



Computational Complexity Theory

Lecture 9: Space complexity classes

Department of Computer Science,
Indian Institute of Science

Space bounded computation

- Here, we are interested to find out how much of work space is required to solve a problem.
- For convenience, think of TMs with a separate read-only input tape and one or more work tapes. Work space is the number of cells in the work tapes of a TM **M** visited by **M**'s heads during a computation.

Space bounded computation

- Here, we are interested to find out how much of work space is required to solve a problem.
- For convenience, think of TMs with a separate read-only input tape and one or more work tapes. Work space is the number of cells in the work tapes of a TM M visited by M 's heads during a computation.
- **Definition.** Let $S: \mathbb{N} \rightarrow \mathbb{N}$ be a function. A language L is in $\text{DSPACE}(S(n))$ if there's a TM M that decides L using $O(S(n))$ work space on inputs of length n .

Space bounded computation

- Here, we are interested to find out how much of work space is required to solve a problem.
- For convenience, think of TMs with a separate read-only input tape and one or more work tapes. Work space is the number of cells in the work tapes of a TM M visited by M 's heads during a computation.
- **Definition.** Let $S: \mathbb{N} \rightarrow \mathbb{N}$ be a function. A language L is in $\text{NSPACE}(S(n))$ if there's a NTM M that decides L using $O(S(n))$ work space on inputs of length n , regardless of M 's nondeterministic choices.

Space bounded computation

- We'll refer to 'work space' as 'space'. For convenience, assume there's a single work tape.
- If the output has many bits, then we will assume that the TM has a separate write-only output tape.

Space bounded computation

- We'll refer to 'work space' as 'space'. For convenience, assume there's a single work tape.
- If the output has many bits, then we will assume that the TM has a separate write-only output tape.
- **Definition.** Let $S: \mathbb{N} \rightarrow \mathbb{N}$ be a function. S is space constructible if $S(n) \geq \log n$ and there's a TM that computes $S(|x|)$ from x using $O(S(|x|))$ space.

Relation between time and space

- Obs. $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.



Hopcroft, Paul & Valiant 1977

Relation between time and space

- **Obs.** $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.
- **Theorem.** $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$, if S is space constructible.
- **Proof.** Uses the notion of configuration graph of a TM. We'll see this shortly.

Relation between time and space

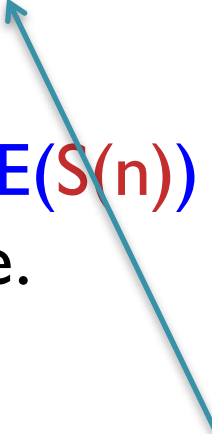
- **Obs.** $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.
- **Theorem.** $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$, if S is space constructible.
- **Definition.**
 $L = \text{DSPACE}(\log n)$
 $NL = \text{NSPACE}(\log n)$
 $\text{PSPACE} = \bigcup_{c > 0} \text{DSPACE}(n^c)$

Relation between time and space

- **Obs.** $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.
- **Theorem.** $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$, if S is space constructible.
- **Definition.**
 $L = \text{DSPACE}(\log n)$
 $NL = \text{NSPACE}(\log n)$
 $\text{PSPACE} = \bigcup_{c > 0} \text{DSPACE}(n^c)$

Giving space at least $\log n$ gives a TM at least the power to remember the index of a cell.

Relation between time and space

- **Obs.** $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.
 - **Theorem.** $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$, if S is space constructible.
 - **Caution.** The *Hopcroft-Paul-Valiant* theorem does not imply $P \subsetneq PSPACE$.
 - **Open.** Is $P \neq PSPACE$?
- 

Relation between time and space

- Obs. $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.
- Theorem. $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$, if S is space constructible.
- Theorem. $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$



