3rd Week

Space Complexity and the Linear Space Hypothesis

Synopsis.

- Circuit Complexity
- Non-Uniform Complexity Classes
- Parameterized Problems
- Sub-Linear-Space Computability
- Linear Space Hypothesis

April 23, 2018. 23:59

Course Schedule: 16 Weeks

Subject to Change

- Week 1: Basic Computation Models
- Week 2: NP-Completeness, Probabilistic and Counting Complexity Classes
- Week 3: Space Complexity and the Linear Space Hypothesis
- Week 4: Relativizations and Hierarchies
- Week 5: Structural Properties by Finite Automata
- Week 6: Stype-2 Computability, Multi-Valued Functions, and State Complexity
- Week 7: Cryptographic Concepts for Finite Automata
- Week 8: Constraint Satisfaction Problems
- Week 9: Combinatorial Optimization Problems
- Week 10: Average-Case Complexity
- Week 11: Basics of Quantum Information
- Week 12: BQP, NQP, Quantum NP, and Quantum Finite Automata
- Week 13: Quantum State Complexity and Advice
- Week 14: Quantum Cryptographic Systems
- Week 15: Quantum Interactive Proofs
- Week 16: Final Evaluation Day (no lecture)

YouTube Videos

- This lecture series is based on numerous papers of T. Yamakami. He gave conference talks (in English) and invited talks (in English), some of which were video-recorded and uploaded to YouTube.
- Use the following keywords to find a playlist of those videos.
- YouTube search keywords:

  Tomoyuki Yamakami conference invited talk playlist

Conference talk video

Main References by T. Yamakami

- ✎ T. Yamakami. Uniform-circuit and logarithmic-space approximations of refined combinatorial optimization problems. In Proc. of COCOA 2013, Lecture Notes in Computer Science, vol. 8287, pp. 318-329 (2013)
- ✎ T. Yamakami. The 2CNF Boolean formula satisfiability problem and the linear space hypothesis. In Proc. of MFCS 2017, LIPIcs 83, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik 62:1-62:14 (2017)
- ✎ T. Yamakami. Parameterized graph connectivity and polynomial-time sub-linear-space short reductions (preliminary report). In Proc. of RP 2017, Lecture Notes in Computer Science, vol. 10506, pp. 176-191 (2017)

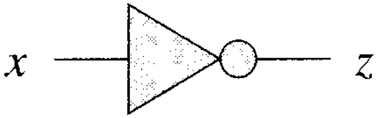
I. Non-Uniform Complexity Classes

1. Boolean Circuits
2. Families of Circuits and Complexity Measures
3. Polynomial-Size Circuits and Non-Uniformity
4. Complexity Class P/poly
5. Advice and Advised Computation
6. Advice Characterization of P/poly
7. Basic Properties of P/poly
8. Complexity Class L/poly

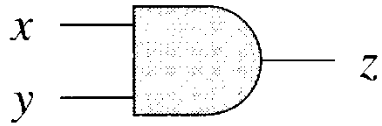
Boolean Circuits

- A Boolean circuit is composed of logical gates and wires (or edges) as illustrated below.

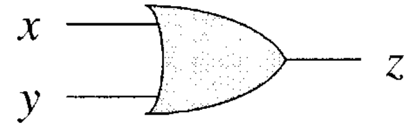
NOT gate

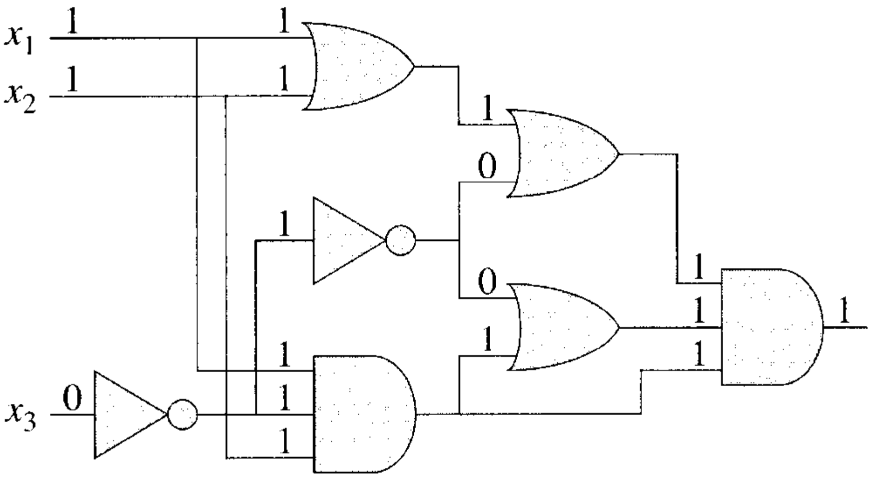


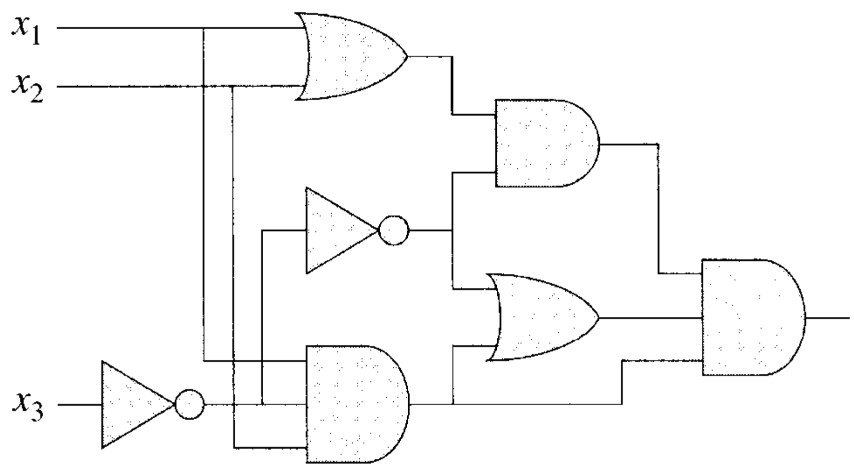
AND gate



OR gate







Truth Assignments of Boolean Circuits

A truth assignment ($x_1 = 1$, $x_2 = 1$, $x_3 = 0$) causes the output to be 1.

Circuit-SAT

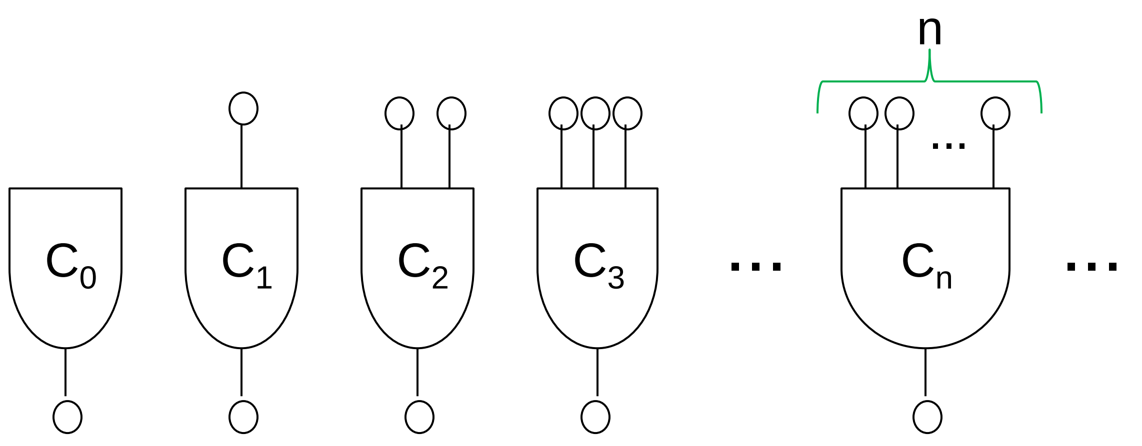
- A Boolean circuit C is called satisfiable if there exists a truth assignment by which C outputs 1.

- We consider the following problem, called Circuit-SAT.
- Circuit Satisfiability Problem (Circuit-SAT)
  - instance: a Boolean circuit C
  - question: is C satisfiable?

- (Claim) Circuit-SAT is NP-complete.
- Hence, Circuit-SAT $\in$ P $\Leftrightarrow$ P = NP.

Families of Circuits and Complexity Measures

- We consider a family $\{C_n\}_{n\in N}$ of Boolean circuits, where each C_n denotes a Boolean circuit taking n-bit inputs.



We treat inputs and outputs as "gates" of indegree 0 and outdegree 0, respectively.

- We use the following complexity measures for circuits.

- Circuit complexity measures:
  - size of circuit C = number of gates in C
  - depth of circuit C = number of logical gates in the longest path from an input to an output

Polynomial-Size Circuits and Non-Uniformity

- We are interested in non-uniform families of Boolean circuits of polynomial size.
- A family $\{C_n\}_{n\in N}$ of circuits computes (or recognizes) language L if, for any $n\in N$ and for any Boolean values $x=x_1x_2...x_n\in\{0,1\}^n$, $x\in L \Leftrightarrow C_{|x|}(x) = 1$.

- A family $\{C_n\}_{n\in N}$ of circuits is said to be of polynomial size if there exists a nonnegative polynomial p such that, for any length n, C_n has size at most p(n).

- We say that a family $\{C_n\}_{n\in N}$ of circuits is non-uniform if there is no specific algorithm to produce a description (or an encoding) of C_n from input 1^n for every $n\in N$.

Complexity Class P/poly

- P/poly is the collection of all decision problems (or languages) computed by non-uniform families of Boolean circuits of polynomial size.

Such a family is called a non-uniform family of circuits

More formally:

- For any decision problem L,

  $L \in$ P/poly $\Leftrightarrow$ there exist a constant $k \geq 1$ and a non-uniform family $\{ C_n \mid n \geq 1 \}$ of Boolean circuits such that

  1) $\text{Size}(C_n) = O(n^k)$ for every $n \geq 1$, and
  2) $x \in L \leftrightarrow C_n(x) = 1$ for every x of length n.

