

**Aim:** Write a java program to implement Dynamic Programming algorithm for the Optimal Binary Search Tree Problem

**Program:**

```
import java.util.Scanner;
public class obst
{
    static int find(int c[][],int i,int j){
    int min =1000,l=0;
        for(int k=i+1;k<=j;k++) {
            if((c[i][k-1]+c[k][j]) < min)
            {
                min = c[i][k-1]+c[k][j];
                l = k;
            }
        }
        return l;
    }
    public static void main(String[] args) {
    System.out.println("Enter the number of Identifiers: ");
    Scanner s = new Scanner(System.in);
    int n = s.nextInt();
    int p[] = new int[n+1];
    int q[] = new int[n+1];
    System.out.println("Enter the Probabilities p: ");
    for(int i=1;i<=n;i++){
        p[i] = s.nextInt();
    }
    System.out.println("Enter the Probabilities q: ");
    int c[][] = new int[n+1][n+1];
    int w[][] = new int[n+1][n+1];
    int r[][] = new int[n+1][n+1];
    for(int i=0;i<=n;i++){
        q[i] = s.nextInt();
        w[i][i] = q[i];
        c[i][i] = 0;
        r[i][i] = 0;
    }
    for(int i=0;i<=n;i++) {
        for(int j=1;j<=n;j++) {
            if(j!=i && j>i)
            {
                w[i][j] = p[j]+q[j]+w[i][j-1];
            }
        }
    }
    for(int m =1;m<=n;m++) {
        for(int i=0;i<n;i++) {
            for(int j=0;j<=n;j++)
            {
                if(j!=i && j>i && j-i==m)
                {
                    int k = find(c,i,j);
                    r[i][j] = k;
                }
            }
        }
    }
}
```

```

                c[i][j] = w[i][j]+c[i][k-1]+c[k][j];
            }

        }

    }

    for(int i=0;i<=n;i++)
    {
        for(int j=0;j<=n;j++)
        {
            if(j>=i)
            {
                System.out.print("j-i:");
                System.out.print(+j-i);
                System.out.print(" ");
                System.out.print("c("+i+", "+j+") = "+c[i][j] + " , ");
                System.out.print("w("+i+", "+j+") = "+w[i][j] + " , ");
                System.out.println("r("+i+", "+j+") = "+r[i][j] + " ");
            }
        }
        System.out.println();
    }
    System.out.println("The Root node of Optiaml Binary Search tree is: t"+r[0][n]);

    s.close();
}
}

```

**Output:**

@ Javadoc Declaration Console Call Hierarchy

<terminated> obst [Java Application] C:\Program Files\Java\jre1.8.0\_25\bin\javaw.exe (07-Dec-2021 1:09:05 pm)

---

Enter the number of Identifiers:

4

Enter the Probabilities p:

3 3 1 1

Enter the Probabilities q:

2 3 1 1 1

j-i:0 c(0,0) = 0 , w(0,0) = 2 , r(0,0) = 0  
j-i:1 c(0,1) = 8 , w(0,1) = 8 , r(0,1) = 1  
j-i:2 c(0,2) = 19 , w(0,2) = 12 , r(0,2) = 1  
j-i:3 c(0,3) = 25 , w(0,3) = 14 , r(0,3) = 2  
j-i:4 c(0,4) = 32 , w(0,4) = 16 , r(0,4) = 2

j-i:0 c(1,1) = 0 , w(1,1) = 3 , r(1,1) = 0  
j-i:1 c(1,2) = 7 , w(1,2) = 7 , r(1,2) = 2  
j-i:2 c(1,3) = 12 , w(1,3) = 9 , r(1,3) = 2  
j-i:3 c(1,4) = 19 , w(1,4) = 11 , r(1,4) = 2

j-i:0 c(2,2) = 0 , w(2,2) = 1 , r(2,2) = 0  
j-i:1 c(2,3) = 3 , w(2,3) = 3 , r(2,3) = 3  
j-i:2 c(2,4) = 8 , w(2,4) = 5 , r(2,4) = 3

j-i:0 c(3,3) = 0 , w(3,3) = 1 , r(3,3) = 0  
j-i:1 c(3,4) = 3 , w(3,4) = 3 , r(3,4) = 4