Follows from the above theorem

Relation between time and space

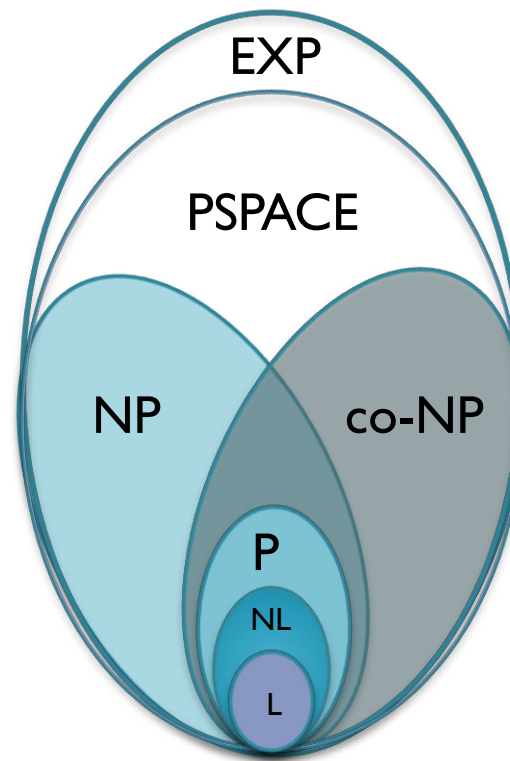
- Obs. $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.
- Theorem. $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$, if S is space constructible.
- Theorem. $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$



Run through all possible choices of certificates of the verifier and **reuse** space.

Relation between time and space

- **Obs.** $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.
- **Theorem.** $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$, if S is space constructible.

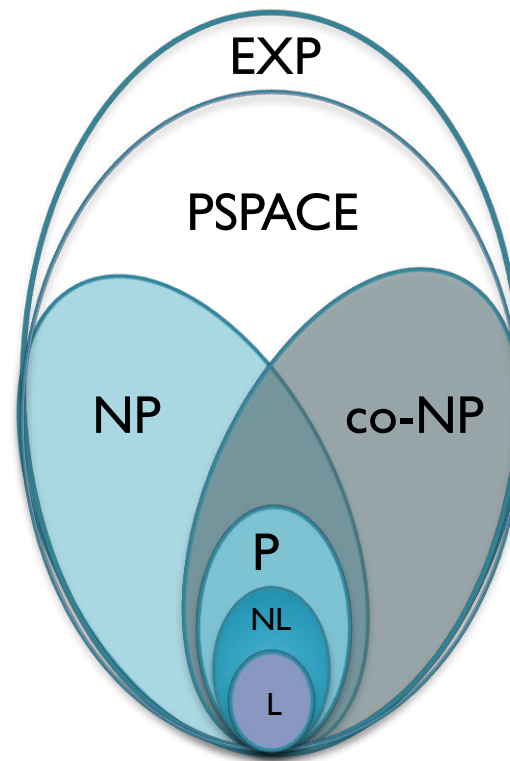


Relation between time and space

- **Obs.** $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.
- **Theorem.** $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$, if S is space constructible.

Homework: Integer addition and multiplication are in (functional) L .

Integer division is also in (functional) L . (Chiu, Davida & Litow 2001)



Configuration graph

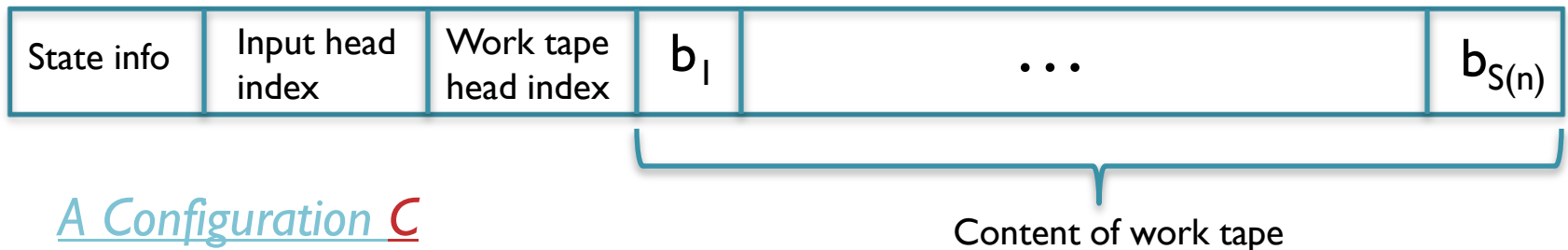
- **Definition.** A *configuration* of a TM **M** on input **x**, at any particular step of its execution, consists of
 - (a) the nonblank symbols of its work tapes,
 - (b) the current state,
 - (c) the current head positions.

