* of an arbitrary closed semiring. Of course, for structures such as the set of all sets of strings over an alphabet, it is not clear that such operations can be implemented at all, let alone in "unit time." However, for the semirings of which we shall make use, the operations will be easy to perform.

Algorithm 5.5. Computation of costs between vertices.

Input. A directed graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$, and a labeling function l: $(V \times V) \to S$, where $(S, +, \cdot, 0, 1)$ is a closed semiring. We take $l(v_i, v_j) = 0$ if (v_i, v_j) is not in E.

Output. For all i and j between 1 and n, the element $c(v_i, v_j)$ of S which is equal to the sum over all paths from v_i to v_j of the label of that path.

Method. We compute C_{ij}^k for all $1 \le i \le n$, $1 \le j \le n$, and $0 \le k \le n$. The intention is that C_{ij}^k should be the sum of the labels of all paths from v_i to v_j such that all vertices on the path, except possibly the endpoints, are in the set $\{v_1, v_2, \ldots, v_k\}$. For example, the path v_9, v_3, v_8 is considered in C_{98}^4 and C_{98}^3 but not in C_{98}^2. The algorithm is as follows.

begin
1. **for** $i \leftarrow 1$ **until** n **do** $C_{ii}^0 \leftarrow 1 + l(v_i, v_i)$;
2. **for** $1 \le i, j \le n$ and $i \ne j$ **do** $C_{ij}^0 \leftarrow l(v_i, v_j)$;
3. **for** $k \leftarrow 1$ **until** n **do**
4. **for** $1 \le i, j \le n$ **do**
5. $C_{ij}^k \leftarrow C_{ij}^{k-1} + C_{ik}^{k-1} \cdot (C_{kk}^{k-1})^* \cdot C_{kj}^{k-1}$;
6. **for** $1 \le i, j \le n$ **do** $c(v_i, v_j) \leftarrow C_{ij}^n$

end □

Theorem 5.5. Algorithm 5.5 uses $O(n^3)$ $+$, $\cdot$, and * operations from the semiring and computes $c(v_i, v_j)$ for $1 \le i, j \le n$.

Proof. It is easy to check that line 5 is executed n^3 times, requiring four operations each time, and that the **for** loops of lines 1, 2, and 6 are iterated at most n^2 times each. Thus $O(n^3)$ operations suffice.

To show correctness, we must prove by induction on k that C_{ij}^k is the sum, over all paths from v_i to v_j with no intermediate vertex (excluding the endpoints) of index higher than k, of the label of such a path. The basis, $k = 0$, is trivial by lines 1 and 2, since any such path is of zero length, or consists of a single edge. The induction step follows from line 5, since a path from v_i to v_j with no intermediate vertex higher than k either

i) has no intermediate vertex higher than $k - 1$ (the term C_{ij}^{k-1}), or
ii) goes from v_i to v_k, then from v_k to v_k some number of times (possibly 0), and finally from v_k to v_j, all with no intermediate vertex higher than $k - 1$ [the term $C_{ik}^{k-1} \cdot (C_{kk}^{k-1})^* \cdot C_{kj}^{k-1}$].

The laws of a closed semiring insure that line 5 correctly computes t of the labels of all these paths. □

A TRANSITIVE CLOSURE ALGORITHM

specialize Algorithm 5.5 to two interesting cases. The first is to the clos iring S_1 described in Example 5.9. In S_1 addition and multiplicati asy to perform, and * is also easy, since $0^* = 1^* = 1$. In fact, since 1 identity, we may replace line 5 of Algorithm 5.5 by†

$$C_{ij}^k \leftarrow C_{ij}^{k-1} + C_{ik}^{k-1} \cdot C_{kj}^{k-1}. \tag{5.}$$

To compute the reflexive-transitive closure of a graph, we define t ng function

$$l(v, w) = \begin{cases} 1, & \text{if } (v, w) \text{ is an edge,} \\ 0, & \text{if not.} \end{cases}$$

$c(v, w) = 1$ if and only if there is a path of length 0 or more from v to y easily be checked, using the laws of the closed semiring $\{0, 1\}$.

le 5.12. Consider the graph of Fig. 5.18, ignoring the numbers on t temporarily. The labeling function $l(v_i, v_j)$ is given as follows.

	v_1	v_2	v_3
v_1	1	1	1
v_2	1	0	0
v_3	0	1	0

$l(v_i, v_j)$

hen, line 1 of Algorithm 5.5 sets $C_{11}^0 = C_{22}^0 = C_{33}^0 = 1$. Line 2 se (v_i, v_j) for $i \ne j$, so we have the following values for the C_{ij}^0's.

	v_1	v_2	v_3
v_1	1	1	1
v_2	1	1	0
v_3	0	1	1

C_{ij}^0

natural to interpret this observation as saying "it is sufficient to consider on ree paths." However, it should be noted that with (5.4) in place of line hm 5.5 will compute sums over sets of paths which include all cycle-free path ne others as well.