1：线性DP：

例题：

给定一个如下图所示的数字三角形，从顶部出发，在每一结点可以选择移动至其左下方的结点或移动至其右下方的结点，一直走到底层，要求找出一条路径，使路径上的数字的和最大。

```
        7
      3   8
    8   1   0
  2   7   4   4
4   5   2   6   5
```

输入格式

第一行包含整数 n，表示数字三角形的层数。

接下来 n 行，每行包含若干整数，其中第 i 行表示数字三角形第 i 层包含的整数。

输出格式

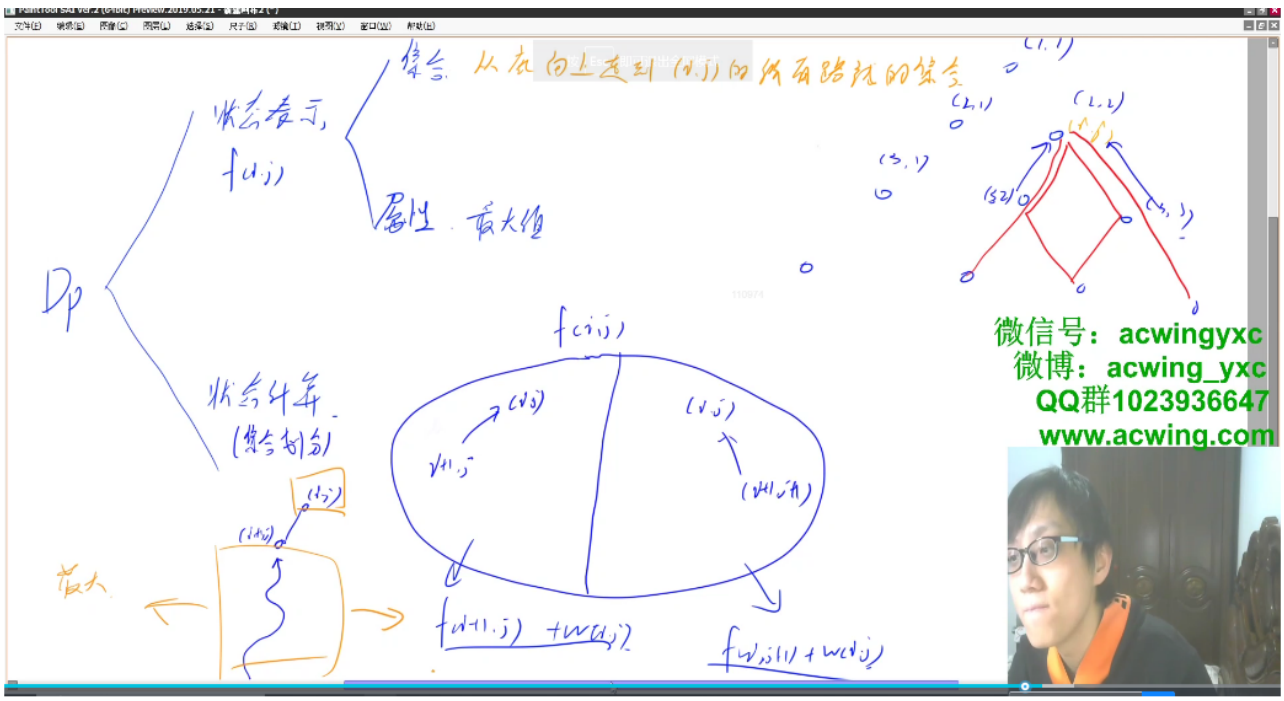
输出一个整数，表示最大的路径数字和。

输入样例：

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

输出样例：

```
30
```




```java
import java.util.*;
public class Main{
    public static int N = 510;
```

```
    public static int[][] f = new int[N][N];
    public static int[][] w = new int[N][N];
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for(int i = 1; i <= n; i++){
            for(int j = 1; j <= i; j++){
                w[i][j] = sc.nextInt();
            }
        }
        for(int i = 1; i <= n; i++){
            f[n][i] = w[n][i];
        }
        for(int i = n - 1; i >= 0; i--){
            for(int j = 1; j <= i; j++){
                f[i][j] = Math.max(f[i + 1][j] + w[i][j], f[i + 1][j + 1] + w[i][j]);
            }
        }
        System.out.println(f[1][1]);
    }
}
```