It captures a ‘snapshot’ of **M** at any particular moment of execution.

Configuration graph

- **Definition.** A configuration of a TM **M** on input **x**, at any particular step of its execution, consists of
 - (a) the nonblank symbols of its work tapes,
 - (b) the current state,
 - (c) the current head positions.

It captures a 'snapshot' of **M** at any particular moment of execution.



Configuration graph

- **Definition.** A configuration of a TM **M** on input **x**, at any particular step of its execution, consists of
 - (a) the nonblank symbols of its work tapes,
 - (b) the current state,
 - (c) the current head positions.

It captures a ‘snapshot’ of **M** at any particular moment of execution.

State info	Input head index	Work tape head index	b_1	...	$b_{S(n)}$
------------	------------------	----------------------	-------	-----	------------

Note: A configuration **C** can be represented using $O(S(n))$ bits if **M** uses $S(n) = \Omega(\log n)$ space on **n**-bit inputs.

Configuration graph

- **Definition.** A *configuration graph* of a TM M on input x , denoted $G_{M,x}$, is a directed graph whose nodes are all the possible configurations of M on input x . There's an edge from one configuration C_1 to another C_2 , if C_2 can be reached from C_1 by an application of M 's transition function(s).

Configuration graph

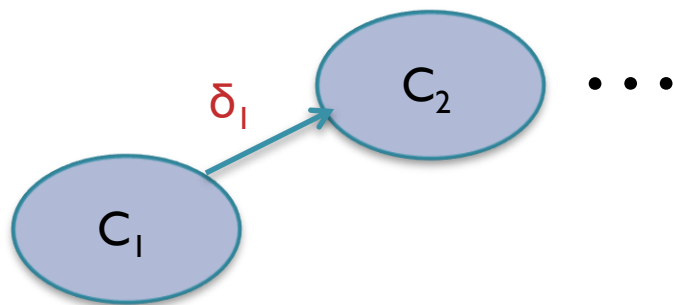
- **Definition.** A *configuration graph* of a TM M on input x , denoted $G_{M,x}$, is a directed graph whose nodes are all the possible configurations of M on input x . There's an edge from one configuration C_1 to another C_2 , if C_2 can be reached from C_1 by an application of M 's transition function(s).
- Number of nodes in $G_{M,x} = 2^{O(S(n))}$, if M uses $S(n)$ space on n -bit inputs

Configuration graph

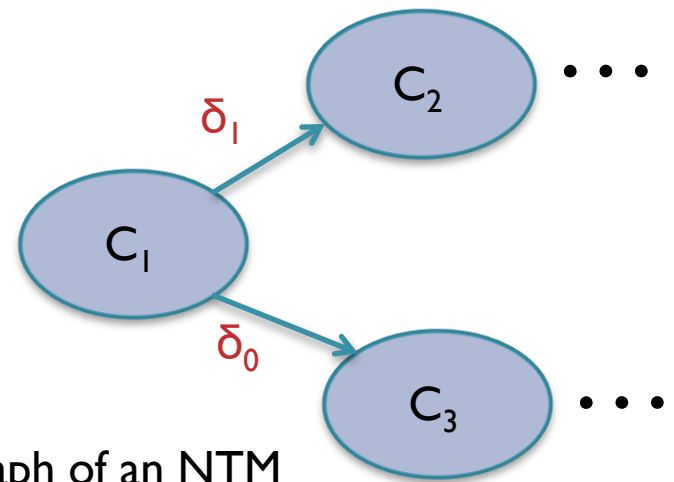
- **Definition.** A *configuration graph* of a TM M on input x , denoted $G_{M,x}$, is a directed graph whose nodes are all the possible configurations of M on input x . There's an edge from one configuration C_1 to another C_2 , if C_2 can be reached from C_1 by an application of M 's transition function(s).
- If M is a DTM then every node C in $G_{M,x}$ has at most one outgoing edge. If M is an NTM then every node C in $G_{M,x}$ has at most two outgoing edges.

