Warshall's Algorithm and the Reachability Matrix

Discussion

The adjacency matrix of any unweighted directed graph defines the adjacency relation on its set of vertices. When row *i* column *j* contains a 1 or true, it means that vertex *u* corresponding to row *i* is adjacent to vertex *v* corresponding to row *j* or that there is an edge *uv* in the graph.

In general that relation is neither reflexive nor transitive. Recall that those properties are defined as follows assuming that V is the set of vertices and R is the relation:

- Reflexive: $\forall v \in V, v \mathrel{R} v$
- Transitive: $\forall u, v, w \in V, u \mathrel{R} v, v \mathrel{R} w \Rightarrow u \mathrel{R} w$

Warshall's algorithm forms the reflexive-transitive closure of a matrix that defines the adjacency relation for a directed unweighted graph. The resulting matrix is sometimes referred to as a reachability matrix. When row *i* column *j* of the reachability matrix contains a 1 or true, it means that there is a directed path from vertex *u* corresponding to row *i* to vertex *v* corresponding to row *j*.

The pseudocode for Warshall's algorithm is shown below where `r` is the adjacency matrix for an unweighted directed graph and `n` is the number of vertices:

```
warshalls (r, n)
  for i = 1 to n
    r[i, i] = 1
  for k = 1 to n
    for i = 1 to n
      for j = 1 to n
        r[i, j] = r[i, j] ∨ r[i, k] ∧ r[k, j]
```

In our pseudocode, we assume that the array subscripts begin at 1. The first `for` loop computes the reflexive closure by filling the main diagonal of the matrix with all 1s. The triply nested loops that follow compute the transitive closure. The assignment in the body of that loop is referred to as processing the triple. In the pseudocode we have used the mathematical $\vee$ (or) and $\wedge$ (and) symbols. If we represent these values as Boolean values, logical operators can be used. If we represent them as numeric values, the bitwise or and and operators can used.

The order in which these three `for` loops are written critical. To ensure that the resulting matrix is the transitive closure, it is essential that the outermost loop be the intermediate vertex.

Sample Problem

Using Warshall's algorithm, compute the reflexive-transitive closure of the relation below. Show the matrix after the reflexive closure and then after each pass of the outermost `for` loop that computes the transitive closure.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution

The following matrix results after the reflexive closure. The values that changed as a result are show in red:

$$A0' = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The following five matrices show the matrix after each iteration of the outer loop.

$$A1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$A2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$A3 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$A4 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$A5 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Floyd’s Algorithm on Unweighted Graphs

Discussion

With a slight modification to Warshall’s algorithm, we can produce a final matrix that contains not only whether a path exists between two vertices but the length of that path. Note that if the graph contains cycles, the path lengths will represent the length of the shortest path. For this algorithm we must assume that the matrix contains a 1 when an edge is present and a 0 when it is not.

There are several changes that must be made. The main diagonal must be set to 0 not 1 because the shortest path from any vertex to itself is the path of length 0. All other 0s must be changed to ∞ to indicate that no edge exists between those two vertices. The way the triple is processed now changes also. The minimum operation replaces or and addition replaces and.

The modified algorithm is shown below where `r` is the adjacency matrix for an unweighted directed graph and `n` is the number of vertices:

```
floyds(r, n)
  for i = 1 to n
    for j = 1 to n
      if i = j
        r[i, j] = 0
      else if r[i, j] = 0
        r[i, j] = ∞
  for k = 1 to n
    for i = 1 to n
      for j = 1 to n
        r[i, j] = min(r[i, j], r[i, k] + r[k, j])
```

This modified algorithm is generally referred to as Floyd’s algorithm, although because the algorithms are so similar they are also often both called the Floyd-Warshall algorithm.

Sample Problem

Using the matrix in the previous problem show the final result of executing Floyd’s algorithm on that matrix to produce a matrix containing path lengths.

Solution

The final matrix after executing Floyd’s algorithm is shown below:

$$\begin{bmatrix} 0 & 1 & \infty & \infty & 2 \\ 1 & 0 & \infty & \infty & 1 \\ 1 & 2 & 0 & 1 & 3 \\ \infty & \infty & \infty & 0 & \infty \\ 1 & 2 & \infty & \infty & 0 \end{bmatrix}$$

Condensation Graphs and Partial Orders

Discussion

Recall that for a relation to be a partial order, it must be reflexive, transitive and antisymmetric. The antisymmetric property is defined as follows assuming that V is the set of vertices and R is the relation:

- Antisymmetric: $\forall u, v \in V, u \mathrel{R} v, v \mathrel{R} u \Rightarrow u = v$

Once we form the reflexive-transitive closure of the adjacency relation of a directed graph, we are assured that the first two properties are satisfied, but not necessarily the third. The antisymmetric property will only be true if the directed graph is acyclic meaning that is a DAG. For this reason every partial order on a finite set can be represented by a DAG and every DAG corresponds to a partial order.

