**Complexity, 2004-2005**
*James Lawson*

1.(a) i. For any input $x \in L$, there is some computation that halts and succeeds and the computation tree has a polynomial depth.

ii. NP is the set of langauges that can be decided by a NDTM in p-time.
PSPACE is the set of langauges that can be decided by a $k$-tape Turing Machine with input-output in p-space.

iii. Take some $L$ in NP. Then there is a NDTM $M$ that decides $L$. Let $N$ have degree of nondeterminism, $D$. $N$ operates in p-time and so every computation can be seen as a sequence of $p(n)$ nondeterministic choices $(d_1, d_2, d_3, ..., d_n)$, $1 \leqslant d_i \leqslant D$, where $p(n)$ is some polynomial. An IOTM can enumerate $D^{p(n)}$ nondeterministic choices, only storing the current and check whether it leads to a successful halting state - if so, halt and succeed. If we exhaust all enumerations without success, halt with failure. Each check requires simulation of $p(n)$ steps of M takes time $O(f(n))$ and so space $O(f(n))$. To get started, the first sequence $p(n)$ 1s can be computed in space $O(f(n))$ since any polynomial $p(n)$ is proper.
This IOTM decides $L$ in p-space. Hence $L$ is in PSPACE.

(b) To show HCE is NP-complete, consider a reduction HC $\leq$ HCE.
Let

$$f((V,E)) = \begin{cases} (\triangle\, e, e) & \text{if } |V| = 1 \\ (\bullet\!-\!\bullet, e) & \text{if } |V| = 2 \\ ((V \cup \{\mathsf{p},\mathsf{q}\}, E \cup \{(x,\mathsf{p}),(\mathsf{p},\mathsf{q})\} & \\ \quad \cup \{(\mathsf{q},y) \mid (x,y) \in E\}), (\mathsf{p},\mathsf{q})) & \text{otherwise} \\ \quad \text{for some } x \in V & \end{cases}$$

We show $(V,E) \in$ HC iff $((V',E'),e) \in$ HCE

- ($\Rightarrow$) Assume $G$ in HC. Consider all possible forms of a HC in $G$.
  i. $z..xy..z$: We can construct a HC in $G'$ that takes the same cycle but now, also goes through p and q, $z..x\mathsf{p}\mathsf{q}y..z$. This cycle is well-formed because edge $(\mathsf{q},y)$ in $G'$ iff $(x,y)$ in $G$ and the cycle does indeed go through edge $e = (\mathsf{p},\mathsf{q})$, so $G'$ is in HCE.
  ii. $x$. Here, $G = \bullet$, is a yes-instance of HC, and $f(G)$ outputs a yes-instance of HCE.
- ($\Leftarrow$) Assume $(G',e)$ in HCE. Consider all possible forms of a HC in $G'$ using edge $e$.
  Note cycles visiting 1 or 4 nodes aren't valid cases. $f$ cannot output $G'$ with degree 1 or degree 4, so such cycles wouldn't be HCs in $G'$.

i. $(z..x\mathsf{p}\mathsf{q}y..z, (\mathsf{p},\mathsf{q}))$: We can construct a HC in $G$ that takes the same cycle but visits neither $\mathsf{p}$ nor $\mathsf{q}$. So, $G$ has HC: $z..xy..z$, which is well-formed since edge $(x,y)$ in $G$ iff $(\mathsf{q},y)$ in $G'$.

ii. $(uvwu, (u,v))$: $f$ only outputs graphs, $G'$, of degree 3 when $G = \bullet$, a yes-instance.

We can compute $f$ in p-time. Checking whether $|V| = 1$ or $|V| = 2$ and adding new nodes/edges can all be done by reading/writing $G$'s adjacency matrix in polynomial time. Hence $f$ is a p-time computable function for the reduction HC $\leq$ HCE.

HCE is in NP because given an instance of HCE, $(G,e)$ we can guess a HC, and check in p-time that it is a cycle, and, check that the cycle uses $e$. Given that HC is NP-complete, since HCE is in NP, and HC $\leq$ HCE, we have that HCE is NP-complete.

(c) *Algorithm*: Start from $x$ and have a counter $c$ with value 1.
Check adj matrix and see if there is only one possible next step. If there exactly one step, make that node the current node, increment $c$ and repeat. Repeat until we reach $y$ - halt and succeed. If $c > n$, then stop loop, halt and fail.
*Reasoning*: Checking only one possible next step can be done with a fixed number of counters bounded in value by $n$. Since we are only *reading* the input adj matrix, there is no cost in space. We only store the current node (rather than the whole path), and the $c$ counter will always use space $\log(n)$. Hence this algorithm describes a IOTM that decides D-RCH in logspace so D-RCH $\in$ LOGSPACE.

