Challenge:



[More](#)

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;
```

```

20     }
21
22 public class NonRecursivePreorder {
23     StackNode top;
24     Node root;
25     public void NonRecursivePreorder() {
26         top = null;
27         root = null;
28     }
29     boolean isEmpty() {
30         if(top == null) {
31             return true;
32         }
33         return false;
34     }
35     void push(Node b) {
36         StackNode temp;
37         temp = new StackNode();
38         if(temp == null) {
39             System.out.printf("Stack is overflow.\n");
40         } else {
41             temp.node = b;
42             temp.next = top;
43             top = temp;
44         }
45     }
46     Node peek() {
47         if (top == null) {
48             return null;
49         }
50         return top.node;
51     }
52     Node pop() {
53         StackNode temp;
54         Node b;
55         if(top == null) {
56             System.out.printf("Stack is underflow.\n");
57             return null;
58         } else {
59             temp = top;
60             top = top.next;
61             b = temp.node;
62             return b;
63         }
64     }
65     void preorderInBST(Node root) {
66         Node curr = root;
67         push(root);
68         while(true) {
69             curr = pop();
70             System.out.printf("%d ",curr.data);
71             if(curr.right != null) {
72                 push(curr.right);
73             }
74             if(curr.left != null) {
75                 push(curr.left);
76             }
77             if(isEmpty())
78                 break;
79         }
80     }
81 /* Insertion into binary search tree */
82     Node insertBinarySearchTree(Node root, int item) {
83
84         /* If the tree is empty new node became root */
85         if (root == null) {
86             root = new Node(item);
87             return root;
88         }
89
90         /* Otherwise, if item is less then root then recur left side */
91         if (item < root.data)

```

```

    root.left = insertBinarySearchTree(root.left, item);
93   else if (item > root.data)
94     root.right = insertBinarySearchTree(root.right, item);
95
96   /* return the root node pointer */
97   return root;
98 }
99
100 // Driver main method Code
101 public static void main(String[] args) {
102   NonRecursivePreorder tree = new NonRecursivePreorder();
103   Scanner sc = new Scanner(System.in);
104   int option;           int item;
105   //System.out.println("Enter 1 to insert\nEnter 2 to display BST in preorder\nEnter 3 to
Exit");
106   while(true) {
107     //System.out.print("Enter your option: ");
108     option = sc.nextInt();
109     switch(option) {
110       default:
111         System.out.println("Enter the right option");
112         break;
113       case 1:
114         //System.out.print("Enter the element to insert: ");
115         item= sc.nextInt();
116         tree.root = tree.insertBinarySearchTree(tree.root,item);
117         break;
118       case 2:
119         if(tree.root == null) {
120           System.out.println("Tree is empty, root is null");
121         }else {
122
123           System.out.println("Preorder Traversal is:");
124           tree.preorderInBST(tree.root);
125           System.out.println();
126
127         }
128         break;
129       case 3:
130         return;
131     }
132   }
133 }
134 }
```

Line: 1 Col: 1

Test against custom input

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

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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 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
```

[f](#) [t](#) [in](#)

Contest ends in 9 days

Submissions: 97

Max Score: 10

Difficulty: Medium

Rate This Challenge:



[More](#)

Java 7



```
1 import java.util.Scanner;
```

```

2 | public class Kruskals
3 | {
4 |     static int parent[],cost[][] , mincost,n,i,j,ne,a,b,min,u,v;
5 |     public void kruskal(int n,int[][] cost)
6 |     {
7 |         ne=1;
8 |         while(ne<n)
9 |         {
10 |             min=999;
11 |             for(i=1;i<=n;i++)
12 |             {
13 |                 for(j=1;j<=n;j++)
14 |                     if(cost[i][j]<min)
15 |                     {
16 |                         min=cost[i][j];
17 |                         a=u=i;
18 |                         b=v=j;
19 |                     }
20 |             }
21 |             u=find(u);
22 |             v=find(v);
23 |             if(v!=u)
24 |             {
25 |                 System.out.println( ne+"edge("+a+","+b+")=="+min);
26 |                 ne=ne+1;
27 |                 mincost=mincost+min;
28 |                 uni(u,v);
29 |             }
30 |             cost[a][b]=cost[b][a]=999;
31 |         }
32 |         System.out.println("The minimum cost of spanning tree is "+mincost);
33 |     }
34 |     public int find (int i)
35 |     {
36 |         while (parent[i] != 0)
37 |             i=parent[i];
38 |         return i;
39 |     }
40 |     public void uni(int i,int j)
41 |     {
42 |         parent[j]=i;
43 |     }
44 |     public static void main(String[] args)
45 |     {
46 |         Scanner sc=new Scanner(System.in);
47 |         //System.out.println("Enter the number of vertices: ");
48 |         n=sc.nextInt();
49 |         int cost[][]= new int [n+1][n+1];
50 |         parent=new int[n+1];
51 |         //System.out.println("Enter the cost matrix:");
52 |         for(i=1;i<=n;i++)
53 |         {
54 |             for(j=1;j<=n;j++)
55 |             {
56 |                 cost[i][j]=sc.nextInt();
57 |                 if(cost[i][j]==0)
58 |                     cost[i][j]=999;
59 |             }
60 |         }
61 |         Kruskals k = new Kruskals();
62 |         k.kruskal(n,cost);
63 |     }
64 |
65 |