Configuration graph

- **Definition.** A *configuration graph* of a TM M on input x , denoted $G_{M,x}$, is a directed graph whose nodes are all the possible configurations of M on input x . There's an edge from one configuration C_1 to another C_2 , if C_2 can be reached from C_1 by an application of M 's transition function(s).



Conf. graph of a DTM



Conf. graph of an NTM

Configuration graph

- **Definition.** A *configuration graph* of a TM M on input x , denoted $G_{M,x}$, is a directed graph whose nodes are all the possible configurations of M on input x . There's an edge from one configuration C_1 to another C_2 , if C_2 can be reached from C_1 by an application of M 's transition function(s).
- By erasing the contents of the work tape at the end, bringing the head at the beginning, and having a q_{accept} state, we can assume that there's a unique C_{accept} configuration. Configuration C_{start} is well defined.

Configuration graph

- **Definition.** A *configuration graph* of a TM M on input x , denoted $G_{M,x}$, is a directed graph whose nodes are all the possible configurations of M on input x . There's an edge from one configuration C_1 to another C_2 , if C_2 can be reached from C_1 by an application of M 's transition function(s).
- M accepts x if and only if there's a path from C_{start} to C_{accept} in $G_{M,x}$.

Relation between time and space

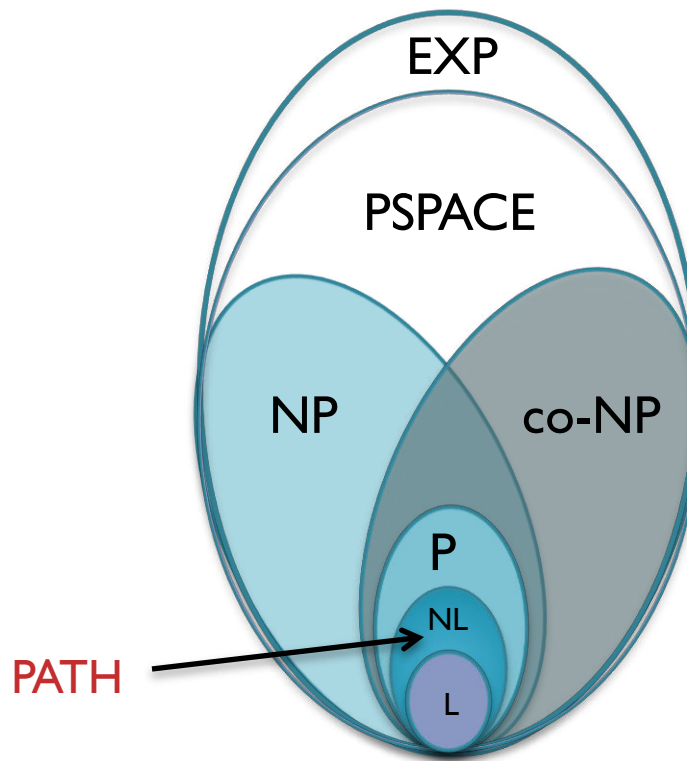
- **Obs.** $\text{DTIME}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n))$.
- **Theorem.** $\text{NSPACE}(S(n)) \subseteq \text{DTIME}(2^{O(S(n))})$, if S is space constructible.
- **Proof.** Let $L \in \text{NSPACE}(S(n))$ and M be an NTM deciding L using $O(S(n))$ space on length n inputs.
- On input x , compute the configuration graph $G_{M,x}$ of M and check if there's a path from C_{start} to C_{accept} . Running time is $2^{O(S(n))}$.

Natural problems?

- Definition.
$$L = \text{DSPACE}(\log n)$$
$$NL = \text{NSPACE}(\log n)$$
$$\text{PSPACE} = \bigcup_{c > 0} \text{DSPACE}(n^c)$$
- Theorem. $L \subseteq NL \subseteq P \subseteq NP \subseteq \text{PSPACE} \subseteq \text{EXP}$.
- Are there natural problems in L , NL and PSPACE ?