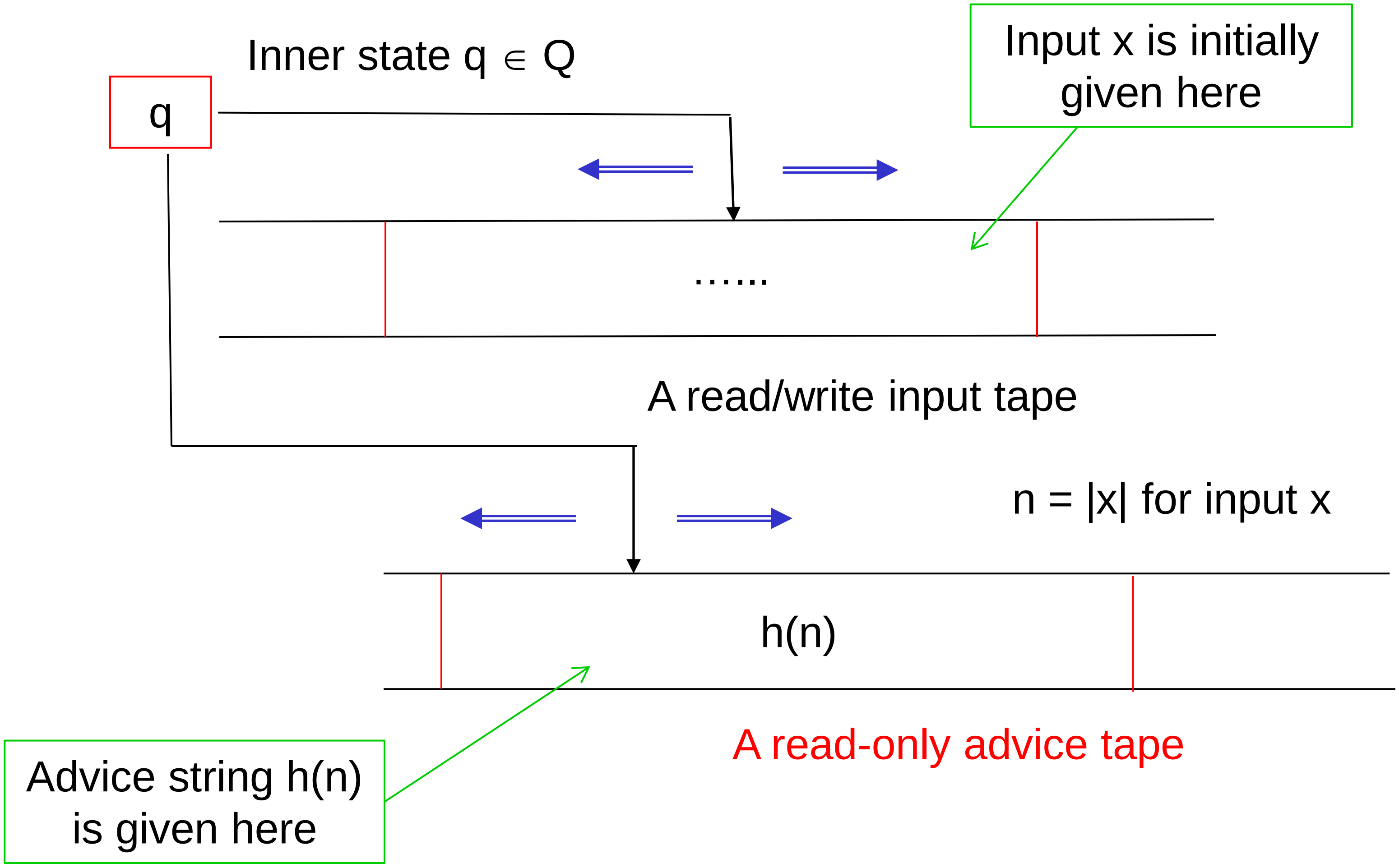
Advice and Advised Computation

- P/poly can be characterized in terms of advice.
- Advice is an external source that can provides with additional information to an underlying machine.
- Karp and Lipton (1982) considered the situation where a single advice string is given to underlying machines for each input length n.
- An advice function $h: N \rightarrow \Sigma^*$ provides advice strings h(n) for each input length n.
- An advised machine is a machine equipped with a read-only advice tape and it takes two types of inputs, a standard input string and also an advice string.

(See the next slide.)

Read-Only Advice Tapes

We provide a machine with an extra read-only advice tape.



Advice Characterization of P/poly I

- We give another characterization of P/poly using advised computation.
- We consider only advice of polynomial length (or size).

- For any decision problem L,

L is in P/poly $\Leftrightarrow$ there exist a constant $k \geq 1$, a polynomial-time DTM M, an advice function $h: N \rightarrow \Sigma^*$ for an advice alphabet such that

1) $|h(n)|=O(n^k)$ for every input length n, and
2) for every x, $x \in L \leftrightarrow$ M accepts input pair $(x,h(|x|))$

This is expressed as $M(x,h(|x|)) = 1$

Advice Characterization of P/poly II

- In other words, for every language L,

L is in P/poly $\Leftrightarrow$ there exist a language A $\in$ P and an advice function h: $N \to \Sigma^*$ such that, for every x,

$$x \in L \leftrightarrow \langle x,h(|x|)\rangle \in A.$$

This is an encoding pair of (x,h(|x|)).

- By changing "P" in the above definition with other complexity classes C, we can define other advised complexity classes C/poly.
- For example, we obtain NP/poly, BPP/poly, UP/poly, etc.

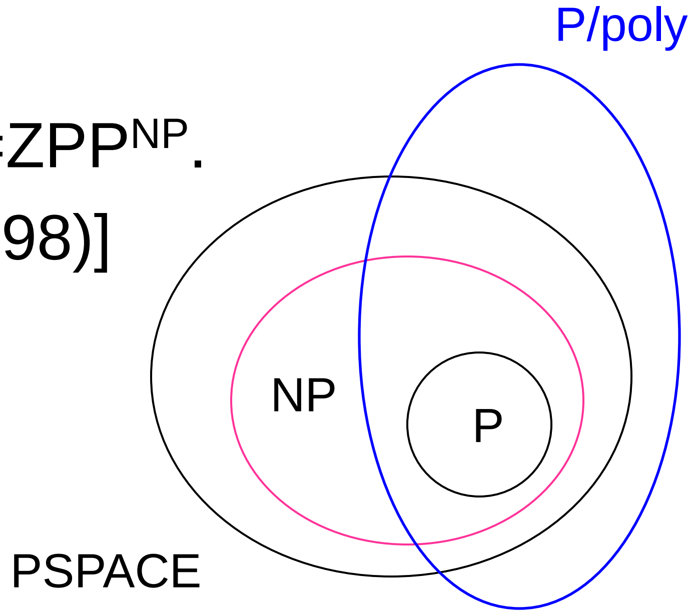
(*) UP will be discussed in Week 4.

Basic Properties of P/poly

- Note that P/poly contains non-recursive problems (that is, problems that cannot be solved by any algorithm) because advice functions may not generally be computable.

- (Claim) BPP $\subseteq$ P/poly
- (Claim) If NP $\subseteq$ P/poly, then PH=ZPP^{NP}.
  [Köbler-Watanabe (1998)]

- Open Problems:
  - Does NP $\subseteq$ P/poly?
  - Does PSPACE $\subseteq$ P/poly?



Many researchers believe in this way.

Complexity Class L/poly

- The use of advice gives rise to many non-uniform complexity classes.
- Here, we introduce another complexity class L/poly using log-space DTMs with polynomial-size advice.
- Let S be any decision problem or a language.

S is in L/poly $\Leftrightarrow$ there exist a constant $k \geq 1$, a log-space DTM M, an advice function $h: N \to \Sigma^*$ such that

1) $|h(n)| = O(n^k)$ for every input length n, and
2) for any x, $x \in S \leftrightarrow M(x,h(|x|)) = 1$.

- (Claim) $L \subseteq L/poly$
- Open Problem: Is $NL \subseteq L/poly$?

Log-space DTMs will be explained shortly.

II. Space-Bounded Computation

1. Space-Bounded Computation
2. Polynomial-Space Solvable Problems
3. Complexity Class PSPACE
4. Random Access Model with Index Tapes
5. Logarithmic-Space Solvable Problems
6. Complexity Class L
7. Complexity Class NL
8. Function Class FL
9. Log-Space Many-One Reductions

Space-Bounded Computation

- Earlier, we have discussed time-bounded computation and time-bounded solvable problems.
- Here, we are focused on space-bounded computation and associated problems.
- Let s be a space-bounding function from N to N such that $s(n) \geq \log(n)$ for all $n \geq 1$.
- We say that an algorithm (i.e., a DTM) solves a (decision) problem using space $O(s(n))$ if, when it is provided a problem instance x of length n, the algorithm can produce the solution using $O(s(n))$ space.
- There is no bound for running time but the algorithm must halt eventually.

Polynomial-Space Solvable Problems

- A problem is said to be s(n)-space solvable if there exists an algorithm to solve it using space O(s(n)).
- When s(n) is a polynomial (i.e., $s(n)=O(n^k)$ for some constant k), a problem is said to be polynomial-space solvable.
- Note that any algorithm that runs in time t(n) also uses space at most t(n).
- Hence, polynomial-time solvable problems are also polynomial-space solvable.
- However, the converse does not hold in general.

How to Solve NP-Complete Problems

- Using polynomial-space, we can easily solve NP-complete problems.
- Take Circuit-SAT as an example of NP-complete problems.
- Consider this algorithm. ⇒

- This algorithm uses only O(n) bits to remember v and O(|code(C)|^2) bits to simulate C(v).
- Hence, Circuit-SAT is polynomial-space solvable.

Algorithm for Circuit-SAT

1. Take Boolean circuit C as an input.
2. Set v=0^n.
3. Check if C(v) = 1.
4. If so, accept and halt.
5. Else, if v = 1^n, reject and halt.
6. Else, increment v by one and go to Step 3.