```

Line: 1 Col: 1

Test against custom input

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
```

[All Contests](#) > [DAA\\_LAB](#) > Depth First Search(DFS)

# Depth First Search(DFS)

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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

[f](#) [t](#) [in](#)**Contest ends in 9 days****Submissions: 115****Max Score: 10****Difficulty: Medium****Rate This Challenge:**[More](#)

Java 7



```
1 import java.io.*;  
2 import java.util.*;  
3 import java.text.*;
```

```

5 import java.math.*;
5 import java.util.regex.*;
6 ▼class Gnode {
7     Gnode next;
8     int vertex;
9
10 }
11 ▼public class Solution {
12     Gnode graph[];
13     boolean visited[];
14     int numVertices;
15     public Solution(int n) {
16         this.numVertices = n;
17         visited = new boolean[n];
18         graph = new Gnode[n];
19
20     }
21     void Dfs(int i) {
22         Gnode p;
23         System.out.printf(" %d",i);
24         p = graph[i];
25         visited[i] = true;
26         while(p != null) {
27             i = p.vertex;
28             if(visited[i] == false)
29                 Dfs(i);
30             p = p.next;
31         }
32     }
33 }
34
35 public static void main(String []args) {
36     int n, e, i, s, d, v;
37     Gnode q, p;
38     Scanner sc = new Scanner(System.in);
39     //System.out.printf("Enter the number of vertices : ");
40     n = sc.nextInt();
41     //System.out.printf("Enter the number of edges : ");
42     e = sc.nextInt();
43     Solution g = new Solution(n);
44     for(i=1;i<=e;i++) {
45         //System.out.printf("Enter source : ");
46         s = sc.nextInt();
47         //System.out.printf("Enter destination : ");
48         d = sc.nextInt();
49         q = new Gnode();
50         q.vertex=d;
51         q.next=null;
52         if(g.graph[s]==null)
53             g.graph[s]=q;
54         else {
55             p=g.graph[s];
56             while(p.next!=null)
57                 p=p.next;
58             p.next=q;
59         }
60     }
61     for(i=0;i<n;i++)
62         g.visited[i] = false;
63     //System.out.printf("Enter Start Vertex for DFS : ");
64     v = sc.nextInt();
65     System.out.printf("DFS of graph :");
66     g.Dfs(v);
67     System.out.printf("\n");
68
69 }
70 }
71

```

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
```

[All Contests](#) > [DAA\\_LAB](#) > [Travelling\\_Sales\\_person](#)

# Travelling\_Sales\_person

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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

[f](#) [t](#) [in](#)**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**

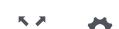
```
4
999 1 3 6
1 999 2 3
3 2 999 1
6 3 1 999
```

**Sample Output 0**

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

**Contest ends in 9 days****Submissions: 96****Max Score: 10****Difficulty: Medium****Rate This Challenge:**[More](#)

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,cost;
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]]+(cost[path[i]][path[n-1]]+cost[path[n-1]][1])<mincost)
20            {
21                mincost=cost[path[start]][path[i]]+cost[path[i]][path[n-1]]+cost[path[n-1]][1];
22                for(k=1;k<=n;k++)
23                    mintour[k]=temp[k];
24            }
25        }
26    }
27 }
```

