Wed April 18, 2018 - page 1

Tuesday, April 17, 2018 8:41 PM

Two Surprising theorems about non-deterministic space classes.

Savitch '70 : NPSPACE = PSPACE.
(NL ⊂ SPACE($\log^2 n$))

Immerman-Szelepscény ('87 | '88) NL = co-NL.

These are the opposite of what we believe to hold for time complexity classes.
(We believe: P ≠ NP NP ≠ co-NP).

Savitch's Theorem ⇒ STCONN ∈ SPACE($\log^2 n$).

⇒ Corollary: NL ⊆ SPACE($\log^2 n$).

Corollary: NPSPACE = PSPACE.

- Configuration graph for PSPACE NTM has size $2^{c \cdot n}$ $|\Sigma|^{n^k}$
- Algorithm to solve STCONN in $\log^2 n$ space doesn't build the whole graph.
  Only asks queries: Is (i, j) an edge?
- So we can solve Connectivity on a graph of size $2^{c \cdot n}$ using space: $\log^2(2^{c \cdot n}) = O(n^k)$.

Wed April 18, 2018 - page 2

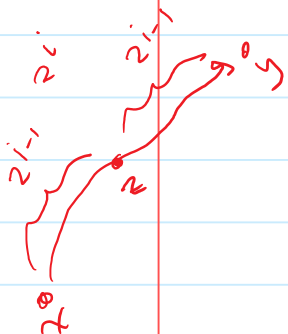
Tuesday, April 17, 2018 8:41 PM

Proof that ST-CONN $\in$ SPACE$(\log^2 n)$.

Input: $G = (V, E)$ s, t.
Is there a path in G from s to t?

$2^{\log n} = n$.

Recursive Alg: → Will return results of PATH$(s, t, \log n)$



PATH$(\underbrace{x, y}_{\in V}, i)$ // is there a path from x to y of length $\leq 2^i$?

- If $i = 0$ return$(x == y \ \vee \ (x,y) \in E)$
  For all node z
  if PATH$(x, z, i-1) \wedge$ PATH$(z, y, i-1)$ return true.
  Return (false).

Space used: $\underbrace{(\text{depth of recursion})}_{\log n} \times (\text{size of stack record})$

Stack Record: (x, y, i) → $O(\log n)$ space.

Can figure out what to do next from current and previous stack record:

$(x, y, i)\ (x, z, i{-}1)$ → if PATH$(x, z, i-1)$ returns YES call $(z, y, i-1)$

$(x, y, i)\ (z, y, i{-}1)$ → pop record + return result of PATH(z, y, i).

$\log n$

$(\_\_\_)(\_\_\_) \ldots (\_\_\_)$

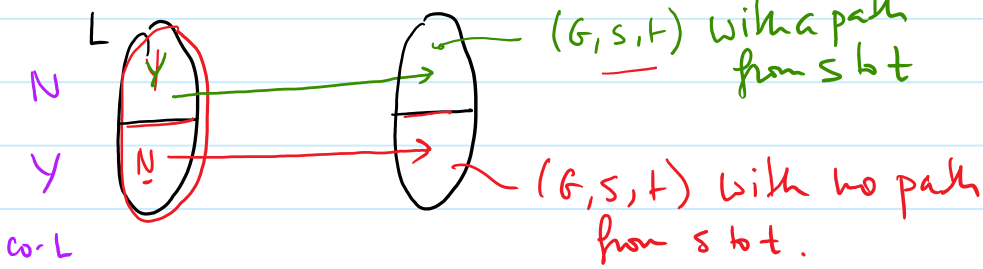
//

Tuesday, April 17, 2018 8:41 PM

ST-NON-CONN

Input: $G=(V,E)$ s,t.
Is there **no** path from s to t in G?

$L \in NL$

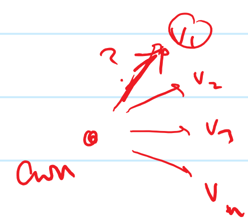


co-L $\in$ N

ST-NON-CONN is complete for co-NL

Immerman-Szelepcsény: ST-NON-CONN $\in$ NL.
$\Rightarrow$ co-NL = NL.

Review of non-deterministic log-space algorithm for ST-CONN:

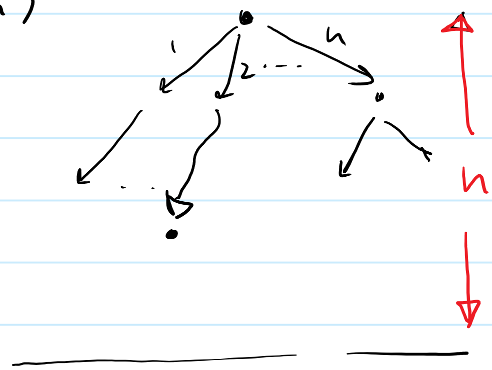



```
Counter = 0
current node = s
while (current node != t + counter < n)
    guess v
    if (current node, v) ∈ E
        current node ← v
        counter++
    else reject.
```

After loop:
```
If current node == t
    accept
else
    reject.
```

Tuesday, April 17, 2018 8:41 PM

Each computation path in the tree is a sequence of node labels : $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$



Computation stops if:

$(v_{i_j}, v_{i_{j+1}})$ not an edge REJ

Or if t is reached ACC.

Suppose we know the # of nodes that can be reached from s. (call this # R).

If we can show R distinct nodes

- all reachable from s.
- none of them $= t$

$\Rightarrow$ t not reachable from s.