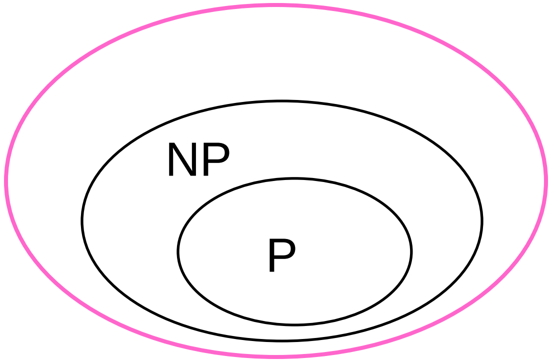
v = 00……00
00……01
00……10
⋮
11……11

increment ↓

Complexity Class PSPACE

- We introduce a complexity class defined by deterministic polynomial-space computations.
- A decision problem L is in PSPACE if there is a DTM M such that, for any input x,
  1. $x \in L \rightarrow$ M accepts x,
  2. $x \notin L \rightarrow$ M rejects x, and
  3. M uses polynomial space.

PSPACE = NPSPACE



- (Claim) $P \subseteq NP \subseteq PSPACE \subseteq EXP$.
- (Claim) PSPACE = NPSPACE. [Savitch (1970)]

Function Class FPSPACE

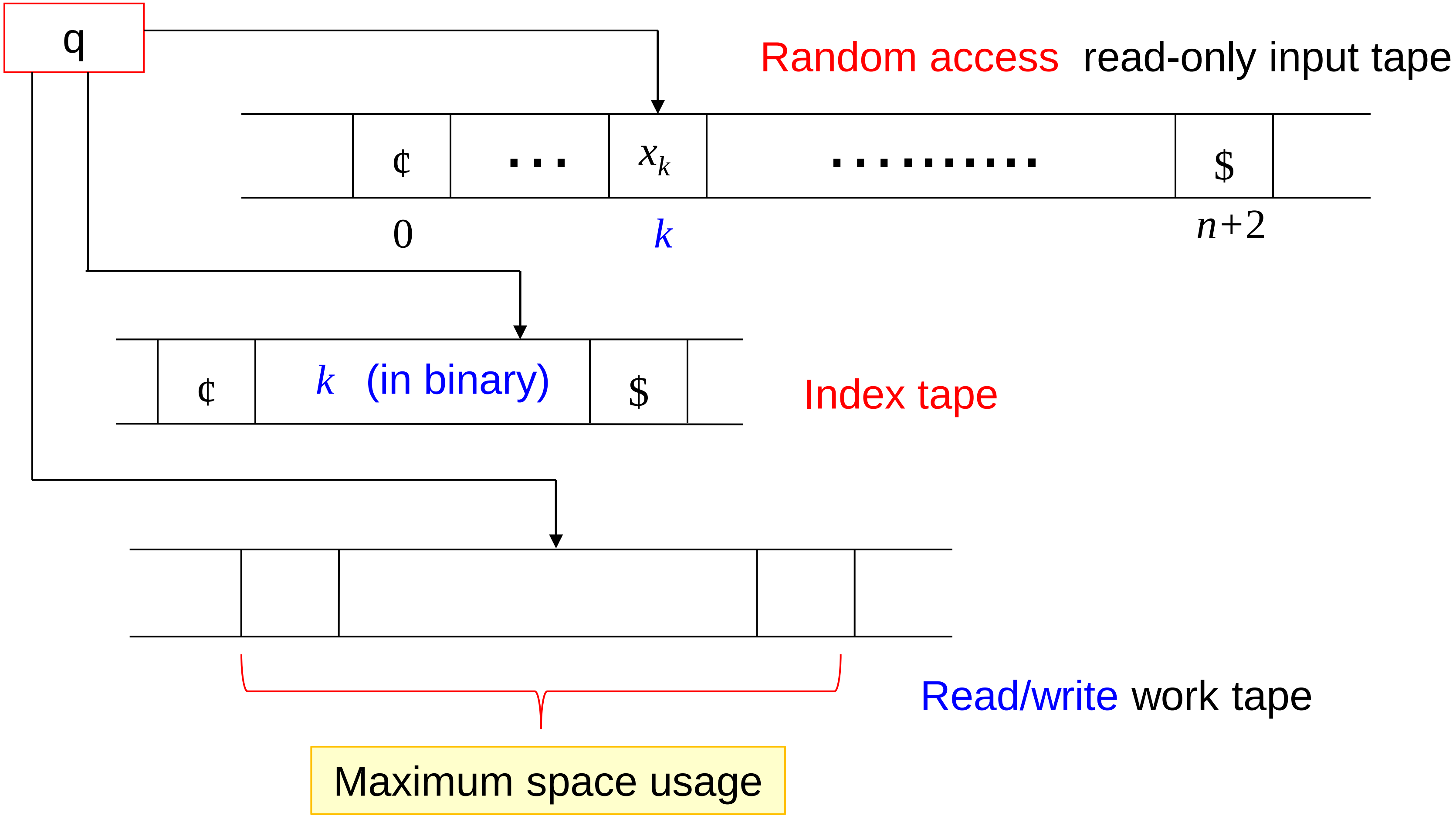
- Next, we consider functions f : Σ* → Σ* (where Σ is an alphabet).

- A function f : Σ* → Σ* is in FPSPACE ⇔
  1. f is p-bounded (i.e., |f(x)|=O(|x|k) for some k≥1, and
  2. there is a DTM M such that, for any input x, M produces f(x) on the output tape using space O(log(|x|)).

- (Claim) PF ⊆ FPSPACE.

Random Access Model with Index Tapes

- To consider logarithmic-space computation, we need a random access model of multi-tape Turing machines.



How to Operate a Machine

- To read each symbol written on an input tape, we need to take a series of steps described below.

1. A machine M writes down an index k in binary on the index tape.
2. M enters a special state, called an index state q_{index}, to initiate the process of random accessing.
3. An input-tape head of M jumps to the cell indexed k.
4. M scans the k-th tape cell and then the index tape is automatically become empty.

- This process is repeatedly taken to read all or some input symbols.

Logarithmic-Space Solvable Problems

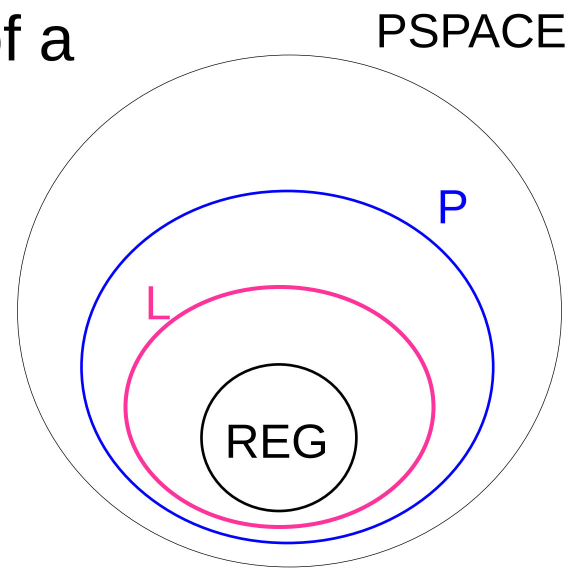
- Here, when we consider the space usage of a machine, we do not include any read/write work tape.

- Let s be a function from N to N.
- We say that an algorithm (or a deterministic Turing machine) solves a problem A using space O(s(n)) if, for any instance x of length n, the algorithm can produce a solution of A using O(s(n)) space.

- A problem is logarithmic-space (or log-space) solvable if there exists an algorithm to solve it using O(log n) space.

Complexity Class L

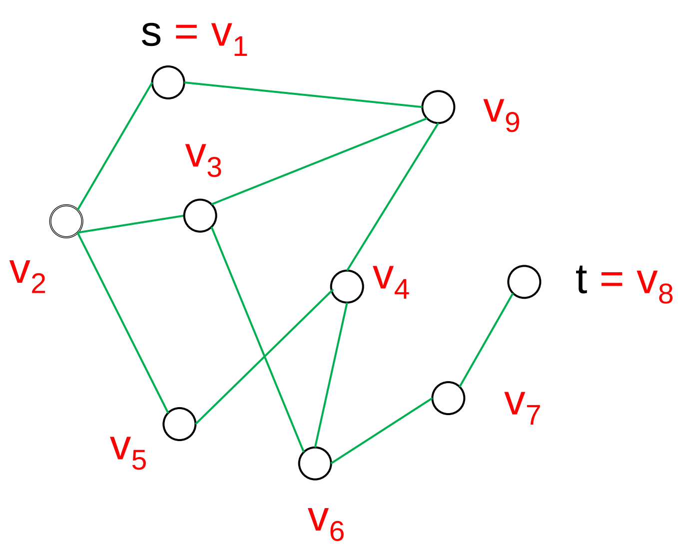
- A decision problem (or a language) A is in L if there is a DTM M such that, for any input x,
  1. $x \in A \rightarrow$ M accepts x,
  2. $x \notin A \rightarrow$ M rejects x, and
  3. M uses logarithmic space (or log space).
- It is possible to trim the running time of a machine to "polynomial time."
- (Claim) REG $\subseteq$ L $\subseteq$ P.
- (Claim) L $\neq$ PSPACE. [Savitch (1970)]

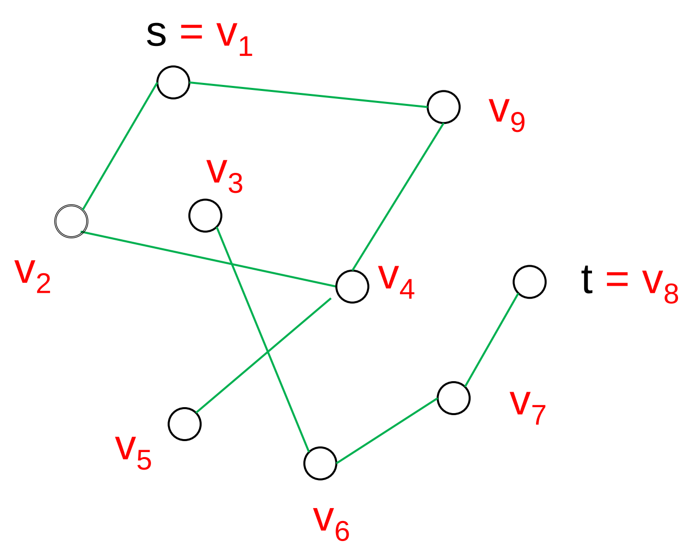


USTCON: Typical Problem in L

- Complexity class L contains the following problem.