j-i:0 c(4,4) = 0 , w(4,4) = 1 , r(4,4) = 0

The Root node of Optiaml Binary Search tree is: t2

**Aim:** Write a Java program to implement Dynamic Programming algorithm for 0/1 Knapsack problem

**Program:**

```
import java.util.*;

public class knapsack01 {
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter the no of instances:");
        int n = s.nextInt();
        int p[] = new int[n+1];
        int w[] = new int[n+1];
        System.out.println("Enter the weights:");
        for(int i=1;i<=n;i++){
            w[i] = s.nextInt();
        }
        System.out.println("Enter the Profits:");
        for(int i=1;i<=n;i++){
            p[i] = s.nextInt();
        }
        System.out.println("Enter the maximum capacity");
        int m=s.nextInt();
        knapsack(w,p,m,n);
    }
    public static void knapsack(int W[],int P[],int M,int N)
    {
        int inf=Integer.MAX_VALUE;
        int[][] m=new int[N+1][M+1];
        int[][] s=new int[N+1][M+1];
        int i,j,n,w,tp=0,tw=0;
        for(i=1;i<=N;i++)
        {
            for(j=0;j<=M;j++)
            {
                int m1=m[i-1][j];
                int m2=inf;
                if(j<=W[i])
                {
                    m2 = m[i - 1][j - W[i]] + P[i];
                    m[i][j] = Math.max(m1, m2);
                    s[i][j] = m2 > m1 ? 1 : 0;
                }
            }
        }
        int[] selected = new int[N + 1];
        for ( n = N, w = M; n > 0; n--)
        {
            if (s[n][w] != 0)
            {
                selected[n] = 1;
                tw+=W[n];
                tp+=P[n];
            }
        }
    }
}
```

```

        w = w - W[n];
    }
    else
    {
        selected[n] = 0;
    }
}
System.out.print("Optimal sol is (");
for( i=1;i<N+1;i++)
{
    if(i!=N)
        System.out.print("x"+i+",");
    else
        System.out.print("x"+i+"): (");
}
for( i=1;i<N+1;i++)
{
    if(i!=N)
        System.out.print(selected[i]+",");
    else
        System.out.println(selected[i]+"");
}
System.out.println("Total profit obtained : "+tp);
System.out.println("Total weight obtained : "+tw);
}
}

```

#### Output:

@ Javadoc Declaration Console Call Hierarchy

<terminated> knapsack01 [Java Application] C:\Program Files\Java\jre1.8.0\_25\bin\javaw.e

Enter the no of instances:

3

Enter the weights:

1 2 3

Enter the Profits:

2 5 6

Enter the maximum capacity

5

Optimal sol is (x1,x2,x3): (0,1,1)

Total profit obtained : 11

Total weight obtained : 5

**Aim:** Write a java program to implement sum of subsets using backtracking Algorithm.

**Program:**

```
import java.util.*;
class sumofsubsets {
    static int[] w;
    static int[] x;
    static int m,n,c=0;
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        System.out.print("Enter the no. of weights:");
        n=s.nextInt();
        w = new int[n + 1];
        x= new int[n + 1];
        int t= 0,i;
        System.out.print("Enter " + n + " weights:");
        for(i=1;i<n+1;i++)
        {
            w[i]=s.nextInt();
            t+=w[i];
        }
        System.out.print("Enter the sum to be obtained:");
        m=s.nextInt();
        if (t<m || m<w[1])
        {
            System.out.println("Not possible to obtain the subset!!");
            System.exit(1);
        }
        System.out.println("The possible solutions are:");
        SumOfSub(0, 1, t);
        //if(c==0)
        //System.out.println(c);
    }


    static public void SumOfSub(int s, int k, int r)
    {
        int i=0,l;
        x[k]=1;
        if(s+w[k]==m)
        {
            c++;
            System.out.print(" "+c+".");
            for(l=1;l<x.length;l++)
            {
                if(l!=x.length-1)
                    System.out.print("x"+l+",");
                else
                    System.out.print("x"+l);
            }
            System.out.print(") = (");
        }
    }
}
```

```

        for(i=1;i<=k;i++)
        {
            if(i!=k)
                System.out.print(x[i]+",");
            else
                System.out.print(x[i]);
        }
        for(i=k+1;i<x.length;i++)
            System.out.print("0");
        System.out.print("\n");
    }
    else if((s+w[k]+w[k+1])<= m)
    {
        SumOfSub(s+w[k],k+1,r-w[k]);
    }
    if((s+r-w[k])>=m && (s+w[k+1])<=m)
    {
        x[k]=0;
        SumOfSub(s,k+1,r-w[k]);
    }
}
}

