Section 23.5: The Exponential Time Hypothesis

Do NP-Hard Problems Require Exponential Time?

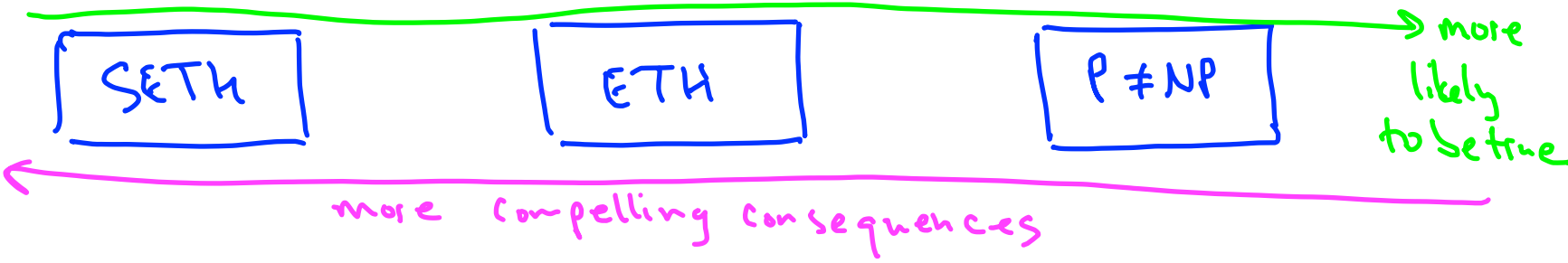
Possibility: $P \neq NP$, yet every NP problem solvable in subexponential time. (e.g., $n^{O(\log n)}$ or $2^{O(\sqrt{n})}$)

Exponential Time Hypothesis (ETH): There is a constant $c > 1$ such that: every algorithm solving 3-SAT has worst-case running time $\geq c^n$ [n = # of variables]

Comment: ETH $\Rightarrow$ most other natural NP-hard problems also require exponential time in the worst case. (via reductions)

Note: Can still beat exhaustive search for 3-SAT. [state of the art: $\approx (1.308)^n$; exhaustive search: 2^n]

The Strong ETH (SETH)



Strong Exponential Time Hypothesis (SETH): for every $c < 2$, exists positive integer k such that: every algorithm solving k-SAT requires time $\geq c^n$ in the worst case. [n = # of variables]

Consequence: for SAT, can't significantly improve over exhaustive search.

Keyword: "fine-grained complexity"

Running Time Lower Bounds for Easy Problems

Parts 1-3: "holy grail" = algorithm with linear or near-linear time.

Non-example: Sequence alignment [dynamic programming, $O(n^2)$ time]

Theorem: (Backurs + Indyk, mid-2010s) a $O(n^{2-\varepsilon})$-time algorithm for sequence alignment would refute the SETH. [e.g., $\varepsilon = .01$]

Trick: given k-SAT with n variables, construct two input strings of length $\approx 2^{n/2}$.

⇒ can get better-than-exhaustive-search algorithm for k-SAT by composing this reduction with a subquadratic-time sequence alignment algorithm.