1. Floyd's algorithm uses dynamic programming to compute the *All-Pairs Shortest-Paths* of a directed graph. It grows the set of intermediate nodes on the path and generates a sequence of distance matrices \( D^{(0)}, D^{(1)}, \ldots, D^{(n)} \). The \( D^{(k)} \) matrix contains the shortest paths using only intermediate nodes from the set of vertices \( \{1, 2, 3, \ldots, k\} \).

```c
void floyd (int n, const number W[][], number D[][])

// Implements Floyd's algorithm for computing all-pairs shortest-paths of a directed graph
// Input: The adjacency matrix W of a digraph with n vertices
// Output: The matrix D containing the shortest distances between pairs of vertices

D[0] <- W

for k <- 1 to n do
    for i <- 1 to n do
        for j <- 1 to n do
            \( D^{(k)}[i][j] \leftarrow \min\{D^{(k-1)}[i][j], D^{(k-1)}[i][k] + D^{(k-1)}[k][j]\} \)

D <- D\((n)\)
```

(Note: We can actually get by with a single \( D \) matrix that is updated repeatedly.)

For the above graph, generates the sequence of distance matrices \( D^{(0)}, D^{(1)}, \ldots, D^{(n)} \).
2. If the following BST is searched according to the above probabilities, what is the average number of comparisons for successful searches?

```
    B
   / \
  A   D
 /   |
C
```

3. For the above probabilities, what would be the optimal BST?

4. Consider developing an algorithm for finding an optimal BST for keys: Key_1 < Key_2 < ... < Key_k < ... < Key_n given known search probabilities of p_1, p_2, ..., p_n, respectively. In a dynamic programming fashion we want to view the problem recursively, but solve the smallest problems to largest problem. To view the problem recursively, it is sometimes helpful to view the final solution. Suppose the following is an optimal BST, answer the following questions:

   a) What keys are in T_A?
   b) What properties must T_A possess?
   c) If x is the average number of comparisons to search T_A, how does this average change when T_A is made a subtree of root Key_k?

```
  Key_k
 /    |
T_A   T_B
```

5. What dimensionality of array(s) do we need to store the answers to smaller problems?