```

#### Output:


 <terminated> sumofsubsets [Java Application] C:\Program Files\Java\jre1.8.0\_101\bin\java.exe

```

Enter the no. of weights:4
Enter 4 weights:7 11 13 24
Enter the sum to be obtained:31
The possible solutions are:
1.(x1,x2,x3,x4) = (1,1,1,0)
2.(x1,x2,x3,x4) = (1,0,0,1)
  
```

**Aim:** Write a java program to implement all pairs shortest path using Floyd triangle

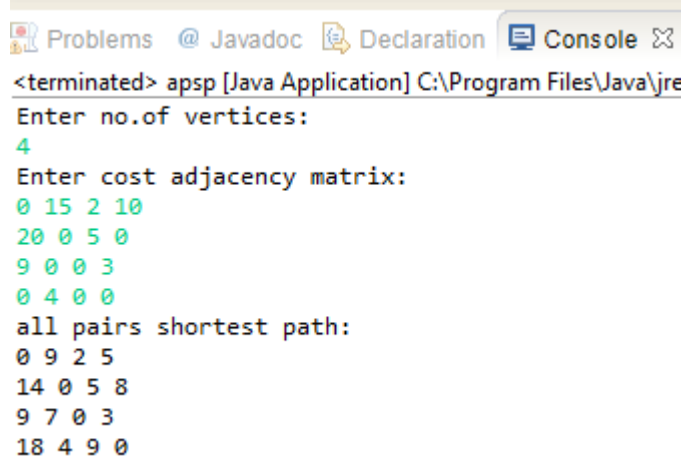
**Program:**

```
import java.util.*;
public class apsp {
    public static void main(String args[])
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter no.of vertices: ");
        int v=sc.nextInt();
        int[][] cost=new int[v][v];
        int inf=9999;
        int i,j,k;
        System.out.println("Enter cost adjacency matrix: ");
        for(i=0;i<v;i++)
        {
            for(j=0;j<v;j++)
            {
                cost[i][j]=sc.nextInt();
                if(i!=j && cost[i][j]==0){
                    cost[i][j]=inf;
                }
            }
        }

        int cm[][]=new int[v][v] ;
        for( i = 0; i<v; i++)
        {
            for(j = 0; j<v; j++)
            {
                cm[i][j] = cost[i][j]; //copy costMatrix to new matrix
            }
        }
        for( k = 0; k<v; k++)
        {
            for( i = 0; i<v; i++)
            {
                for( j = 0; j<v; j++)
                {
                    if(cm[i][k]+cm[k][j] < cm[i][j])
                    {
                        cm[i][j] = Math.min(cm[i][j],cm[i][k]+cm[k][j]);
                    }
                }
            }
        }
        System.out.println("all pairs shortest path:");
        for(i=0;i<v;i++)
        {
            for(j=0;j<v;j++)
            {
                if(cm[i][j]>=9999)
                {
                    System.out.println("INF");
                }
                else
                    System.out.print(cm[i][j]+" ");
            }
        }
    }
}
```

```
        System.out.println();
    }
}
```

### Output:



The screenshot shows a Java IDE with a console window. The console displays the following text:

```
<terminated> apsp [Java Application] C:\Program Files\Java\jre
Enter no.of vertices:
4
Enter cost adjacency matrix:
0 15 2 10
20 0 5 0
9 0 0 3
0 4 0 0
all pairs shortest path:
0 9 2 5
14 0 5 8
9 7 0 3
18 4 9 0
```



**Aim:** To write a java program to implement Backtracking algorithm for the Hamiltonian Cycles problem.

**Program:**

```
import java.util.*;
public class hamiltonian {
    static int n,c=0;
    static int[] x;
    static int[][] g;
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        System.out.print("Enter the no. of vertices:");
        n=s.nextInt();
        int i,j,e,y,z;
        x=new int[n+1];
        x[1]=1;
        for(i=2;i<=n;i++)
            x[i]=0;
        g=new int[n+1][n+1];
        System.out.println("Enter the Adjacency matrix of the Graph:");
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
                g[i][j]=s.nextInt();
        }
        System.out.println("Hamiltonian Cycles are:") ;
        Hamiltonian(2);
        if(c==0)
            System.out.println(c+"\nNo Hamiltonian Cycle for the given Graph!!");
    }

    static void NextValue(int k)
    {
        int j;
        do
        {
            x[k]=(x[k]+1)%(n+1);
            if(x[k]==0)
                return;
            if(g[x[k-1]][x[k]]!=0)
            {
                for(j=1;j<k;j++)
                {
                    if(x[j]==x[k])
                        break;
                }
                if(j==k)
                {
                    if((k<n) || ((k==n) && g[x[n]][x[1]]!=0))
                        return;
                }
            }
        }while(true);
    }

    static void Hamiltonian(int k)
```