```

25
26     }
27     for(i=1;i<=n;i++)
28     {
29         path[i]=mintour[i];
30     }
31     public static void main(String[] args)
32     {
33         int mincost,n,i,j;
34         Scanner s = new Scanner(System.in);
35         //System.out.println("enter the no of cities: ");
36         n=s.nextInt();
37         int path[] =new int[n+1];
38         cost = new int[n+1][n+1];
39         //System.out.println("Enter the cost matrix:");
40         for(i=1;i<=n;i++)
41         {
42             for(j=1;j<=n;j++)
43             {
44                 cost[i][j]=s.nextInt();
45             }
46         }
47         path[1]=1;
48         Tsp obj = new Tsp();
49         mincost=obj.tsp(path,1,n);
50         System.out.println("tsp tour");
51         for(i=1;i<=n;i++)
52         {
53             System.out.print(path[i] + "---->");
54         }
55         System.out.println("1");
56         System.out.println("Tourcost=" + mincost);
57     }
58 }
```

Line: 1 Col: 1

Test against custom input

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

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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
```

Contest ends in 9 days

Submissions: 119

Max Score: 10

Difficulty: Medium

Rate This Challenge:



[More](#)

```
1 import java.io.*;
2 import java.util.*;
3 import java.text.*;
4 import java.math.*;
5 import java.util.regex.*;
6
7 public class Solution {
8     static int front = -1, rear = -1;
9     static int arr[] = new int[50];
10    public void insert(int num, int ele) {
11        if (rear == num - 1) {
12            System.out.println("Queue is overflow");
13        } else {
14            rear++;
15            arr[rear] = ele;
16            System.out.println("Inserted Element is "+ele);
17        }
18        if (front == -1) {
19            front++;
20        }
21    }
22    public void delete() {
23        if (front == -1) {
24            System.out.println("Queue is underflow");
25        } else {
26            System.out.println("Dequeued Element is "+ arr[front]);
27        if (front == rear) {
28            front = rear = -1;
29        } else {
30            front++;
31        }
32    }
33}
34    public void display(int[] arr, int num) {
35        if (rear == -1 && front == -1) {
36            System.out.print("Queue is Empty");
37        } else {
38            System.out.print("ELEMENTS : ");
39        for (int i = front; i <= rear; i++) {
40            System.out.print(arr[i] + " ");
41        }
42    }
43    System.out.println();
44}
45    public static void main(String[] args) {
46        Solution qe = new Solution();
47        Scanner sc = new Scanner(System.in);
48        int num, opt;
49        int ele;
50        num = sc.nextInt();
51        Boolean kl = true;
52        while (kl) {
53            opt = sc.nextInt();
54            switch (opt) {
55                case 1:
56                    ele = sc.nextInt();
57                    qe.insert(num, ele);
58                    break;
59                case 2:
60                    qe.delete();
61                    break;
62                case 3:
63                    qe.display(arr, num);
64                    break;
65                case 4:
66                    kl = false;
67                    break;
68                default :
69                    break;
70            }
71        }
72    }
73}
```

```
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  
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

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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
```

Contest ends in 9 days

Submissions: 103

Max Score: 10

Difficulty: Medium

Rate This Challenge:



[More](#)

```
1 import java.util.Scanner;
2 class MergeSort {
3     private int a[];
4     public MergeSort(int[] a) {
5         this.a = a;
6     }
7     void merge ( int low, int mid, int high ) {
8         int b[] = new int[high + 1];
9         int h = low;
10        int i = low;
11        int j = mid + 1;
12        int k;
13        while ( ( h <= mid ) && ( j <= high ) ) {
14            if ( a[h] <= a[j] ) b[i ++] = a[h ++];
15            else b[i ++] = a[j ++];
16        }
17        if ( h > mid) {
18            for ( k = j; k <= high; ++ k )
19                b[i ++] = a[k];
20        }
21        else {
22            for ( k = h; k <= mid; ++ k )
23                b[i ++] = a[k];
24        }
25        for (k=low; k<= high; ++ k)
26            a[k] = b[k];
27    }
28    void mergeSort ( int low, int high ) {
29        int mid;
30        if ( low < high ) {
31            mid = ( low + high ) / 2;
32            mergeSort ( low, mid );
33            mergeSort ( mid + 1, high );
34            merge ( low, mid, high );
35        }
36    }
37 }
38 public class MergeSortDemo {
39     public static void main(String[] args) {
40         int n, a[], i;
41         Scanner input = new Scanner(System.in);
42         //System.out.println("Enter the Size of an Array: ");
43         n = input.nextInt();
44         a = new int[n + 1];
45         //System.out.println("System automatically generates numbers ");
46         for ( i = 0; i < n; ++ i ) {
47             a[i] = input.nextInt();
48         }
49         a[i] = 100000;
50         MergeSort mSort = new MergeSort(a);
51         System.out.println("Before Sort: ");
52         for ( i = 0; i < n; ++ i ) {
53             System.out.print(a[i] + "\n");
54         }
55         int low = 0;
56         int high = n - 1;
57         mSort.mergeSort(low, high);
58         System.out.println("After sort: ");
59         for ( i = 0; i < n; ++ i ) {
60             System.out.print(a[i] + "\n");
61         }
62     }
63 }
64 }
```