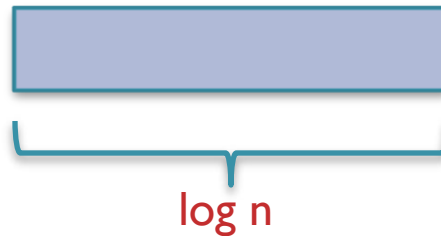
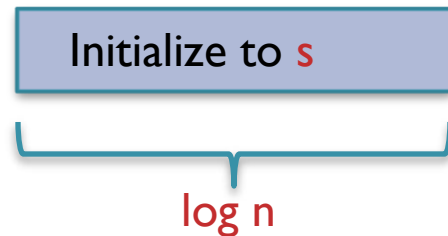
PATH: A canonical problem in NL

- **PATH** = $\{(G,s,t) : G \text{ is a directed graph having a path from } s \text{ to } t\}$.
- Obs. **PATH** is in **NL**.



PATH: A canonical problem in NL

- **PATH** = $\{(G,s,t) : G \text{ is a directed graph having a path from } s \text{ to } t\}$.
- **Obs.** **PATH** is in **NL**.
- **Proof.** Count the no. of vertices in **G**, let it be **n**. Set aside two memory locations of **log n** bits each. Initialize a counter, say **Count = m < n**.



Count = m

PATH: A canonical problem in NL

- **PATH** = $\{(G,s,t) : G \text{ is a directed graph having a path from } s \text{ to } t\}$.
- **Obs.** **PATH** is in **NL**.
- **Proof.** Count the no. of vertices in **G**, let it be **n**. Set aside two memory locations of **log n** bits each. Initialize a counter, say **Count = m < n**.

Initialize to **s**

Guess a vertex **v₁**

Count = m

If there's a edge from **s** to **v₁**, decrease count by **1**.
Else o/p **0** and stop.

PATH: A canonical problem in NL

- **PATH** = $\{(G,s,t) : G \text{ is a directed graph having a path from } s \text{ to } t\}$.
- **Obs.** **PATH** is in **NL**.
- **Proof.** Count the no. of vertices in **G**, let it be **n**. Set aside two memory locations of **log n** bits each. Initialize a counter, say **Count = m < n**.

Set to v_1

Guess a vertex v_2

Count = m-1

If there's a edge from v_1 to v_2 , decrease count by 1.
Else o/p 0 and stop.

PATH: A canonical problem in NL

- **PATH** = $\{(G,s,t) : G \text{ is a directed graph having a path from } s \text{ to } t\}$.
- **Obs.** **PATH** is in **NL**.
- **Proof.** Count the no. of vertices in **G**, let it be **n**. Set aside two memory locations of **log n** bits each. Initialize a counter, say **Count = m < n**.

Set to v_2

Guess a vertex v_3

Count = m-2

If there's a edge from v_2 to v_3 , decrease count by 1.
Else o/p **0** and stop.

...and so on.

PATH: A canonical problem in NL

- **PATH** = $\{(G,s,t) : G \text{ is a directed graph having a path from } s \text{ to } t\}$.
- **Obs.** **PATH** is in **NL**.
- **Proof.** Count the no. of vertices in **G**, let it be **n**. Set aside two memory locations of **log n** bits each. Initialize a counter, say **Count = m < n**.

Set to v_{m-1}

Set to **t**

Count = 1

If there's a edge from v_{m-1} to **t**, o/p **1** and stop.
Else o/p **0** and stop.

PATH: A canonical problem in NL

- **PATH** = $\{(G,s,t) : G \text{ is a directed graph having a path from } s \text{ to } t\}$.
- **Obs.** **PATH** is in **NL**.
- **Proof.** Count the no. of vertices in **G**, let it be **n**. Set aside two memory locations of **log n** bits each. Initialize a counter, say **Count = m < n**.