例题：

```
import java.util.*;
public class Main{
    public static int N = 1010;
    public static int[] f = new int[N];
    public static int[] a = new int[N];
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for(int i = 1; i <= n; i++){
            a[i] = sc.nextInt();
        }
        for(int i = 1; i <= n; i++){
            f[i] = 1;
            for(int j = 1; j < i; j++){
```

```
                if(a[i] > a[j]){
                    f[i] = Math.max(f[i], f[j] + 1);
                }
            }
        }
        int res = 0;
        for(int i = 1; i <= n; i++){
            res = Math.max(res, f[i]);
        }
        System.out.println(res);
    }
}
```

例题：

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int N = 1010;
        char[] a = new char[N];
        char[] b = new char[N];
        int[][] f = new int[N][N];
        int n = sc.nextInt();
        int m = sc.nextInt();
        String A = sc.next();
        String B = sc.next();
        for(int i = 1; i <= n; i++){
            a[i] = A.charAt(i - 1);
        }
        for(int i = 1; i <= m; i++){
            b[i] = B.charAt(i - 1);
        }
        for(int i = 1; i <= n; i++){
            for(int j = 1; j <= m; j++){
                f[i][j] = Math.max(f[i - 1][j], f[i][j - 1]);
                if(a[i] == b[j]){
                    f[i][j] = Math.max(f[i][j], f[i - 1][j - 1] + 1);
```

```
                }
            }
        }
        System.out.println(f[n][m]);
    }
}
```

2：区间DP：

例题：

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int N = 310;
        int[][] f = new int[N][N];
        int[] a = new int[N];
        int[] s = new int[N];
        for(int i = 1; i <= n; i++){
            a[i] = sc.nextInt();
        }
        for(int i = 1; i <= n; i++){
            s[i] = s[i - 1] + a[i];
        }
        for(int len = 2; len <= n; len ++){
            for(int i = 1; i + len - 1 <= n; i++){
                int j = i + len - 1;
                f[i][j] = 0x3f3f3f3f;
                for(int k = i; k < j; k++){
                    f[i][j] = Math.min(f[i][j], f[i][k] + f[k + 1][j] + s[j] - s[ i - 1]);
                }
            }
        }
        System.out.println(f[1][n]);
    }
}
```

3：计数DP：

例题：

```java
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int N = 1010;
        int mod = (int)(1e9 + 7);
        int[] f = new int[N];
        int n = sc.nextInt();
        f[0] = 1;
        for(int i = 1; i <= n; i++){
            for(int j = i; j <= n; j++){
                f[j] = (f[j] + f[j - i]) % mod;
            }
        }
        System.out.println(f[n]);
    }
}
```

4：状态压缩DP：

例题：

```java
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int N = 12,M = 1 << N;
        long[][] f = new long[N][M];
        int[][] state = new int[M][M];
        boolean[] st = new boolean[M];
        while(true){
            int n = scan.nextInt();
            int m = scan.nextInt();
```

```java
            if(n == 0 && m == 0){
                break;
            }
            for(int i = 0 ; i < 1 << n ; i ++ ){
                int cnt = 0;
                boolean flag = true;
                for(int j = 0 ; j < n ; j ++ ){
                    if((i >> j & 1) == 1){
                        if((cnt & 1) == 1){
                            flag = false;
                            break;
                        }
                        cnt = 0;
                    }else cnt ++ ;
                }
                    if((cnt & 1) == 1) flag = false;
                    st[i] = flag;
            }
            for(int i = 0 ; i < 1 << n  ; i ++ ){
                Arrays.fill(state[i],0);
                for(int j = 0 ; j < 1 << n ; j ++ ){
                    if((i & j) == 0 && st[i | j] ){
                        state[i][j] = 1;
                    }
                }
            }
            for(int i = 0 ; i < N ; i ++ )
                Arrays.fill(f[i],0);
                f[0][0] = 1;
            for(int i = 1 ; i <= m ; i ++ ){
                for(int j = 0 ; j < 1 << n ; j ++ ){
                    for(int k = 0 ; k < 1 << n ; k ++ ){
                        if(state[j][k] == 1) f[i][j] += f[i - 1][k];
                    }
                }
            }
            System.out.println(f[m][0]);
        }
    }
}
```