Goals:

- Prove PATH is NL complete

PATH ∈ NL - Complete

(nondeterministic logspace)

Step 1: PATH ∈ NL

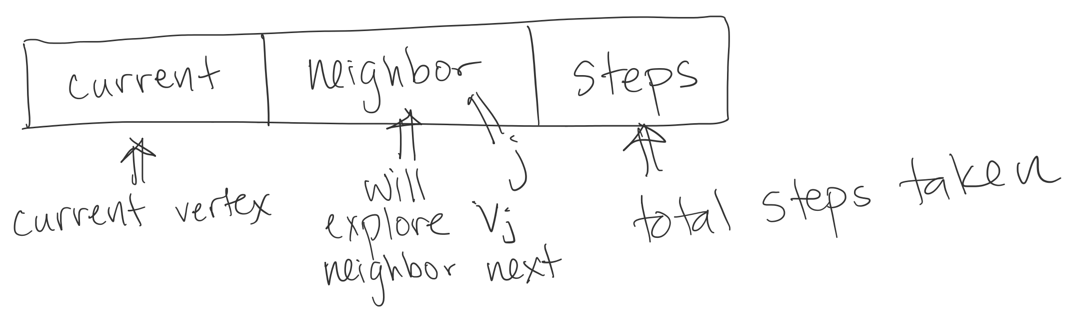
PATH = { ⟨G, s, t⟩ : There is a path from s to t in graph G }

n vertices; G is an adjacency matrix

	v_1	v_2	v_3	v_4	...	v_n
v_1	0	1	0	0		
v_2	1	0	1	1		
v_3	0	0	0	1		
⋮						
v_n						

Strategy: Do BFS

R/W Tape:

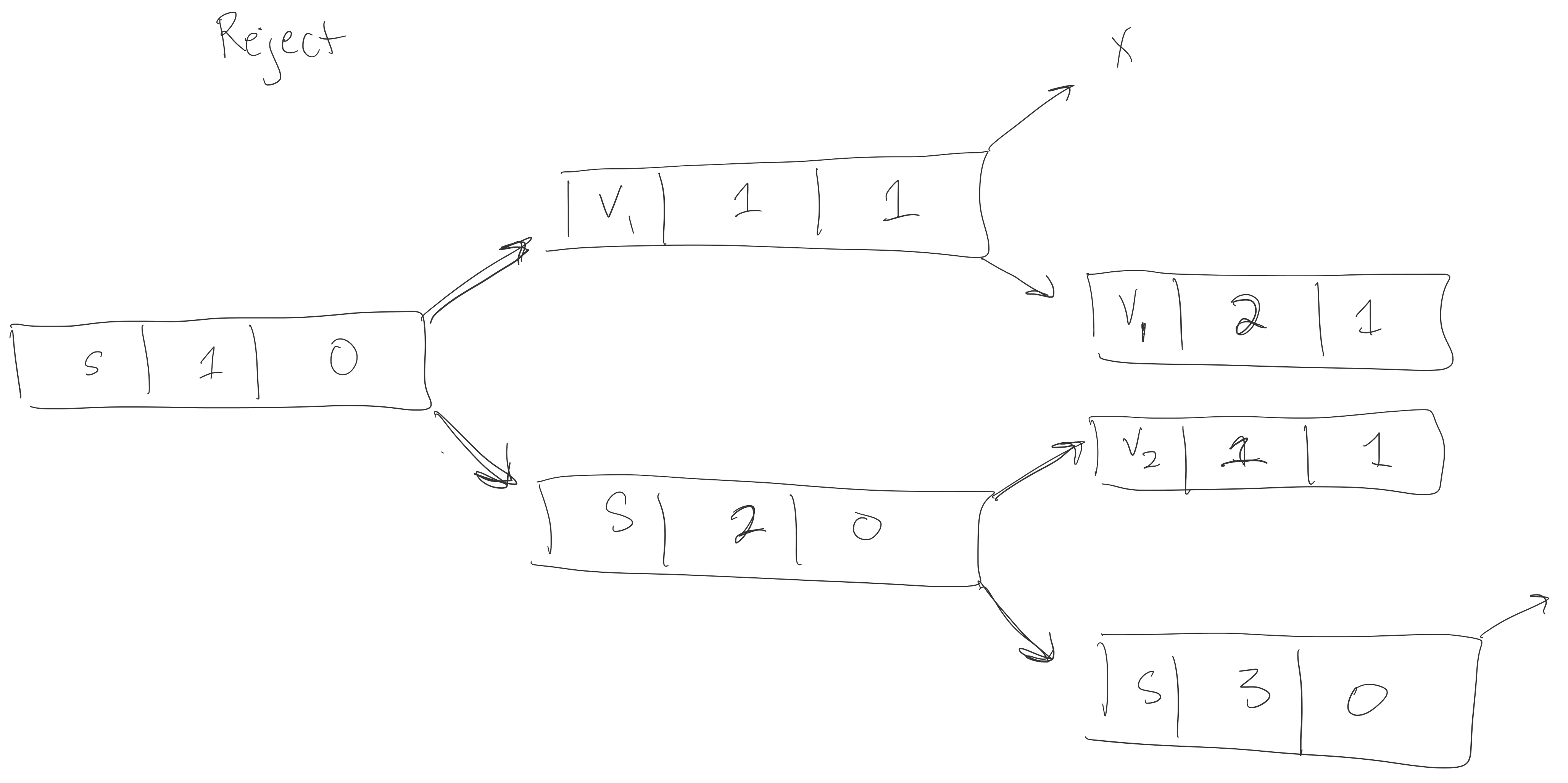


Initialize

While (steps ≤ n)

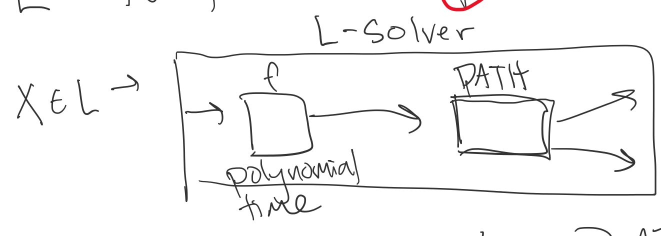
- If current == t : Accept
- Nondeterministic Choice 1: // Go to neighbor if possible if (edge b/t current + neighbor)
  - current = neighbor
  - neighbor = 1
  - steps = steps + 1
- Nondeterministic Choice 2: // Update neighbor
  - current = current
  - neighbor = neighbor + 1
  - steps = steps

Reject



PATH is NL-Hard

- ∀ L ∈ NL, L $\le_P$ PATH

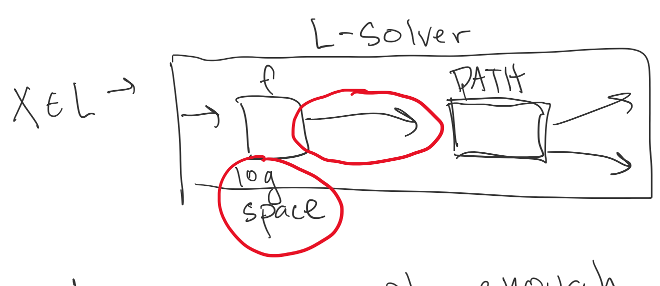


f solve L

x ∈ L → { (s,t), s, t }

x ∉ L → { ∅, s, t }

- ∀ L ∈ NL, L $\le_{ls}$ PATH (log space)