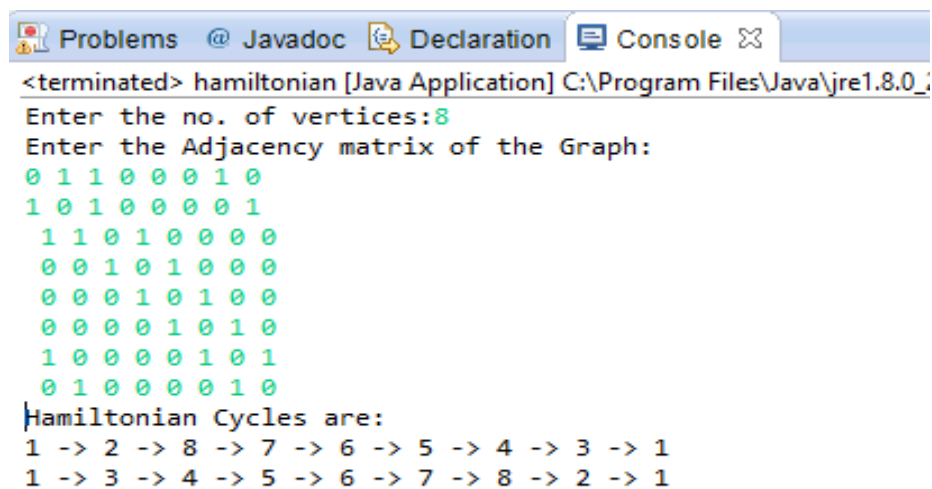
```

{
    int i;
    do{

        NextValue(k);
        if(x[k]==0)
            return;
        if(k==n)
        {
            for(i=1;i<=n;i++)
                System.out.print(x[i]+" -> ");
            System.out.print(x[1]);
            System.out.println();
            c++;
        }
        else
            Hamiltonian(k+1);
    }while(true);
}
}

```

Output:



```

Problems @ Javadoc Declaration Console
<terminated> hamiltonian [Java Application] C:\Program Files\Java\jre1.8.0_
Enter the no. of vertices:8
Enter the Adjacency matrix of the Graph:
0 1 1 0 0 0 0 1 0
1 0 1 0 0 0 0 0 1
1 1 0 1 0 0 0 0 0
0 0 1 0 1 0 0 0 0
0 0 0 1 0 1 0 0 0
0 0 0 0 1 0 1 0 1
1 0 0 0 0 1 0 1 0
0 1 0 0 0 0 0 1 0
Hamiltonian Cycles are:
1 -> 2 -> 8 -> 7 -> 6 -> 5 -> 4 -> 3 -> 1
1 -> 3 -> 4 -> 5 -> 6 -> 7 -> 8 -> 2 -> 1

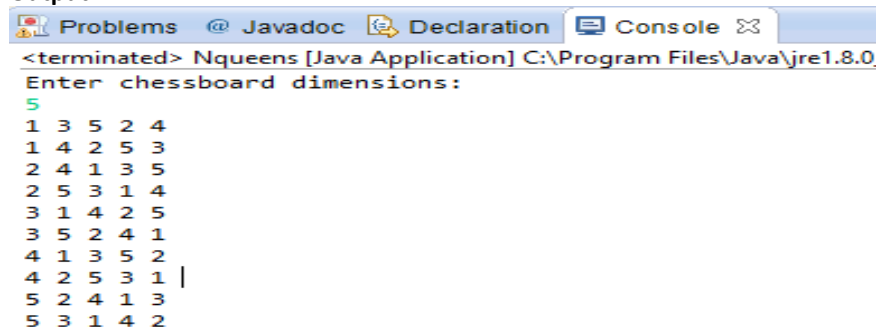
```