- Undirected s-t Connectivity Problem (USTCON)
  - instance: an undirected graph G and two vertices s,t
  - question: Is there a path between s and t?

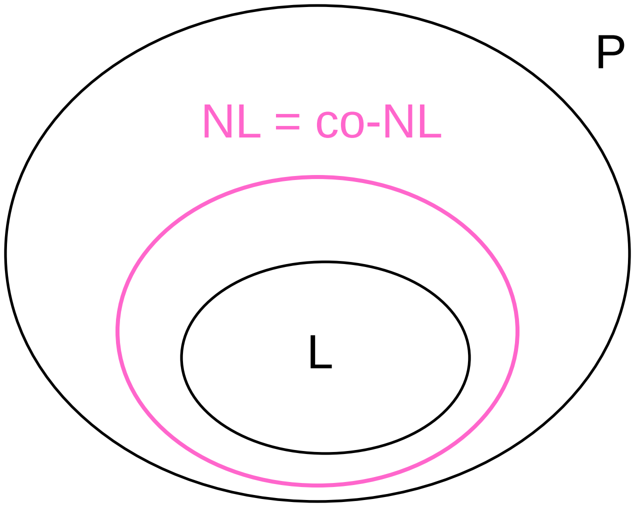




Complexity Class NL

- A decision problem (or a language) L is in NL if there is an NTM (nondeterministic Turing machine) M such that, for any input x,
  1. $x \in L \leftrightarrow$ there exists an accepting computation path of M on x (or x is accepted by M), and
  2. M uses logarithmic space (or log space) on all inputs.

- (Claim) $L \subseteq NL \subseteq P$
- (Claim) NL = co-NL
  [Immerman (1988),
  Szelepcsényi (1988)]
- Hence, NL looks different from NP.



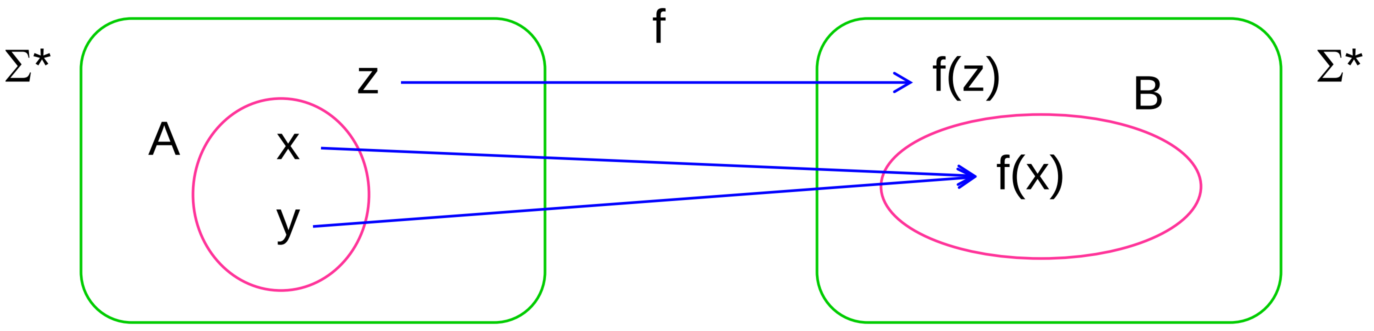
Function Class FL

- Let us recall the function class FP from Week 1.
- Here, we consider a log-space version of FP.
- Let f: $\Sigma^* \to \Sigma^*$ be any function, where Σ is an alphabet.
- This function f: $\Sigma^* \to \Sigma^*$ is called log-space computable if
  1. f is p-bounded (i.e., $|f(x)|=O(|x|^k)$ for some $k>0$),
  2. there exists a DTM M with an output tape such that, on each input $x \in \Sigma^*$, M produces f(x) on its output tape, and
  3. on input x, M uses only O(log(n)) space on the work tape (but no space bound is imposed on the output tape).
- Let FL denote the collection of all p-bounded log-space computable functions.

Log-Space Many-One Reductions

- A language A is log-space many-one reducible (L-m-reducible or $\leq_m^L$-reducible) to language B if there exists a log-space DTM M such that, for any input x,
  - o $x \in A \Leftrightarrow$ M on input x produces y, which is p-bounded, and $y \in B$.
- In this case, we write $A \leq_m^L B$.
- In other words,

$$A \leq_m^L B \Leftrightarrow \exists f \in FL\ \forall x \in \Sigma^* [\ x \in A \leftrightarrow f(x) \in B\ ]$$



2SAT is NL-Complete

- Recall from Week 2 that 3SAT is NP-complete.
- Complexity class NL contains the following problem.

- 2-Satisfiability Problem (2SAT)
  - instance: a Boolean formula φ of 2CNF (2-conjunctive normal form)
  - question: Is φ satisfiable?

2 literals

- E.g., 2CNF: $\varphi \equiv (x_1 \vee x_2) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_2 \vee x_3)$

- (Claim) 2SAT is NL-complete. [Jones (1975)]

$2SAT_k$ is also NL-Complete

- It turns out that 2SAT is not suitable for our purpose.
- Thus, we consider a restricted variant of 2SAT.

- $2SAT_k$ is the set of all 2SAT formula, each variable of which appears as literals at most k times.
- Example: k=3

  $\varphi \equiv (x_1 \vee \neg x_6) \wedge (x_2 \vee x_3) \wedge (\neg x_5 \vee x_2) \wedge (\neg x_4 \vee \neg x_2)$

  $m_{vbl}(\varphi) = 6$, $m_{cls}(\varphi) = 4$

Each x_i appears at most 3 times

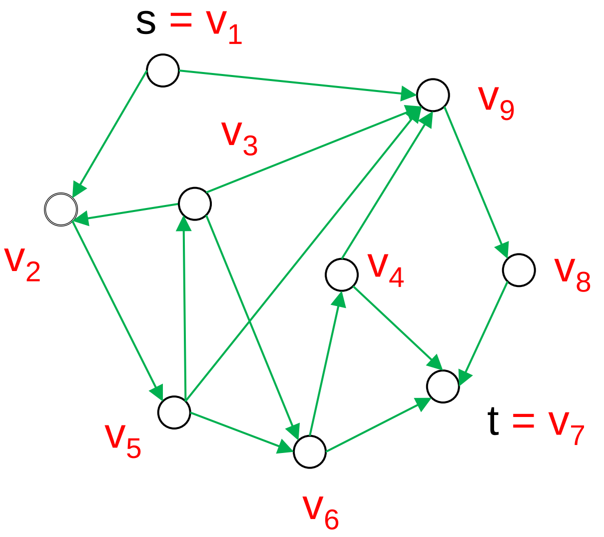
- (Claim) $2SAT_k$ ($k \geq 3$) is NL-complete.
- However, it is not known that $2SAT_k \notin L$.

DSTCON is NL-complete

- Complexity class NL contains the following problems.

- Directed s-t Connectivity Problem (DSTCON)
  - instance: a directed graph G and two vertices s,t
  - question: Is there a path from s to t?

- (Claim) DSTCON is NL-complete. [Jones (1975)]



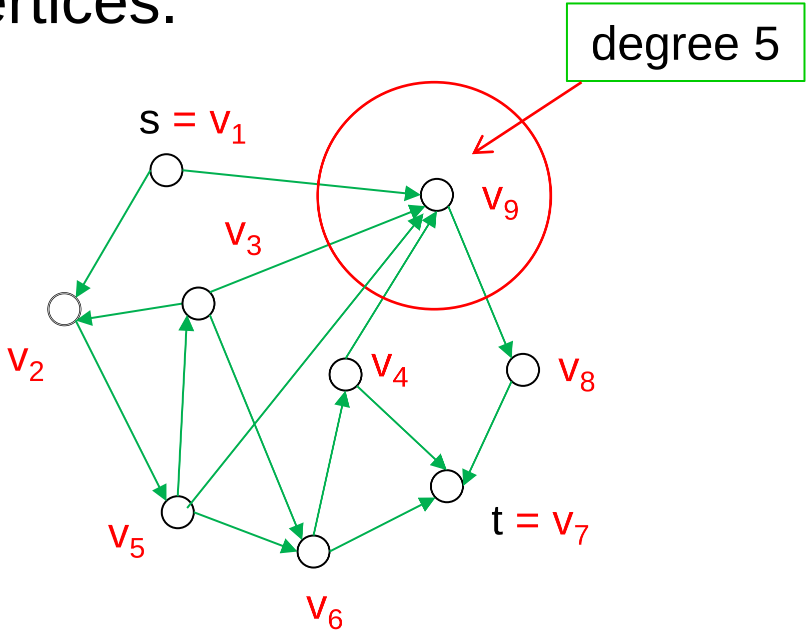
kDSTCON is also NL-complete

- Consider a restricted variant of DSTCON.
- The degree of a vertex (or a node) is the number of edges connected to the vertex.

- kDSTCON consists of DSTCON instances whose graphs have degree at most k at all vertices.

- (Claim) For any constant $k \geq 3$, kDSTCON is NL-complete.

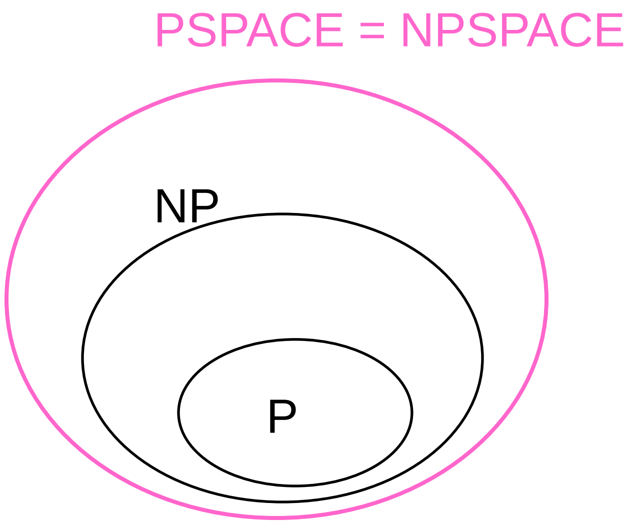
- However, it is not known that 3DSTCON $\notin$ L.

