2017-12-21

- Definition of P, NP, coNP:
  - **P** := $\cup_{k\geq 1}$ TIME[n^k].
  - **NP** := $\cup_{k\geq 1}$ NTIME[n^k].
  - **coNP** := $\{L|\Sigma^* - L \in \text{NP}\}$.
- Definition of NP-hard, NP-complete:
  - **NP-hard**: $\{L|\forall L' \in \text{NP}, L' \leq_p L\}$.
  - **NP-complete**: $\{L|L \in \text{NP} \text{ and } L \in \text{NP-hard}\}$.
- **Theorem**: Every $f(n)$ time k-tape TM M has an equivalent $O(f^2(n))$ time single-tape TM S.
  - Each of the k active portions on S has length at most $f(n)$ because M uses $f(n)$ tape cells in $f(n)$ steps.
  - To simulate each of M's steps, S performs two scans and possibly up to k rightward shifts. Each uses $O(f(n))$ time, so the total time for S to simulate one of M's steps is $O(f(n))$.
  - Afterward, S simulates each of the $f(n)$ steps of M, using $O(f(n))$ steps.
  - Thus, the entire simulation uses $O(f^2(n))$ steps.
- **Theorem**: Every $f(n)$ time single-tape NTM N has an equivalent $2^{O(f(n))}$ time single-tape DTM D.
  - Every branch of N's nondeterministic computation tree has a length of at most $f(n)$.
  - Every node in the tree can have at most b children, where b is the maximum number of legal choices given by N's transition function.
  - Thus, the total number of leaves in the tree is at most $b^{f(n)}$.
  - Thus, the running time of D is $O(f(n)b^{f(n)}) = 2^{O(f(n))}$.
- **Cook-Levin Theorem**: SAT is NP-complete.
  - Suppose that a given NP problem can be solved by the NTM M. For each input word w, we specify a Boolean expression B which is satisfiable if and only if M accepts w.
  - Consider the space-time diagram, we define the following Boolean variables:
    - $T_{i,j,k}$: True if tape cell i contains symbol j at step k of the computation.
    - $H_{i,k}$: True if the M's read/write head is at tape cell i at step k of the computation.
    - $Q_{q,k}$: True if M is in state q at step k of the computation.
  - Define the Boolean expression B that describes the accepting run of M on w. Then, B is satisfiable if and only if M accepts w.
- **Theorem**: 3-SAT is NP-complete. (Intuition: SAT $\leq_p$ 3-SAT)
  - Construct a binary parser tree for input formula Φ and introduce a variable y_i for the output of each internal node.
  - Rewrite Φ as the conjunction of the root variable and clauses describing the operation of each node.
  - Convert each clause Φ_i' to CNF by constructing a truth table and applying DeMorgan's Law.
- **Theorem**: 3-colorable is NP-complete. (Intuition: 3-SAT $\leq_p$ 3-colorable)
  - Create triangle with node True, False, Base.

- For each variable x_i, create two nodes v_i and v_i' connected in a triangle with common Base.
- For each clause $C_j = (a \vee b \vee c)$, add OR-gadget graph with input nodes a, b, c and connect output node of gadget to both False and Base.

- **Theorem**: Clique is NP-complete. (Intuition: 3-SAT $\leq_p$ Clique)
- **Theorem**: Independent Set is NP-complete. (Intuition: Clique $\leq_p$ Independent Set)
- **Theorem**: Vertex Cover is NP-complete. (Intuition: Clique $\leq_p$ Vertex Cover)
- **Theorem**: Dominating Set is NP-complete. (Intuition: Vertex Cover $\leq_p$ Dominating Set)
  - Given a graph G, we replace each edge of G by a triangle to create G'.
  - Subdivide each edge (u, v) by the addition of a vertex, and add an edge directly from u to v.
  - G has a vertex cover of size k iff the same set of vertices forms a dominating set in G'.