**Aim:** Write a Java program to implement Backtracking algorithm for the N-queens problem.

**Program:**

```
import java.util.*;
public class Nqueens{
    public boolean place(int k,int i,int x[]){
        int j;
        for(j=1;j<k;j++){
            if((x[j]==i || (Math.abs(x[j]-i)==Math.abs(j-k))))
            {
                return false;
            }
        }
        return true;
    }
    public void nqueens(int k,int n,int x[]){
        int i,j;
        for(i=1;i<=n;i++){
            if(place(k,i,x)==true){
                x[k]=i;
                if(k==n){
                    for(j=1;j<=n;j++){
                        System.out.print(x[j]+" ");
                    }
                    System.out.println();
                }
                else
                    nqueens(k+1,n,x);
            }
        }
    }
    public static void main(String args[])
    {
        int n;
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter chessboard dimensions:");
        n=sc.nextInt();
        int x[]=new int[n+1];
        Nqueens obj=new Nqueens();
        obj.nqueens(1,n,x);
    }
}
```

**Output:**



```
<terminated> Nqueens [Java Application] C:\Program Files\Java\jre1.8.0
Enter chessboard dimensions:
5
1 3 5 2 4
1 4 2 5 3
2 4 1 3 5
2 5 3 1 4
3 1 4 2 5
3 5 2 4 1
4 1 3 5 2
4 2 5 3 1
5 2 4 1 3
5 3 1 4 2
```

**Aim:** Write a Java program to implement Backtracking algorithm for the Sum of subsets problem.

**Program:**

```
import java.util.*;
class sumofsubsets {
    static int[] w;
    static int[] x;
    static int m,n,c=0;
    public static void main(String args[])
    {
        Scanner s=new Scanner(System.in);
        System.out.print("Enter the no. of weights:");
        n=s.nextInt();
        w = new int[n + 1];
        x= new int[n + 1];
        int t= 0,i;
        System.out.print("Enter " + n + " weights:");
        for(i=1;i<n+1;i++)
        {
            w[i]=s.nextInt();
            t+=w[i];
        }
        System.out.print("Enter the sum to be obtained:");
        m=s.nextInt();
        if (t<m || m<w[1])
        {
            System.out.println("Not possible to obtain the
subset!!");
            System.exit(1);
        }
        System.out.println("The possible solutions are:");
        SumOfSub(0, 1, t);
        //if(c==0)
        //System.out.println(c);
    }

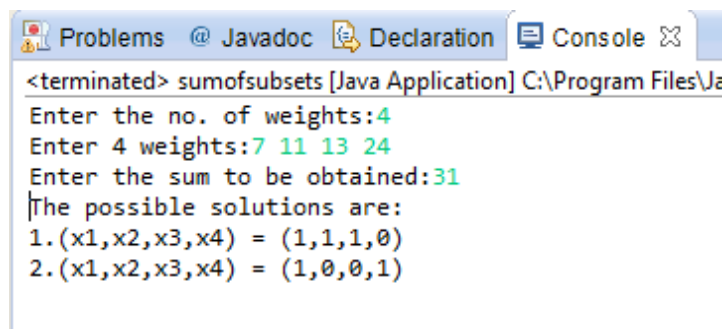
    static public void SumOfSub(int s, int k, int r)
    {
        int i=0,l;
        x[k]=1;
        if(s+w[k]==m)
        {
            c++;
            System.out.print(""+c+".");
            for(l=1;l<x.length;l++)
            {
                if(l!=x.length-1)
                    System.out.print("x"+l+",");
                else
                    System.out.print("x"+l);
            }
            System.out.print(") = (");
            for(i=1;i<=k;i++)
```

```

        {
            if(i!=k)
                System.out.print(x[i]+",");
            else
                System.out.print(x[i]);
        }
        for(i=k+1;i<x.length;i++)
            System.out.print(",0");
        System.out.print("\n");
    }
    else if((s+w[k]+w[k+1])<= m)
    {
        SumOfSub(s+w[k],k+1,r-w[k]);
    }
    if((s+r-w[k])>=m && (s+w[k+1])<=m)
    {
        x[k]=0;
        SumOfSub(s,k+1,r-w[k]);
    }
}
}

```

**Output:**



```

<terminated> sumofsubsets [Java Application] C:\Program Files\Ja
Enter the no. of weights:4
Enter 4 weights:7 11 13 24
Enter the sum to be obtained:31
The possible solutions are:
1.(x1,x2,x3,x4) = (1,1,1,0)
2.(x1,x2,x3,x4) = (1,0,0,1)

```