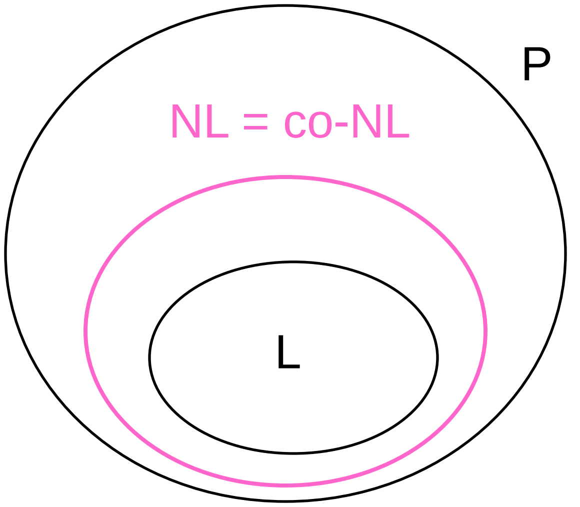


Open Problems

- The following questions regarding L, NL, and PSPACE are not yet answered.

  - Is P = PSPACE?
  - Is NP = PSPACE?
  - Is L = P?
  - Is L = NL?
  - Is NL = NP?
  - Is CFL $\subseteq$ L?





III. Sub-Liner-Space Computability

1. Space Usage for Solving DSTCON
2. Parameterized Problems
3. Size Parameter Matters
4. Poly-Time Sub-Linear-Space Computability
5. Complexity Class PsubLIN
6. Oracle Turing Machines
7. SLRF-T-Reducibility
8. Short SLRF-T-Reducibility
9. Relationships by Short Reductions

Space Usage for Solving DSTCON

- Consider the following directed s-t connectivity problem.
- DSTCON(m,n)
  - instance: a directed graph G of n vertices and m edges, and two vertices s, t
  - question: is there any path from s to t?
- Barnes, Buss, Ruzzo, and Schieber (1998) gave an algorithm that solves DSTCON(m,n) in O(m+n) time using $n^{1-c/\sqrt{\log(n)}}$ space for an appropriate constant c>0.
- Open Problem:

  Can we improve the above space bound down to

  $O(n^{\varepsilon}$ polylog(m+n)) for certain $\varepsilon \in [0,1)$?

Size Parameters

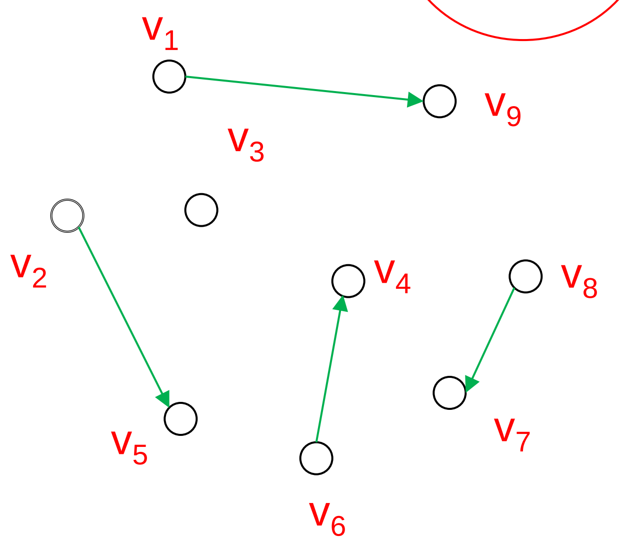
- It is useful to parameterize problems by taking appropriate size parameters.
- A size parameter m: $\Sigma^* \rightarrow N$ is a function that gives a "size" m(x) of an input x (e.g., m(x) = |x|).
- Here are 2 simple examples.
- For a CNF Boolean formula φ:
  - $m_{vbl}(\varphi)$ = number of different variables in φ
  - $m_{cls}(\varphi)$ = number of clauses in φ
- For a directed/undirected graph G:
  - $m_{ver}(G)$ = number of vertices in G
  - $m_{edg}(G)$ = number of edges in G

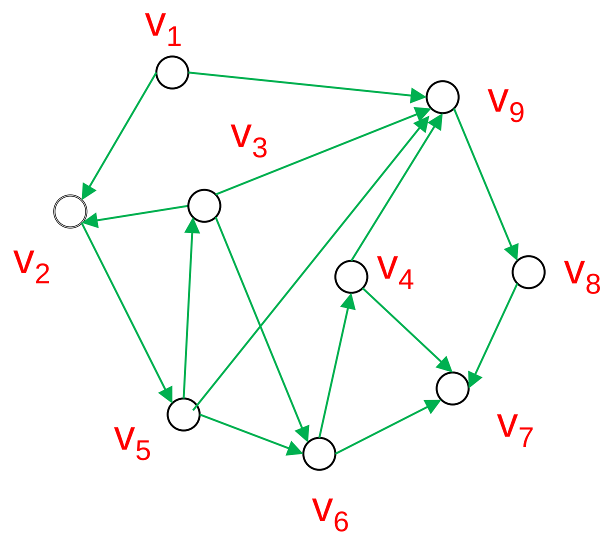
Size Parameter Matters

- How different is the gap between $m_{ver}(G)$ and $m_{edg}(G)$?
  - ✓ $m_{ver}(G)$ = number of vertices in G
  - ✓ $m_{edg}(G)$ = number of edges in G

$$m_{ver}(G) \le 2m_{edg}(G)$$ if there is no isolated vertex

$$m_{edg}(G) \le m_{ver}(G)^2$$ A huge difference!





Parameterized Problems

- In practice, execution time and space usage are often measured according to size m(x) of input x.
- Impagliazzo, Paturi, and Zane (2001) took a new approach toward kSAT and Search-kSAT, parameterized by m_{vbl} and m_{cls}.
- A parameterized decision problem is a pair (A,m) of a (standard) decision problem $A \subseteq \Sigma^*$ and a size parameter m: $\Sigma^* \rightarrow N$.
- Parameterized decision problem (A,m)
  - instance: x with size m(x)
  - question: is $x \in A$?

Poly-Time Sub-Linear-Space Computability

- We use deterministic Turing machines (DTMs), each of which has an input tape and a work tape.
- We are interested in DTMs that use only $O(n^c)$ time and restricted space to solve given decision problems.
- Let m be a size parameter.
- An informal term "sub linear w.r.t. m" means

$$m(x)^{\varepsilon}\ \mathrm{polylog}(|x|)$$

for a fixed constant $\varepsilon \in [0,1)$ and a polylogarithmic function polylog(n) (i.e., $c\log^k(n)+d$ for some $c>0$ and $k \geq 0$).

- Here, we are focused on deterministic algorithms that run in polynomial time using only sub-linear space.

PTIME,SPACE(...)

- It is useful in practice to introduce a new notation.
- Let (L,m) denote a parameterized problem, where L is a decision problem and m is a size parameter.

- (L,m) $\in$ PTIME,SPACE(f(n)) $\Leftrightarrow$

  $\exists$M:DTM s.t. $\forall$x

1) $x \in L \rightarrow$ M accepts x
2) $x \notin L \rightarrow$ M rejects x
3) M runs in time polynomial in |x| using O(f(m(x))) space.

O(|x|^k) time for some k>0

Complexity of 2SAT(m,n)

- Theorem: [Yamakami (2017)]
  $\exists c>0\ \exists l$:polylog function s.t.
  $$2SAT(m,n) \in PTIME,SPACE(n^{1-c/\sqrt{\log(n)}}\, l(m+n)),$$
  where a 2SAT(m,n)-instance has n variables and m clauses.

- The proof of the above theorem follows directly from Barnes, Buss, Ruzzo, and Schiebe's (1998) fast algorithm for DSTCON.

- Open Problem:
  Is it true that 2SAT(m,n) $\in$PTIME,SPACE(n^{ε}) for a constant $\varepsilon\in(0,1)$?

Complexity Class PsubLIN

- We define a new practical complexity class called PsubLIN.
  - ✓ "P" stands for "polynomial-time."
  - ✓ "subLIN" stands for "sub-linear space."
- PsubLIN = class of (parameterized) decision problems or search problems (L,m) such that L is solved in time polynomial in |x| using sub-linear space (w.r.t. m)
- That is,

$$\mathrm{PsubLIN} = \cup_{0\leq\varepsilon<1} \mathrm{PTIME,SPACE}(m(x)^{\varepsilon}\mathrm{polylog}(|x|))$$

- (Claim) $L \subseteq \mathrm{PsubLIN} \subseteq P$.

