# Oblig 4 - INF2080

runehovd

11. mai 2018

## Problem 1 (Sipser 8.14)

An undirected graph is bipartite if its nodes may be divided into two sets so that all edges go from a node in one set to a node in the other set. A graph is bipartite if and only if it does not contain a cycle that has an odd number of nodes. Using this fact, show that the language $BIP = \{G \mid G \text{ is a bipartite graph}\}$ is in NL.

Since NL = coNL, we can show that $BIP$ is in NL by proving that $\overline{BIP}$ is in NL. $\overline{BIP} = \{G \mid G \text{ is a graph that contains a cycle that has an odd number og nodes}\}$, or:
$\overline{BIP} = \{G \mid G \text{ is not a bipartite graph}\}$
An NL machine M for $\overline{BIP}$ :
M on input <G>:

1. Choose a node nondeterministically and store it as $v$
   - Let max be number of nodes i G
   - Set counter to 1
   - Choose a neighbornode to $v$ nondeterministcally as $u$
2. if $v = u$ and counter is odd, accept
3. nondeterministically choose a neighbornode to $u$ as $u$ and increase counter by 1
4. if $counter < max$, go to 2. Else, reject.

We can see that the space required to store this is in log-space. This proves that $\overline{BIP}$ is in NL, and $BIP$ is in coNL, but since NL = coNL, $BIP$ is also in NL.

## Problem 2 (Sipser 9.20)

Describe the error in the following fallacious "proof" that P $\neq$ NP. Assume for contradiction that P = NP. Then SAT $\in$ P and so for some k, SAT $\in$ TIME($n^k$). Because every language in NP is polynomial time reducible to SAT, we have that NP $\subseteq$ TIME($n^k$). Therefore, P $\subseteq$ TIME($n^k$). But by the time hierarchy theorem, TIME($n^{k+1}$) contains a language that isn't in TIME($n^k$), which contradicts P $\subseteq$ TIME($n^k$). Therefore, P $\neq$ NP.

The error in the proof is the line *Because every language in NP is polynomial time reducible to SAT, we have that NP $\subseteq$ TIME($n^k$).*
The problem is that even though some problem is reducible to SAT in polynomial time, it does not mean that it is in TIME($n^k$).
If you have a problem that takes polynomial time to reduce to SAT, and also runs the SAT algorithm n times, then it will be in TIME($n^{k+1}$) because you have $n*$TIME($n^k$).

## Problem 3

Show that P $\neq$ SPACE($n$). Hint: Assume that they are equal, and look for a contradiction involving the space hierarchy theorem.

Assume that P = SPACE($n$).
Then there exists an algorithm to simulate a Turing machine that uses SPACE($n$) and TIME($n^c$) for some constant. That also means there exists an algoritm that uses SPACE($n^2$) and uses TIME($n^{2c}$).
If this is the case, it contradicts the Space Hierarchy Theorem, because that would mean:
$SPACE(n) = SPACE(n^2)$.