LOGARITHMIC SPACE

Whenever an input encoding is reasonable (non-redundant), all input symbols must be read. Thus it is not customary to consider sublinear time bounds when dealing with the standard Turing machine model (the situation is different with so called random access TM models).

When considering space bounds we can make the following distinction:

- tape cells that are needed to store the input,
- tape cells that the computation may use to store " the internal data structures" needed by the algorithm.

For sublinear space bounds we use a k-tape Turing machine model, $k \geq 2$, where

- the first tape is a read-only tape containing the input,
- the other $k-1$ tapes are ordinary TM worktapes.

The space used by the computation counts only the squares visited on the $k-1$ worktapes.

Definition.

$$\mathrm{L} = \mathrm{SPACE}(\log n)$$

$$\mathrm{NL} = \mathrm{NSPACE}(\log n)$$

Note. Using logarithmic space we can encode the position of the tape head on the read-only input tape, and thus the input tape head position cannot be used to encode information that is "outside" the given space bound. The above problem occurs when dealing with $o(\log n)$ space bounds.

Constant ($O(1)$) space bound gives a finite automaton with the difference that the tape head can move in two directions. Two-way (non)deterministic finite automata (2NFA, 2DFA) recognize only the regular languages.

Example. The following languages can be recognized in deterministic logarithmic space:

$$\{0^n 1^n \mid n \geq 0\}$$

$$\{w \in \{0,1\}^* \mid w = w^R\}$$

(Will be discussed in class.)

Example. Define

$$\text{PATH} = \{< G, s, t > \quad | \quad G \text{ is a directed graph that has a directed path from node } s \text{ to node } t \}$$

We have previously observed that PATH can be decided by a deterministic polynomial time TM.

The following *nondeterministic* algorithm decides PATH:
On input $< G, s, t >$, where $G = (V, E)$, $\#V = n$:

set count $= n$; node $= s$;

`while` count > 0 `do` {

`if` (node, t) $\in E$ then accept

else {

guess $x \in V$;

count $=$ count $- 1$;

`if` (node, x) $\in E$, then node $= x$ `else` reject;

}

}

reject;

We observe that the space requirement of the algorithm is logarithmic (explain). Thus PATH $\in$ NL. It is not known whether or not PATH is in L.

Note that the reachability problem for undirected graphs (the variant of PATH where G is undirected) can be solved in deterministic logarithmic space. This is a non-trivial result (O. Reingold 2005).

By definition L $\subseteq$ NL but it remains an open question whether the inclusion is strict. Note that Savitch's theorem implies that

$$\text{NL} \subseteq \text{SPACE}((\log n)^2)$$

Note. Savitch's theorem (and the same proof) works for space bounds $f(n) \geq \log n$ if we use the Turing machine model with a separate read-only input tape. The assumption $f(n) \geq \log n$ is needed because otherwise configurations of the machine cannot be stored in space $O(f(n))$ since the encoding of a configuration has to include the position of the input tape head.

Theorem. NL $\subseteq$ P

Discussion on the proof (details to be presented in class):

Note that if M is a nondeterministic $\log n$ space Turing machine, the number of nodes in a computation tree corresponding to an input of length n is not bounded by any polynomial in n. Thus a polynomial time Turing machine cannot simulate M simply by trying all possible computations.

Instead, the deterministic machine builds a reachability graph for the configurations of M (on the given input) and checks in polynomial time whether the accepting configuration[1]

[1]The original machine can be modified so that it has a unique accepting configuration (to be explained in class).

is reachable from the start configuration.

Log-space reductions

Polynomial time mapping reducibility is too "coarse" a measure to compare the difficulty of problems within classes like P or NL. Why?

Definition. (Def. 8.21) A log space transducer is a *deterministic* Turing machine with

- a read-only input tape,
- a read-write work tape that may contain $O(\log n)$ symbols on an input of length n,
- a write-only output tape.

The machine is not allowed to go back to check what it has earlier written on the output tape (the output tape is one-way, write-only).

A log space transducer M computes a function $f_M : \Sigma^* \longrightarrow \Sigma^*$ where $f_M(w)$ is the string on the output tape at the end of the computation when M is started with input w. We require that all computations of a log space transducer must halt. A function is said to be "log space computable" if it is computed by some log space transducer.

Note. In the above definition the logarithmic space bound is imposed only on the work tape. The number of symbols written on the output tape may be much larger. What is an upper bound for the length of the output?

We say that a language A is log space reducible to a language B, $A \leq_L B$, if there is a log space computable mapping reduction f from A to B.

Now we can define completeness for classes NL and P.

Definition. Let X be the class NL or P. A language B is said to be X-complete if

1. $B \in X$, and

2. every language $C \in X$ is log space reducible to B.

Theorem. If $A \leq_L B$ and $B \in$ L then $A \in$ L.

Proof: in class

Note. Here the naive idea of constructing a composition of the TM computing the mapping reduction f and the TM recognizing B would not work because, due to the logarithmic space bound, the string $f(w)$ may be too long to write down during the computation!

Corollary. If some NL-complete (respectively, P-complete) language is in L, then L = NL (respectively, L = P).

Theorem. PATH is NL-complete.

The proof uses an idea similar to the one we used earlier for showing NL $\subseteq$ P.

Finally we mention (without proofs) some P-complete problems.

A *Boolean circuit* is a representation of a Boolean formula as a directed acyclic graph. In the graph we can identify identical subformulas.

- each node (gate) is labeled by $\vee$, $\wedge$, $\neg$, T, F or a variable x_i
- the indegree of the nodes is

  $\vee$, $\wedge$: 2

  $\neg$: 1

  T, F, x_i: 0

- the outdegree is unbounded

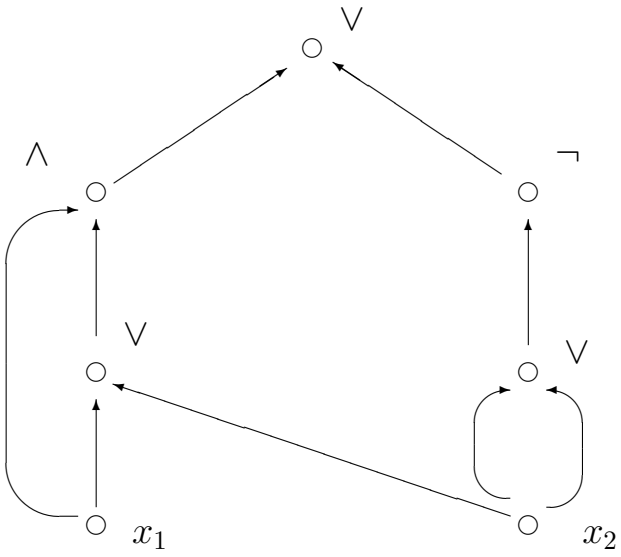

Figure 1: Example of a Boolean circuit

We define

$$\text{TCV} = \{ < \phi > \quad | \quad \phi \text{ is a Boolean circuit not containing variables that evaluates to true } \}$$

Note that TCV (True Circuit Value) is "obtained from" SAT by fixing the values of the variables.

Theorem. TCV is complete for P with respect to log space reductions.
Proof: omitted.

Another example of a P-complete problem is the emptiness problem for context-free grammars:

$$E_{CFG} = \{< G >| \; G \text{ is a CF grammar and } L(G) = \emptyset\}$$

Note that if we can show $E_{CFG} \in$ NL then P = NL.