Reductions for Parameterized Problems

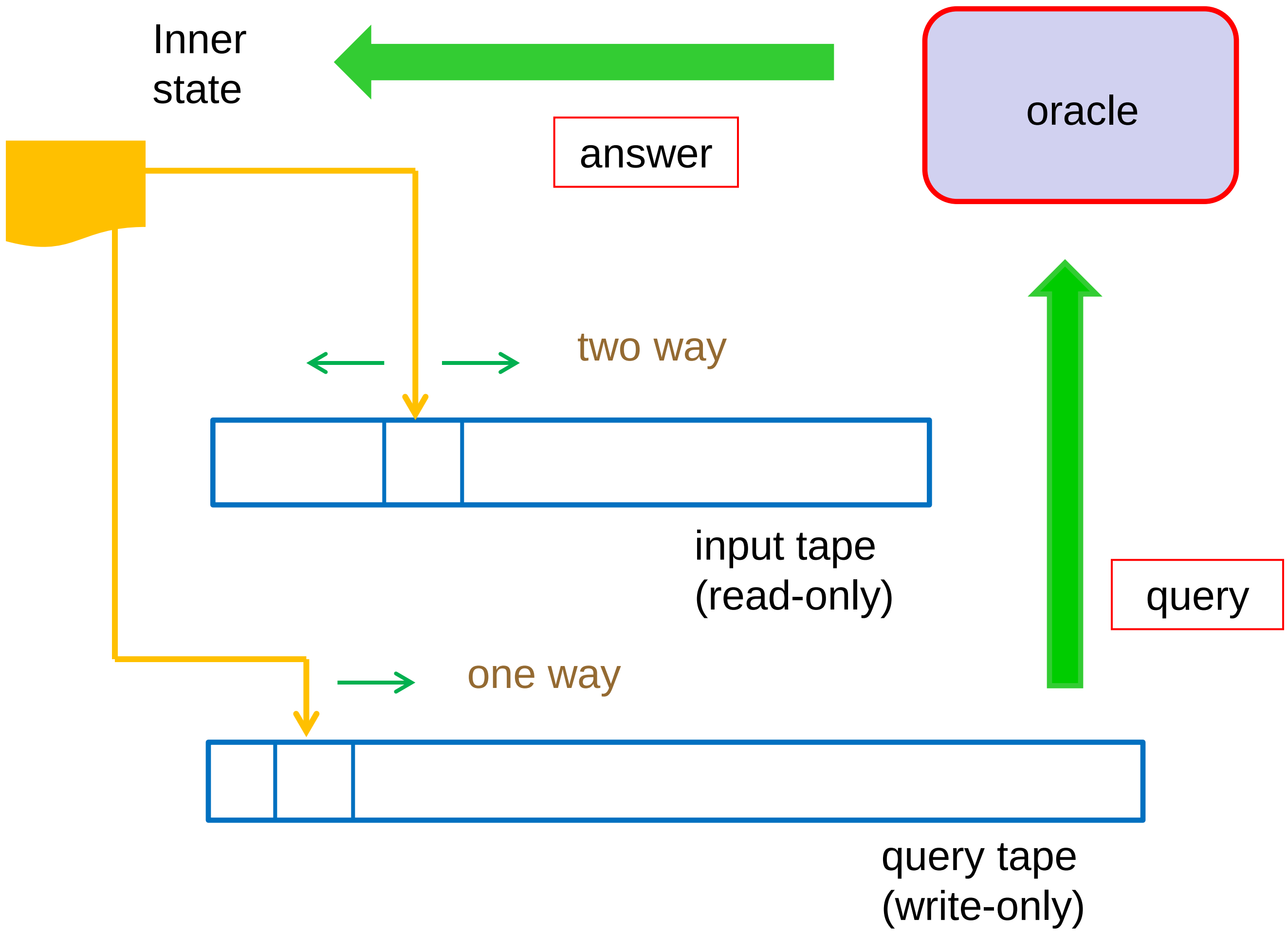
- Reductions or reducibility has been so successful to discuss “complete” problems, such as NP-complete problems.
- Now, our goal is to define suitable reductions among parameterized problems in PsubLIN.
- First of all, for a wider rage of application, we expand “many-one reduction” to “Turing reduction.”
- To define Turing reduction, we need to introduce a notion of oracle Turing machine and a notion of oracle.

Oracle Turing Machines I

- Here, we give briefly general notions of oracle Turing machine and oracle.
- (*) The notion of OTMs will be discussed extensively in Week 4.

- An oracle Turing machine (OTM) is equipped with an additional tape, called a query tape, in which the machine make a query to an oracle.
- An oracle is an external information source, which can provide the machine with necessary information via a process of query and answer.

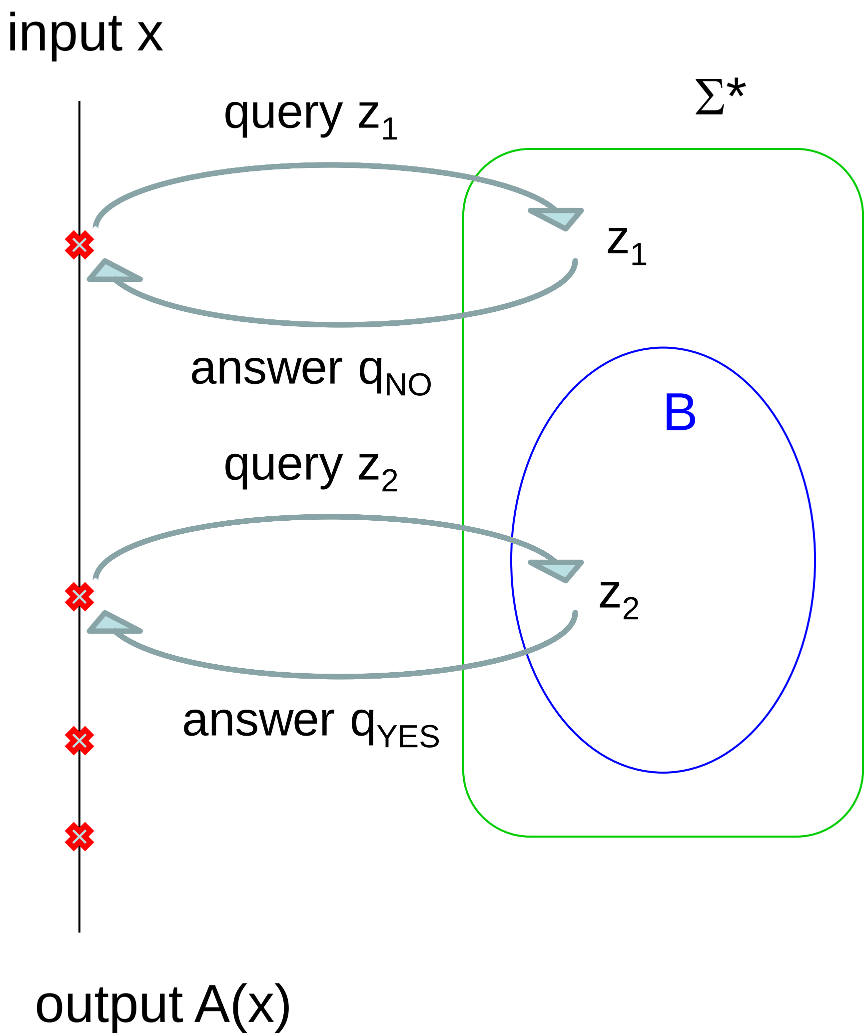
Oracle Turing Machines II



Oracle Computation

- M: OTM for A
- B: oracle

1. M starts with input x.
2. Whenever M writes a query word z on its query tape and enters a query state q_{query}, z is automatically sent to B.
3. The oracle B returns its answer (YES/NO) by changing M's inner state to either q_{yes} or q_{no}.
4. M resumes its computation, starting with q_{yes} or q_{no}.
5. If M halts, output M(x). Otherwise, go to Step 2.



SLRF-T-Reducibility

- We define a notion of (polynomial-time) sub-linear-space reduction family (SLRF).

- $(P_1,m_1) \leq^{SLRF}_T (P_2,m_2) \Leftrightarrow$
  $\forall \varepsilon>0 \exists M$:oracle DTM$\exists l$:polylog$\exists k_1,k_2>0$ s.t.
  1. $M^{P2}(x)$ runs in $\leq p(|x|)$ time and $\leq m_1(x)^{\varepsilon} l(|x|)$ space
  2. Whenever M makes a query to oracle P_2, M receives its answer and continues a computation.
  3. If M make a query z to P_2, then $m_2(z) \leq m_1(x)^{k1}+k_1$ and $|z| \leq |x|^{k2}+k_2$.

- All queried words z have size polynomial in the size of inputs (w.r.t. size parameters).

Short Reductions are Needed

- Unfortunately, in SLRF-T-reduction, query words are too long to make functional composition for sub-linear-space machines.
- This raises a serious question whether PsubLIN may not be closed under $\leq_m^L$-reductions.
- This forces us to look for a more restricted notion of reductions to discuss the computational complexity of PsubLIN.
- A simple remedy is to make only “short” queries.
- Namely, we demand that the size of queried word is linear in the size of input (w.r.t. given size parameters).

Short SLRF-T-Reductions

- We say that (P_1,m_1) is short SLRF-T-reducible to (P_2,m_2), denoted by $(P_1,m_1)\leq^{sSLRF}_{T} (P_2,m_2)$, if the following hold.

This bound is different from SFRF-T-reductions

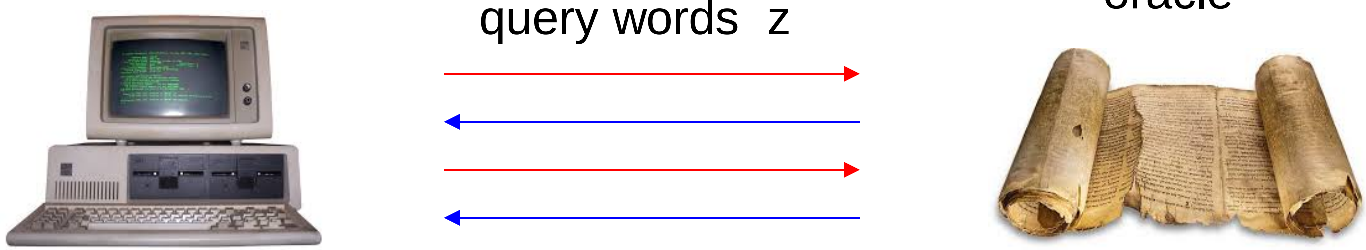
- $(P_1,m_1)\leq^{sSLRF}_{T} (P_2,m_2) \Leftrightarrow$

$\forall\varepsilon>0\exists M$:oracle TM$\exists l$:polylog$\exists k_1,k_2>0$ s.t.

