$$\overline{\text{PATH}} \in \mathbf{NL}.$$

First, compute $k =$ number of nodes reachable from s.

To this end, define $A_i =$ set of nodes of distance at most i from s. So $A_0 = \{s\}$, $A_i \subseteq A_{i+1}$.

(A_i is *for our eyes only*, to trace the algorithm; it is too big to fit in log-space.)

A_m contains all the nodes reachable from s.

$k_i = |A_i|$, and $k = k_n = |A_n|$.

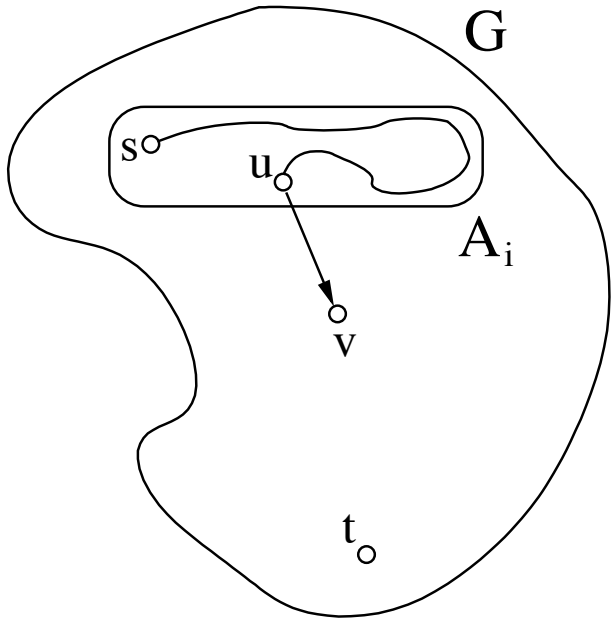
We are interested in computing $k_0, k_1, k_2, \ldots, k_n = k$ using the *inductive counting method*.

Compute k_{i+1} from k_i:

To determine if $v \in A_{i+1}$, go through each node u, and test if it is in A_i (to test, we guess a path from s to u of length at most i, and check that it is indeed a path).

As we have k_i, we know how many u's we should get (one way to be wrong: get too few u's, in which case the branch dies).

For each $u \in A_i$, we check if (u, v) is an edge.



Once we have $k = k_n$, we can now test if t is reachable from s.

For each node u_i in $s = u_1, u_2, \ldots, t = u_n$, guess a path $s \rightsquigarrow u_i$.

If during the path guessing we encounter t, we reject.

If we fail to guess a path $s \rightsquigarrow u_i$, move on to u_{i+1}.

If we guess a path $s \rightsquigarrow u_i$, decrease k by 1, move to u_{i+1}.

By the time we have dealt with u_{n-1}, if $k = 0$, we know that we have successfully accounted for all the nodes in A_n, and t at this stage is for sure not one of them, so we accept.

Otherwise, we go on to $u_n = t$. We guess a path to t, and if we succeed we reject. Otherwise, $k > 0$, and we know we missed some node in A_n (perhaps t), so we reject.