

Program 3

CS 231
Fall 1989
Craig A. Rich

Complete implementations of the operations in the following abstract data type which reads and operates on directed graphs of type DIGRAPH.

digraph_adt.pas

```
[environment]

module DIGRAPH_ADT;

const MAX_VERTICES = 20;

type VERTEX = 1..MAX_VERTICES;
    ADJACENCY_MATRIX = array [VERTEX,VERTEX] of BOOLEAN;
    DIGRAPH = record
        N: 0..MAX_VERTICES;    (* The number of vertices *)
        E: ADJACENCY_MATRIX
    end;

procedure READ_DIGRAPH (var INFILE: TEXT; var G: DIGRAPH); ...
function SOURCE (V: VERTEX; G: DIGRAPH): BOOLEAN; ...
function SINK (V: VERTEX; G: DIGRAPH): BOOLEAN; ...
procedure TRANSITIVE_CLOSURE (G: DIGRAPH; var CLOSURE: DIGRAPH); ...

end.
```

Write a Pascal program which reads an adjacency lists representation of a digraph from a file, then enters an interactive query loop which processes one of the following single-letter queries:

- S: Is vertex i a source?
- K: Is vertex i a sink?
- I: Is vertex i isolated?
- C: Is the digraph cyclic?
- P: Is there a path from vertex i to vertex j ?
- X: Exit the query loop.

Digraph ADT Module Implementation Notes

1. `READ_DIGRAPH` reads the digraph `G` from the Pascal file `INFILE`. The input file will consist of `N` lines—one for each vertex in the digraph. Line i of the input file contains the numbers of the vertices which are adjacent to vertex i , for $1 \leq i \leq N$. While the file actually contains an adjacency lists representation of the digraph (each line is an adjacency list), the digraph is represented internally by an adjacency matrix `E`.
2. `SOURCE` and `SINK` return true if and only if the vertex `V` in the digraph `G` is a source or sink, respectively.
3. `TRANSITIVE_CLOSURE` should compute `CLOSURE` to be the transitive closure of the digraph `G`. Do not include the reflexive closure in `CLOSURE`.

Program 3 Implementation Notes

1. Determining whether a vertex i is a source, a sink, or isolated can be done using the functions `SOURCE` and `SINK`.
2. Determining whether the digraph is cyclic or there is a path from vertex i to vertex j can be done using the transitive closure of the digraph. Since computing the transitive closure is the least efficient operation (as a function of the number of vertices `N`), compute the transitive closure only once before the query loop is entered.
3. Construct a separate procedure for obtaining a vertex number from the interactive input file, since most of the queries require a vertex to be input.

What to Hand In

1. A single contiguous printout containing compiler listings of the digraph ADT module and main program, and listings of at least one sample input file and the output of your program when run on that file—in that order.