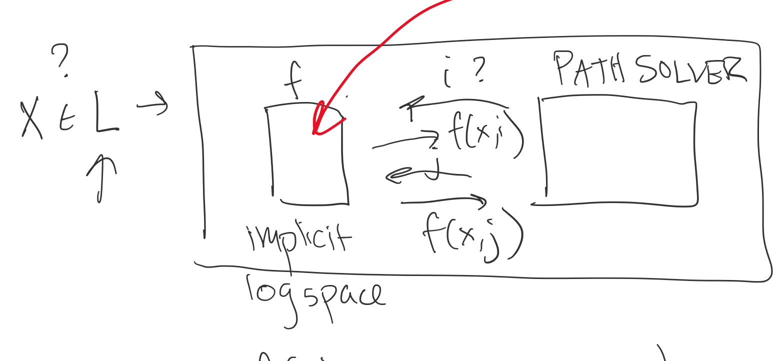


Not ok → logspace is not enough space for f to print out adjacency matrix

$\le_{ls}$ ≡ implicit log space

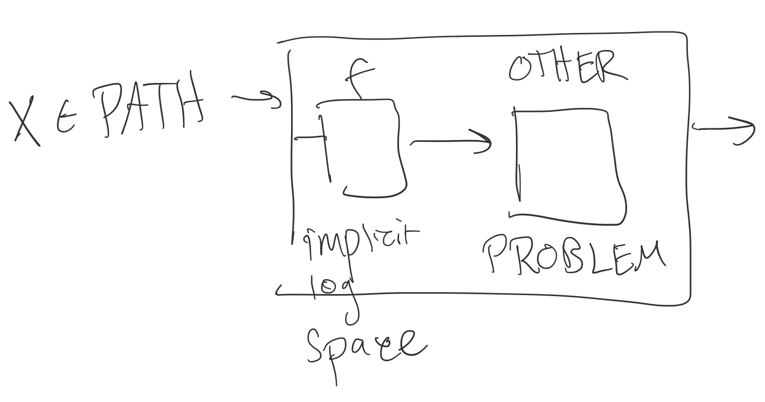
f(x, i) outputs the i-th bit of f(x)

x · description of M's transition function



f(x) is a graph

f(x,i) → is there an edge i



Big Idea of PATH ∈ NL-Hard

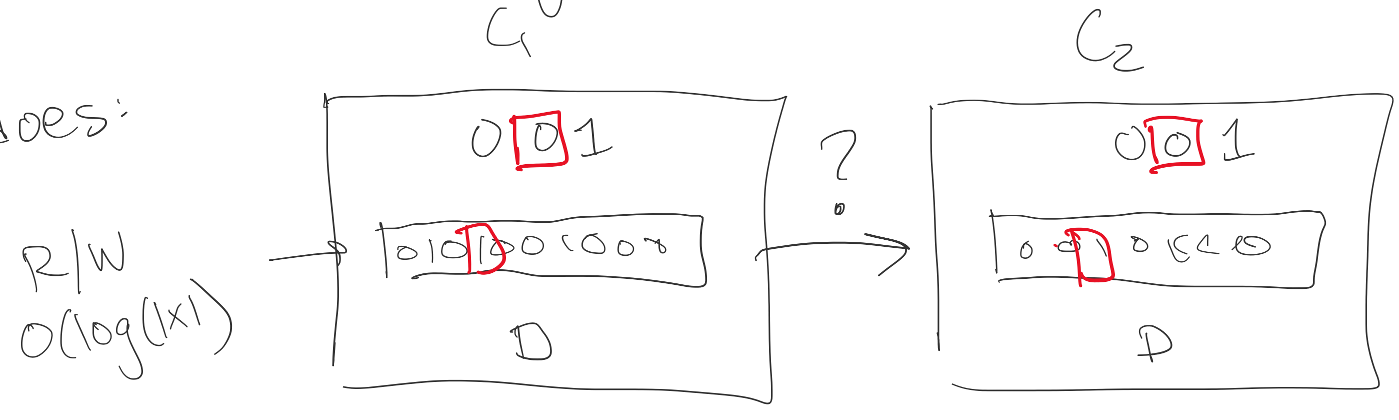
x ∈ NL → ∃ a NTM M that only uses O(log|x|) bits of its R/W tape

f "creates" input ⟨ G, $C_{start,x}$, $C_{accept,x}$ ⟩

(G: configuration graph of M)

PATH SOLVER queries: C_1 connected to C_2 in G?

f does:



Can check in O(log(|x|)) space