First transformation: Reachability

Given: graph $G$, nodes $s, t$
Goal: construct log space NTM that accepts if no $s \rightarrow t$ path

Rephrase as reachability:
find all nodes of $G$ that are reachable from $S$.

Can’t store set of reachable nodes in $\log n$ space.
Could we count?
Assume we could know/guess count of reachable nodes.
Does this help?

Finding the count of reachable nodes

Idea: do BFS, count the nodes at each level
Let $A_i =$ set of nodes at level (distance) $i$ from $s$ in BFS
We will count $c_i = |A_i|$.
How? Like before: (again!) guess $c_i$ and verify
Loop through all nodes in $G$, guess if in $A_{i+1}$.
How to check? Need to know predecessor in $A_i$.
Catch-22? Recurrence? No, can guess!

Check reachable nodes knowing count

Given any node $u$, we can check in $\log n$ space if reachable
this is PATH!

Idea: we can do this sequentially!
Iterate through all nodes, at each flip bit, guess if reachable.
if guessed unreachable, go to next
if guessed reachable and node is $t$, reject (don’t want that)
else verify: if not reachable, reject.
else increment count
Finally, check if count is expected value, accept/reject.

Summary: if nondeterministically:
we can select the right number of reachable nodes
and we did not select $t$
and each is indeed reachable
then we know the other nodes (incl. $t$) are not reachable, accept.

Finding node count at each level

We already know $c_i$ (from previous iteration, $c_0 = 1$).
Will re-count and re-find them for each candidate for $A_{i+1}$

Compute $\{c_i \}_{i=0}^{m}$ (nodes of $A_{i+1}$):
loop through all nodes $v$ (candidate for $A_{i+1}$)
start re-counting $A_i$ ($r_i = 0$)
loop through all $v$ (candidate for $A_i$)
  guess if $u$ in $A_i$
  check (guess path of length $\leq i$), reject if not
  else increment re-count $r_i$
  if edge $(u, v)$, increment $c_{i+1}$, take next $v$
  if recount $r_i \neq c_i$, reject
We compute $c_i$ until $c_{i+1} = c_i$ (at most $c_m$)
To implement, need these counters:

- $i$: counts distance/levels ($A_i$), up to $\leq m$
- $c_i$: count of level $A_i$
- $c_{i+1}$: count of next level
- temp counter to re-count/check $c_i$ and $c_m$
- $u, v$: previous / current nodes

$L \subseteq NL = coNL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$

We’ll see $NL \not\subseteq PSPACE$
so $P$ (separating them must be different from one of them).