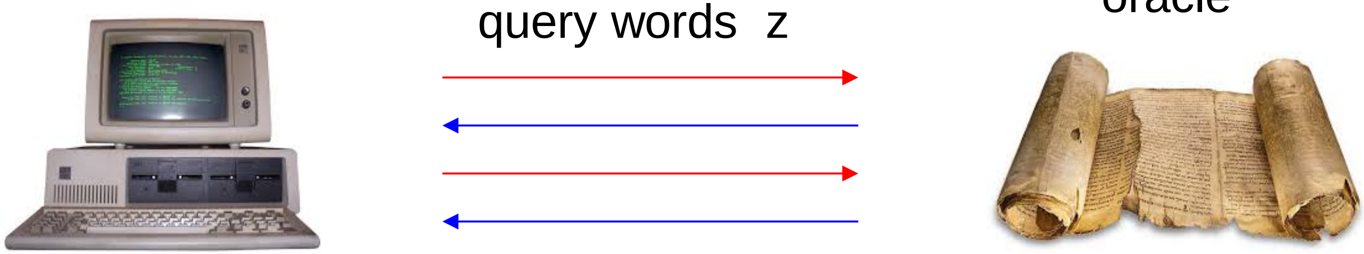
1. $M^{P2}(x)$ runs in $\leq p(|x|)$ time and $\leq m_1(x)^{\varepsilon}l(|x|)$ space
2. Follow the same oracle mechanism
3. If M^{P2} queries z to P_2, then $m_2(z) \leq k_1 m_1(x)+k_1$ and $|z| \leq |x|^{k2}+k_2$.

- $A \equiv_r B \Leftrightarrow A \leq_r B$ and $B \leq_r A$ for any reduction type r

Comparison of Query Size



$$m_2(z) \leq m_1(x)^{k_1} + k_1$$



$$m_2(z) \leq k_1 m_1(x) + k_1$$

Properties of Short Reductions

- Proposition: [Yamakami (2017)]
  1) $\leq^{SLRF}_{T}$ and $\leq^{sSLRF}_{T}$: reflexive and transitive.
  2) PsubLIN is closed under $\leq^{sSLRF}_{T}$-reductions.
  3) $\exists$X,Y: recursive s.t. $X \leq^{SLRF}_{T} Y$ but $X \not\leq^{sSLRF}_{T} Y$.

- Proposition: [Yamakami (2017)]
  $\forall m \in \{m_{vbl}, m_{cls}\}\ \forall k \geq 3$
  1) $(2SAT_k, m) \equiv^{sSLRF}_{m} (2SAT_3, m)$
  2) $(2SAT_3, m_{vbl}) \equiv^{sSLRF}_{m} (2SAT_3, m_{cls})$

However, we don't know if we can replace $2SAT_3$ by 2SAT.

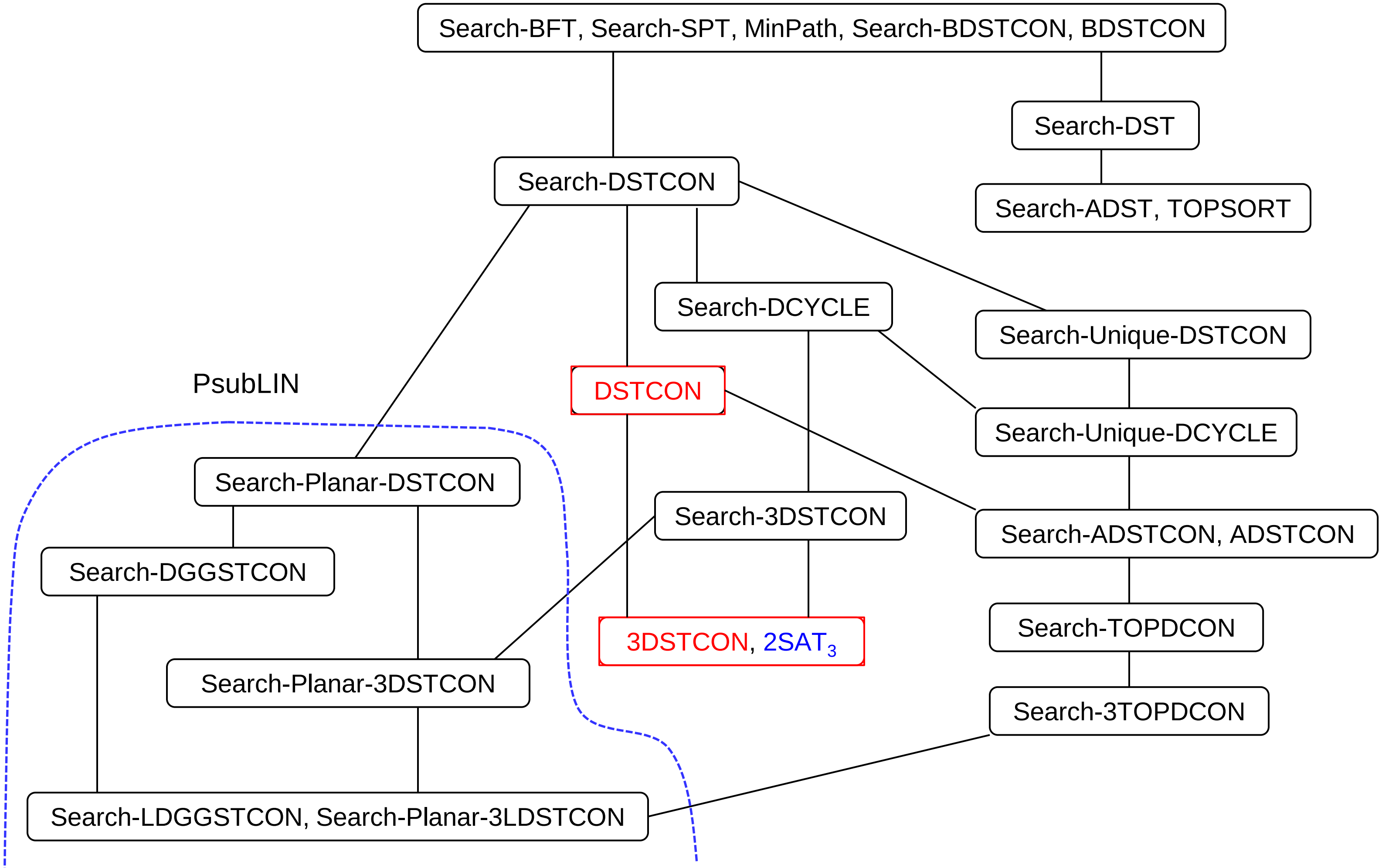
- Hence, it suffices to focus only on $(2SAT_3, m_{vbl})$.

Relationships by Short Reductions

- As a simple example of $\leq^{sSLRF}_{T}$, let us consider the directed s-t connectivity problem (DSTCON) and its variants.
- The next slide will illustrate certain known relationships among numerous variants of DSTCON problems associated with acyclic graph, planar graph, shortest-path, etc.
- (*) In the next slide, "Search-C" means a search problem in which we are asked to find (and output) a solution to the original decision problem C.

Ordered by sSLRF-reductions

Size parameter: $m_{ver}(x)$ = # of vertices



IV. Linear Space Hypothesis

1. New, Practical Working Hypothesis
2. The Linear Space Hypothesis (LSH)
3. Other NL-Complete Problems
4. Other Characterizations of LSH

New, Practical Working Hypothesis

- As noted earlier, 2SAT with n variables and m clauses is solvable in polynomial time using at most

  $$n^{1-c/\sqrt{\log(n)}} \times \text{polylog}(m+n) \text{ space}.$$

- However, we do not know whether 2SAT (even $2SAT_3$) is solved in polynomial time using $n^{\varepsilon} \times$ polylog(m+n) space for a fixed constant $\varepsilon \in [0,1)$.

- We want to propose a new, practical working hypothesis, which is expected to serve as a driving force to obtain better lower bounds of the computational complexity of various problems.

The Linear Space Hypothesis (LSH) I

- We introduce a working hypothesis called the linear space hypothesis (LSH).

- LSH (or LSH for $2SAT_3$) states:

  There is no deterministic algorithm that solves $2SAT_3$ in time $p(|x|)$ using at most $m_{vbl}(x)^{\varepsilon} l(|x|)$ space on instance x for a certain polynomial p, a certain polylog function l, and a certain constant $\varepsilon \in [0,1)$.

- Open Problem

  Prove or disprove that LSH for $2SAT_3 \leftrightarrow$ LSH for 2SAT.

The Linear Space Hypothesis (LSH) II

- The previous definition uses the parameterized problem ($2SAT_3$,m_{vbl}). How about ($2SAT_3$,m_{cls})?

- (Claim) We can replace m_{vbl} in the above by m_{cls}.

❑ Proof Sketch:

This is because ($2SAT_3$,m_{vbl}) $\equiv^{sSLRF}_{m}$ ($2SAT_3$,m_{cls}) and PsubLIN is closed under $\leq^{sSLRF}_{m}$-reductions.

- Theorem: [Yamakmai (2017)]

If LSH for $2SAT_3$ holds, then L $\neq$ NL.

- The converse is not yet known.

Other NL-Complete Problems I

- For two column vectors $x = (x_1,x_2,...,x_n)^T$ and $y=(y_1,y_2,...y_n)^T$, we define

  $$x \geq y \Leftrightarrow x_i \geq y_i \text{ for all index } i \in \{1,2,...,n\}.$$

- $LP_{2,k}$ (linear programming problem)
  - instance: a rational $m \times n$ matrix A, a rational column vector $b \in Q^n$, where each row of A has at most two non-zero entries and each column of A has at most k non-zero entries
  - question: is there any {0,1}-vector x s.t. $Ax \geq b$?
  - ➢ $m_{col}(x)$ = # of columns in A
  - ➢ $m_{row}(x)$ = # of rows in A

- (Claim) $LP_{2,k}$ is NL-complete for any $k \geq 3$.

See the next slide.

Other NL-Complete Problems II

$$\overset{\textbf{A}}{\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{n1} \\ a_{21} & a_{22} & \cdots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}} \overset{\textbf{x}}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}} \geq \overset{\textbf{b}}{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}}$$

- A: m×n matrix
- x: n-dimensional column vector
- b: n-dimensional column vector

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m \end{cases}$$

Other Characterizations of LSH

- We have seen $2SAT_3$, 3DSTCON, and $LP_{2,3}$ so far.
- Interestingly, those three NL-complete problems have a common feature.