Set to v_{m-1}

Set to **t**

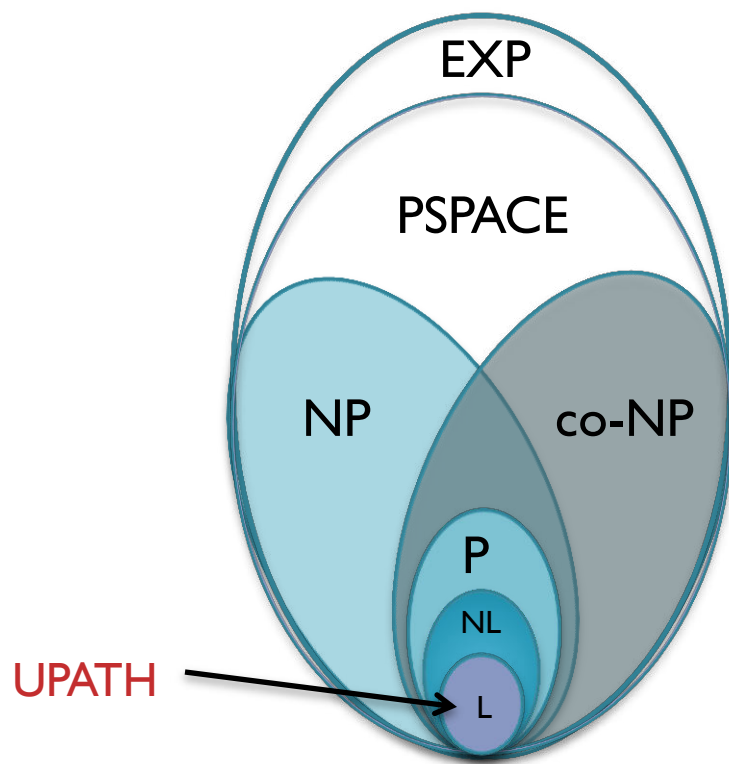
Count = 1

If there's a edge from v_{m-1}
to **t**, o/p **1** and stop.
Else o/p **0** and stop.

Space complexity = $O(\log n)$

UPATH: A problem in L

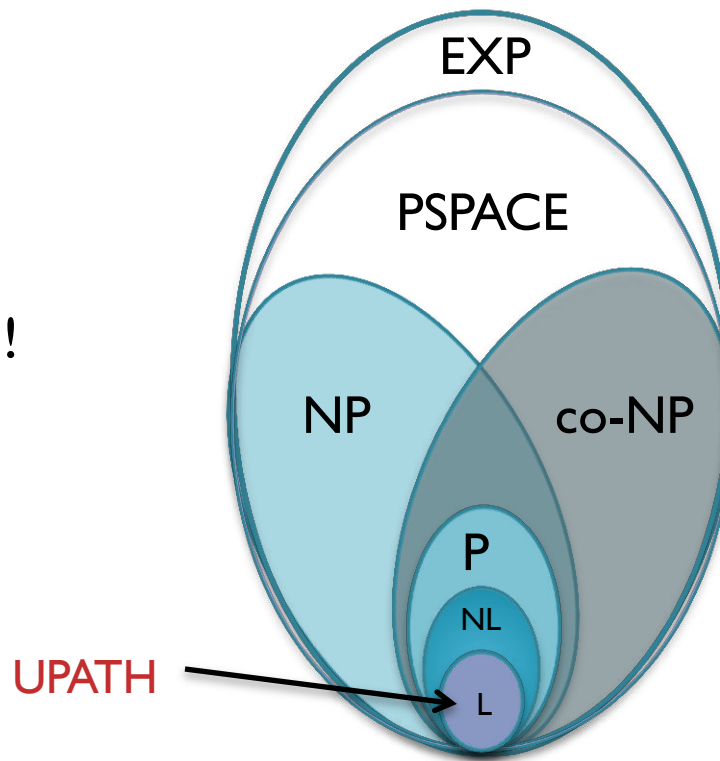
- **UPATH** = $\{(G,s,t) : G \text{ is an undirected graph having a path from } s \text{ to } t\}$.
- **Theorem** (*Reingold 2005*). **UPATH** is in **L**.



UPATH: A problem in L

- **UPATH** = $\{(G,s,t) : G \text{ is an undirected graph having a path from } s \text{ to } t\}$.
- **Theorem** (Reingold 2005). **UPATH** is in **L**.

Is **PATH** in **L** ?
If yes, then **L** = **NL** !
(will prove later)



Space Hierarchy Theorem

- **Theorem.** (*Stearns, Hartmanis & Lewis 1965*) If f and g are space-constructible functions and $f(n) = o(g(n))$, then $\text{SPACE}(f(n)) \subsetneq \text{SPACE}(g(n))$.
- **Proof.** Homework.
- **Theorem.** $L \subsetneq \text{PSPACE}$.