Line: 1 Col: 1

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
```

[All Contests](#) > [DAA\\_LAB](#) > N Queen's problem

# N Queen's problem

[Problem](#)[Submissions](#)[Leaderboard](#)[Discussions](#)

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

[f](#) [t](#) [in](#)**Input Format**

4

**Constraints**

No Constraints

**Output Format**

0 0 1 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

**Contest ends in 9 days****Submissions: 90****Max Score: 10****Difficulty: Medium****Rate This Challenge:**[More](#)

Java 7



```
1 import java.util.*;
2 public class NQueenBacktracking
3 {
4     int n;
5     NQueenBacktracking(int n)
6     {
7         this.n = n;
8     }
9
10    /* Display solution*/
11    void displaySolution(int queenBoard[][])
12    {
13        for (int i = 0; i < n; i++)
14        {
15            for (int j = 0; j < n; j++)
16                System.out.print(" " + queenBoard[i][j] + " ");
17            System.out.println();
18        }
19    }
20
21    /* isSafe() function check if a queen can be placed on queenBoard[row][col]. */
22    boolean isSafe(int queenBoard[][], int row, int col)
23    {
24        int i, j;
25        /* for row on left side */
```

```

26         for (i = 0; i < col; i++)
27             if (queenBoard[row][i] == 1)
28                 return false;
29         /* for upper diagonal on left side */
30         for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
31             if (queenBoard[i][j] == 1)
32                 return false;
33         /* for lower diagonal on left side */
34         for (i = row, j = col; j >= 0 && i < n; i++, j--)
35             if (queenBoard[i][j] == 1)
36                 return false;
37     return true;
38 }
39
40 /* Utility function for N Queen problem solution */
41 boolean utilityFunctionNQueen(int queenBoard[][], int col)
42 {
43     /* base case when all queens are placed */
44     if (col >= n)
45         return true;
46     /* for this column try placing this queen in all rows one by one */
47     for (int i = 0; i < n; i++)
48     {
49         /* Check is it safe at queenBoard[i][col] */
50         if (isSafe(queenBoard, i, col))
51         {
52             /* Place this queen in board[i][col] */
53             queenBoard[i][col] = 1;
54
55             /* recurrence to place rest of the queens */
56             if (utilityFunctionNQueen(queenBoard, col + 1) == true)
57                 return true;
58
59             /* Backtrack: If solution doesn't achieved then remove queen from queenBoard[i]
60             [col] */
61             queenBoard[i][col] = 0;
62         }
63     }
64
65     /* If we cannot place queen in any row in this column col, then return false */
66     return false;
67 }
68
69 /* uses solveNQUtil () to solve the problem. Note that there may be more than one
70 /* solutions, this function prints one of the feasible solutions.*/
71 boolean mainSolutionNQueen()
72 {
73     int queenBoard[][] = new int[n][n];
74     if (utilityFunctionNQueen(queenBoard, 0) == false)
75     {
76         System.out.print("Solution does not exist");
77         return false;
78     }
79     displaySolution(queenBoard);
80     return true;
81 }
82
83 // Driver main method
84 public static void main(String args[])
85 {
86     int n;
87     //System.out.print("Enter size of queen board i.e. N: ");
88     Scanner sc = new Scanner(System.in);
89     n = sc.nextInt();
90     NQueenBacktracking queen = new NQueenBacktracking(n);
91     queen.mainSolutionNQueen();
92 }
93

```

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
```