2.(a) i. NL: The class of languages that have a nondeterministic $k$-tape turing machine with input-output decide the language in logarithmic space.
co-NL: The class of complemented NL languages. ($L \in$ NL iff $\overline{L} \in$ co-NL)

ii. Given a directed graph $G$ and a node $x$, let $N(x)$ be the set of nodes reachable from $x$. Then $f((G,x)) = |N(x)|$ is computable in logspace.

iii. NL is closed under complementation. That is, NL = co-NL. The machine in the proof of the S-I theorem can be modified to calculate $N(x)$. This allows us to use the *reachability method* to exam a config graph in logspace and see what successfuly halting configurations are reachable. In particular, suppose $M$ decides a language in L, we can build machine $M'$ that constructs $G(M,x)$.
We use the S-I theorem corollary to calculate $N(\text{init config of } M(x))$ in logspace and say that $M'$ succeeds iff all nodes in $N(\text{init config of } M(x))$ are not successful halting config. $M'$ decides $\overline{L}$. We can apply the same argument, for any $\overline{L} \in$ co-NL to so show that $\overline{\overline{L}} \in$ NL, i.e. that $L \in$ NL. Hence NL = co-NL.

(b) i. A language, $L$ is NL-hard iff $L' \in \mathsf{NL}$ implies $L' \leqslant L$.

ii. Assume is $L$ is NL-hard.
Assume $L' \in \mathsf{NL}$.
Since, NL = co-NL, we have $\overline{L}' \in \mathsf{NL}$ and, since $L$ is NL-hard, we must have $\overline{L}' \leqslant L$. This is true iff $\overline{\overline{L}'} \leqslant \overline{L}$ iff $L' \leqslant \overline{L}$.
Hence $\overline{L}$ is NL-hard.

(c) i. 2SAT: Given a set of clauses with two literals, is there a satisfying assignment?

ii. $\overline{\text{RCH}}$: Given $G$, is there no path from $x$ to $y$.
Consider a logspace reduction $\overline{\text{RCH}} \leqslant_{\log} 2\text{SAT}$.
To show: $(G,s,t) \in L$ iff $\varphi \in L'$.
For each node $i$ define variable $x_i$. For each edge $(i,j)$ of the $G$, have clause $\neg x_i \vee x_j$ in $\varphi$ (which is logically equivalent to $x_j \to x_i$).
Add the clauses $(x_s \vee x_s)$ and $(\neg x_t \vee \neg x_t)$. $\varphi$ can be computed in logspace (only store info about current node, use fixed counters when accessing adj matrix, write to output tape before moving to next node).
From the previous part, since RCH is NL-hard, the complement, $\overline{\text{RCH}}$, is also NL-hard. Since we have $\overline{\text{RCH}} \leqslant_{\log} 2\text{SAT}$, have that $\overline{2\text{SAT}}$ is NL-hard.

3.(a) i. A family of circuits is *uniform* iff there is a logspace bounded IOTM which, given input of $1^n$, outputs circuit $C_n$ - where $C_n(n) = \mathsf{tt}$ if $n \in S$, $C_n(n) = \mathsf{ff}$ if $n \notin S$, for any undecidable set $S$ of natural numbers.

ii. $\mathsf{NC}_j$ is the class of languages that have some uniform family of circuits that decide the language in $O(\log^j(n))$ parallel time and $O(n^k)$ work for some $k$.

(b) We can take the boolean adjacency matrix for $A$ and square $\lceil \log(n) \rceil$ times to give the transitive closure, then read entry $(x,y)$ so see whether, from $x$, we can reach $y$. By using the parallel matrix multiplication alg (whic has time $\log(n)$), our squaring algorithm have a total time of $\log^2(n)$ and $O(n^3) \times \log(n) = O(n^k)$ work. Hence $L \in \text{NC}_2$.

(c) Connect every input to a not gate. connect consecutive not gates to an and gate. Then connecting the outputs of the and gates to form a knockout-tournament of or-gates. This has depth $\log(n)+k$ and size $n+(n-1)+(n-2) = O(n^k)$ for some $k$. Hence there is some uniform family of circuits that decides in $O(\log^j(n))$ parallel time and $O(n^k)$ work for some $k$. So $L \in \text{NC}_j$.

(d) Assume $L_1, L_2 \in \mathsf{NC}_j$.
Since $L_1$ is in $\mathsf{NC}_j$ and so there is a logspace bounded IOTM which, given input of $1^n$, outputs circuit $C_n$ - where $C_n$ decides $L_1$. Similarly, there is a machine that on input $1^n$ outputs $C'_n$ that decides $L_2$.

In parallel compute: $C'_n(x)$, $(C_1(x_1), C'_{n-1}(x_2..x_n))$, $(C_2(x_1x_2), C'_{n-2}(x_3..x_n))$, ... $(C_n(x_1...x_n))$. For the $n-1$ of these that are pairs, $(C(...), C'(...))$ connect each pair of outputs to an and-gate. Now form a knockout tournament with players being the outputs of these $n-1$ and gates as well as the outputs from $C'_n(x)$ and $C_n(x)$. This circuit has depth bounded by $O(\log^j(n))$ (doing the $C$'s in parallel) + 1 (and-gates) + $\lceil(\log(n+1))\rceil$ (tournament) which is $O(\log^j(n))$ depth overall. The circuit has size $(n+1)O(n^k)$ (doing the $C$'s in parallel) + $n-1$ (and-gates) + $n$ (or-gates), which, overall is $O(n^{k'})$ for some $k'$.

Hence there is some uniform family of circuits that decides $L_1L_2$ in $O(\log^j(n))$ parallel time and $O(n^{k'})$. So $L_1L_2 \in \mathrm{NC}_j$.

4.(a) i. An approximation algorithm for a minimisation problem is *k-optimal* iff if it guarantees a solution which is no worse than $k$ times the optimal.

ii. x

iii. x