Turing Machines

What can go wrong with stack memory?
Tape memory and “random access”.
High-level descriptions of Turing Machines.
Low level detailed descriptions of Turing Machines.
1. Another way to solve a problem: list the language out.

   Recursive definition of a set of strings.

3. Parse Trees. Parse trees need not be unique.

4. CFGs are more powerful than finite automata for example $L = \{0^n1^n|n \geq 0\} \in \text{CFG}$.

5. Are $\{ww^R|w \in \Sigma^*\}$ and/or $\{ww|w \in \Sigma^*\}$ CFGs?

6. CFG ‘solves’ $L = \{0^n1^n|n \geq 0\}$ by having access to a ‘stack memory’ - only have access to top of memory stack.

7. Powerful? $\{ww|w \in \Sigma^*\}$ and $\{0^n1^n(01)^n|n \geq 0\}$ are not CFGs.
   Need something more powerful than stack memory: RAM and the Turing Machine.
Today: Turing Machines

1. Stack memory helps to solve $L = \{0^n#1^n\}$ but isn’t enough for $L = \{0^n#1^n#0^n\}$.

2. Tape memory: read, write and move around capability.

3. Using tape memory to solve $L = \{0^n#1^n#0^n\}$.
   
   High-level description of a Turing Machine (TM) — *pseudo-code*.

4. Low-level detailed description of a TM to solve $L = \{0^n#1^n#0^n\}$ — *machine-code*.

5. High-level description of a TM for multiplication, $L = \{0^i#1^j#0^{i\times j} | i, j \geq 1\}$.
   
   What about the detailed low-level description?

6. A TM is more powerful than an actual computer which has finite memory.