First show an alg for ST-NON-CONN assuming we know the # (R) nodes reachable from s.

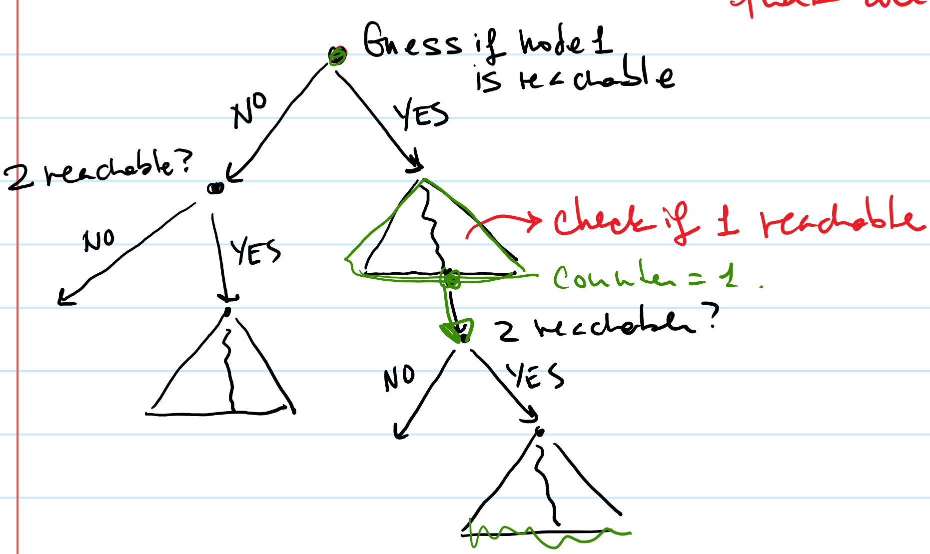
Wed April 18, 2018 - page 5

Tuesday, April 17, 2018 8:41 PM

counter = 0
For each v = 1, 2, ..., n
  Non-deterministically guess if v is reachable from s.
  If guess = YES
    Solve ST-CONN (s, v) using log space.
      Guess the path from s to v. If guess doesn't lead to v, REJECT.
      If path does lead to v.
        If (v == t) REJECT.
        Else counter = counter + 1.

If (counter == R) accept.
Else reject.

↳ the only way to accept if R distinct nodes are reached that are not = t.



Computation guesses a subset of the nodes:

v_1 v_2 . . . . . v_n
Y/N Y/N Y/N.

Computation continues only if each YES guess is correct.

- t is not in the subset.
- subset has size ≥ R.

Tuesday, April 17, 2018 8:41 PM

$R(n) = R$.

How to compute R?

$R(i)$ = # nodes reachable in $\le i$ steps.

$R(0) = 1$.

Compute $R(i+1)$ from $R(i)$ using non-determinism.

Uses only $\log n$ bits.

Eventually have $R(n) = R$

Only need to keep $R(i)$ to get $R(i+1)$

Initialize $R(i+1) = 0$

For each $v \in V$ guess if v is reachable from s in $\le i+1$ steps.

- If guess = YES

  Same as NL alg for ST-CONN Threshold for counter $= i+1$.

  Use NL procedure to verify path from s to v with $\le i+1$ edges. There will be an accepting path if guess is correct.

  If guess is correct, increment $R(i+1)$.

- If guess = NO.

  Still need to explain this part

  Use NL procedure to verify that there is no path from s to v that uses $\le i+1$ edges. There will be an accepting path if the guess is correct. Use $R(i)$.

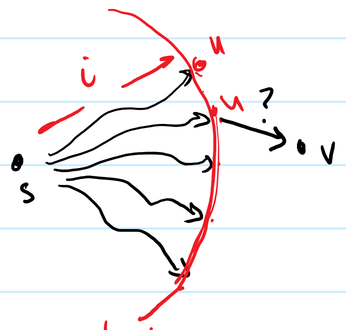
At the end of n guesses, only one sequence of guesses will survive. This one will have the correct value of $R(i+1)$.

Wed April 18, 2018 - page 7

Tuesday, April 17, 2018 8:41 PM

NL procedure to verify that there is no path from s to v with $\le i+1$ edges.

- Use $R(i)$ to reach every node that can be reached from s in $\le i$ steps.
  For each vertex w that is reached, verify that there is no edge (w, v).
  Reject if v is reached
  Accept otherwise.



know $R(i)$ so we can make sure we have reached all w reachable from s in $\le i$ steps.

Wed April 18, 2018 - page 8

Tuesday, April 17, 2018 8:41 PM

Expanded version of procedure to check if v is reachable from s in $\le i+1$ edges.
(Assuming we know $R(i)$).

Counter = 0
For $u = 1, 2, 3, \ldots, n$
  Non-deterministically guess if u is reachable from s in $\le i$ steps.
  If guess = YES
    CurrentNode = s.
    $\ell = 0$
    While ($\ell <$ ~~5~~ i and curr $\neq u$)
      Non-deterministically guess node w
      If (currentNode, w) $\in E$
        currentNode $\leftarrow w$.
        ℓ++.
      Else reject.
    If (currentNode $\neq u$)
      Reject.
    Counter++
    If there is an edge (u, v)
      Reject.
If Counter $\neq R(i)$
  Reject

Computation continues only if YES guess was correct.

check for path of length $i+1$ to v through u.

Only accept if all $R(i)$ nodes reachable from s in $\le i$ steps were reached.