- Theorem: [Yamakami (2017)]
  The following three statements are logically equivalent.
  - LSH for $2SAT_3$ (with m_{vbl} or m_{cls})
  - LSH for $LP_{2,3}$ (with m_{row} or m_{col})
  - LSH for 3DSTCON (with m_{ver} or m_{edg})

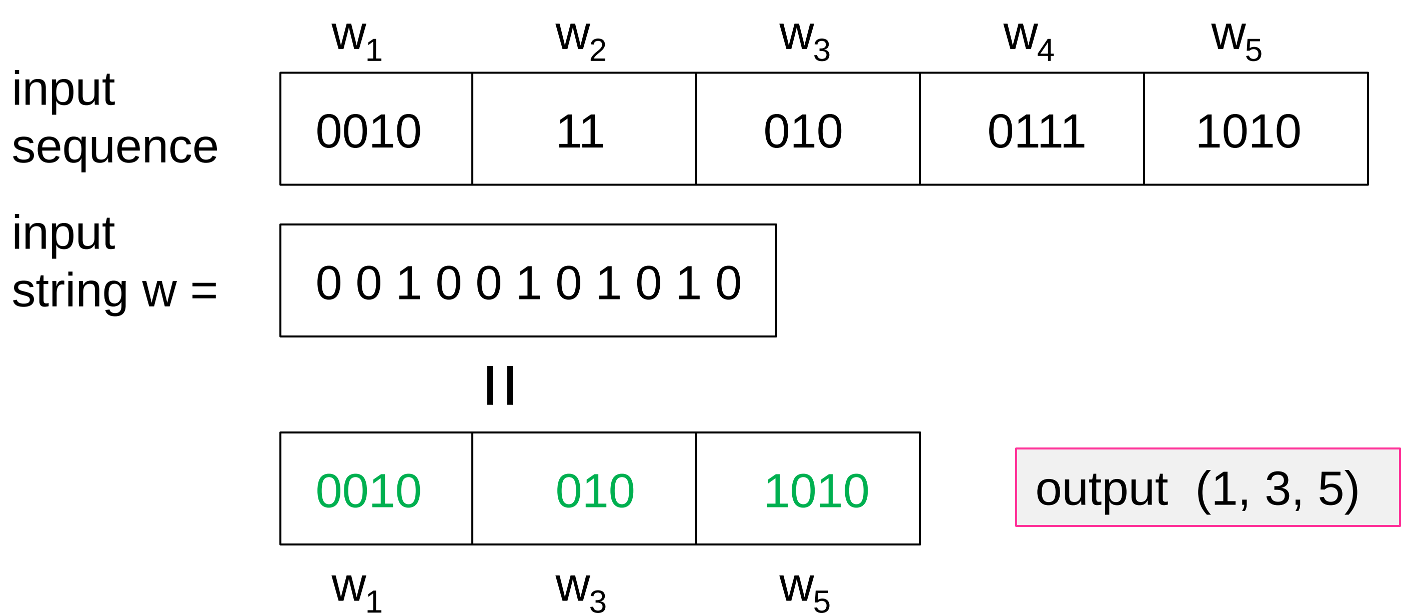
- However, not all NL-complete problems seem to share the above special property concerning LSH.

V. Applications of LSH

1. NL Search Problems
2. Complexity of Search-UOCK
3. NL Optimization Problems
4. Complexity of Max-HPP
5. Topological Sort
6. Complexity of TOPSORT

NL Search Problems

- The first application is in the field of NL search problems.
- Search-UOCK (a variant of Knapsack Problem)
  - instance: a string w, a sequence $(w_1,w_2,\ldots,w_n)$ of strings s.t., $\forall i \in [n]$, if w_i is a substring of w then w_i is unique
  - solution: a sequence $(i_1,i_2,\ldots,i_k)$ of indices with $k \geq 1$ s.t. $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ and $w = w_{i1}w_{i2}\ldots w_{ik}$.



Complexity of Search-UOCK

- Search-UOCK (again)
  - instance: a string w, a sequence $(w_1,w_2,\ldots,w_n)$ of strings over alphabet Σ s.t., $\forall i\in[n]$, if w_i is a substring of w then w_i is unique
  - solution: a sequence $(i_1,i_2,\ldots,i_k)$ of indices with $k \geq 1$ s.t. $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ and $w = w_{i1}w_{i2}\ldots w_{ik}$.
- size parameter: $m_{elm}(x) = n$ (the number of elements)
- Theorem: [Yamakami (2017)]

  If LSH (for $2SAT_3$) holds, then, for $\forall\varepsilon>0$, there is no polynomial-time $O(n^{1/2-\varepsilon})$-space algorithm for (Search-UOCK,m_{elm}).

NL Optimization Problems I

- The second application is in the field of NL optimization problems.
- In an optimization problem, intuitively speaking, we are asked to search for optimal solutions satisfying certain predetermined properties for each given input, where "optimality" is measured by cost functions m.

- NLO = class of NL optimization problems [Tantau (2007), Yamakami (2013)]

- (*) We will discuss optimization problems extensively in Week 9.

NL Optimization Problems II

- We further define an approximation class.

- LSAS$_{NLO}$ = class of NLO problems that have log-space approximation schemes [Tantau (2007), Yamakami (2013)]
- A log-space approximation scheme for problem P is a DTM M that takes (x,k) as input and outputs a solution y of P using at most f(k)log(|x|) space with performance ratio R(x,y) $\leq$ 1+1/k, where f $\in$ FL. Such y is called a (1+1/k)-approximate solution.
- Performance ratio R(x,y) = max{ |m(x,y)/m*(x)|, |m*(x)/m(x,y)| }, where m*(x) = max{ m(x,y) | y is a solution for input x }.

Complexity of Max-HPP

- Max-HPP (maximum hot potato) [Tantau (2007)]
  - instance: an $n \times n$ matrix A whose entries are in [n], a $d \in [n]$, a start index $i_1 \in [n]$ for $n \in N^+$
  - solution: an index sequence $S = (i_1, i_2, \ldots, i_d)$ with $i_j \in [n]$
  - measure: total weight

$$w(S) = \sum_{j=1}^{d-1} A_{i_j, i_{j+1}}$$

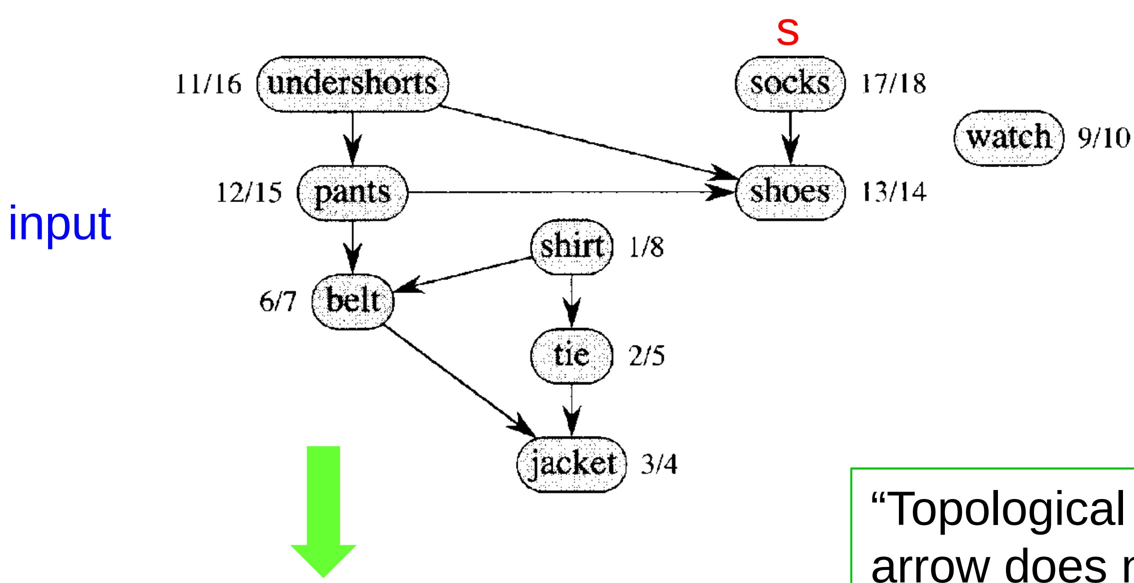
- size parameter: $m_{col}(x) = n$
- Max-HPP is in $LSAS_{NLO}$ [Tantau (2007)] but it is hard for LO_{NLO} under approximation-preserving exact NC^1-reduction [Yamakami (2013)].

- Theorem: [Yamakami (2017)]

  If LSH for $2SAT_3$ holds, then, for $\forall \varepsilon > 0$, there is no polynomial-time $O(k^{1/3} \log(m_{col}(x)))$-space algorithm finding $(1+1/k)$-approximate solutions of $(\text{Max-HPP}, m_{col})$.

Topological Sort

- Topological sorting problem (TOPSORT)
  - instance: an acyclic directed graph G and a source s in G
  - output: a topological sort of G starting from s



"Topological sort" means that every arrow does not go backward!

Complexity of TOPSORT

- LSH can tell how difficult to solve TOPSORT.
- More precisely, we obtain the following result.

- Theorem: [Yamakami (2017)]

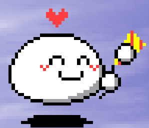
  If LSH (for $2SAT_3$) holds, then no DTM solves $(TOPSORT, m_{ver})$ in polynomial-time using $O(m_{ver}(x)^{\varepsilon/2})$ space on instances x for any fixed constant $\varepsilon \in [0,1)$

Open Problems

- There are numerous problems that have been left unsolved concerning LSH.

- Here are several important questions.
  1. Find more interesting and practical applications of LSH.
  2. Prove or disprove that LSH is true.
  3. Discuss the relationships between LSH for $2SAT_3$ and LSH for 2SAT.

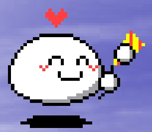
- (*) We will return to a discussion on LSH in Week 6.



Thank you for listening

Q & A

I’m happy to take your question!



END