When a directed graph does have cycles, it is still possible to define a partial order that corresponds to that graph by first reducing the graph to its *condensation graph*, sometimes also called its *component graph*. The condensation graph condenses every strongly-connected component to a single vertex. An edge exists between any distinct pair of condensed vertices if there is an edge between any of their component vertices.

The resulting condensation graph will always be acyclic because all cycles have been reduced to a single vertex. If we form the reflexive-transitive closure of the adjacency matrix of the condensation graph, the resulting relation will always be a partial order.

The algorithm is shown below where `G` is an unweighted directed graph:

```
partial_order(G)
  find the SCCs components of G
  create a set of vertices that correspond to the SCCs
  create the set of edges on those vertices induced by the original graph
  form the reflexive-transitive closure of the adjacency matrix of that graph
```

The algorithm for finding SCCs discussed two weeks ago can be used for the first step and Warshall’s algorithm can be used for the last step.

Sample Problem

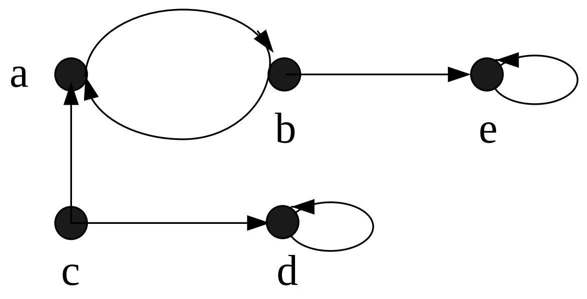
Given the following adjacency matrix:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

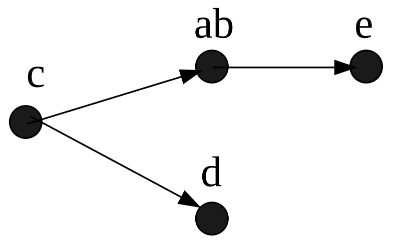
Show the graph that corresponds to that matrix assuming the rows and columns correspond to the vertices a, b, c, d and e. Show its condensation graph, renaming its vertices. Determine any topological order of that graph and create an adjacency matrix with the vertices ordered in that topological order. Finally compute the reflexive-transitive closure of that matrix. What characteristic of that matrix indicates that it defines a partial order?

Solution

Shown below is the original graph:



There are four SCCs. The vertices a and b form one and the vertices c, d and e are the other three. The condensation graph is shown below:



One topological order is c, d, ab, e. The adjacency matrix with the vertices in that order is shown below:

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Performing the reflexive-transitive closure gives the following matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The fact that there are all 0s below the main diagonal is what shows it is a partial order.

Floyd's Algorithm for Weighted Graphs

Discussion

When we have a matrix for a directed weighted graph, we assume that absence of an edge is already denoted by ∞ and main diagonal already contains 0s. Although we allow edges with negative weights, we make the absence of a negative weight cycles a precondition. The reason being that is that there is no shortest path between the vertices in a negative weight cycle. With these assumptions Floyd's algorithm is simplified.

The simplified algorithm is shown below where `r` is the adjacency matrix for an unweighted directed graph and `n` is the number of vertices:

```
floyds(r, n)
  for k = 1 to n
    for i = 1 to n
      for j = 1 to n
        r[i, j] = min(r[i, j], r[i, k] + r[k, j])
```

We can relax the precondition if we add logic to detect negative weight loops.

Sample Problem

Using Floyd's algorithm, compute the distance matrix for the weight directed graph defined by the following matrix:

$$\begin{bmatrix} 0 & 1 & 5 & 2 \\ 2 & 0 & \infty & 3 \\ 6 & \infty & 0 & -2 \\ \infty & -2 & 5 & 0 \end{bmatrix}$$

Show the intermediate matrices after each iteration of the outermost loop.

Solution

The following four matrices show the matrix after each iteration of the outer loop. The values that changed as a result are show in red:

$$D1 = \begin{bmatrix} 0 & 1 & 5 & 2 \\ 2 & 0 & 7 & 3 \\ 6 & 7 & 0 & -2 \\ \infty & -2 & 5 & 0 \end{bmatrix}$$

$$D2 = \begin{bmatrix} 0 & 1 & 5 & 2 \\ 2 & 0 & 7 & 3 \\ 6 & 7 & 0 & -2 \\ 0 & -2 & 5 & 0 \end{bmatrix}$$

$$D3 = \begin{bmatrix} 0 & 1 & 5 & 2 \\ 2 & 0 & 7 & 3 \\ 6 & 7 & 0 & -2 \\ 0 & -2 & 5 & 0 \end{bmatrix}$$

$$D4 = \begin{bmatrix} 0 & 0 & 5 & 2 \\ 2 & 0 & 7 & 3 \\ -2 & -4 & 0 & -2 \\ 0 & -2 & 5 & 0 \end{bmatrix}$$