CS310

P vs NP

the steel cage death match

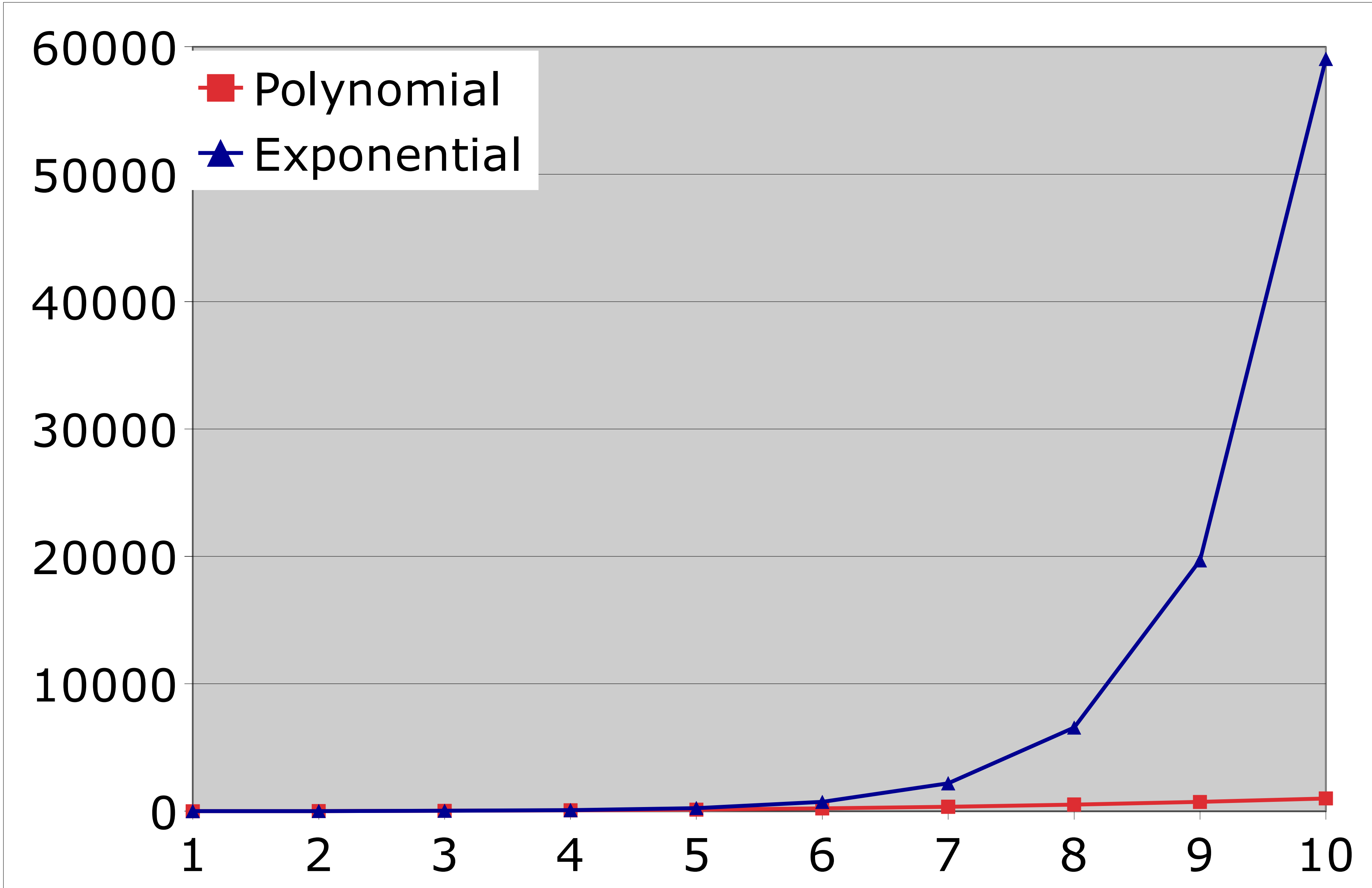
How hard is a problem to solve?

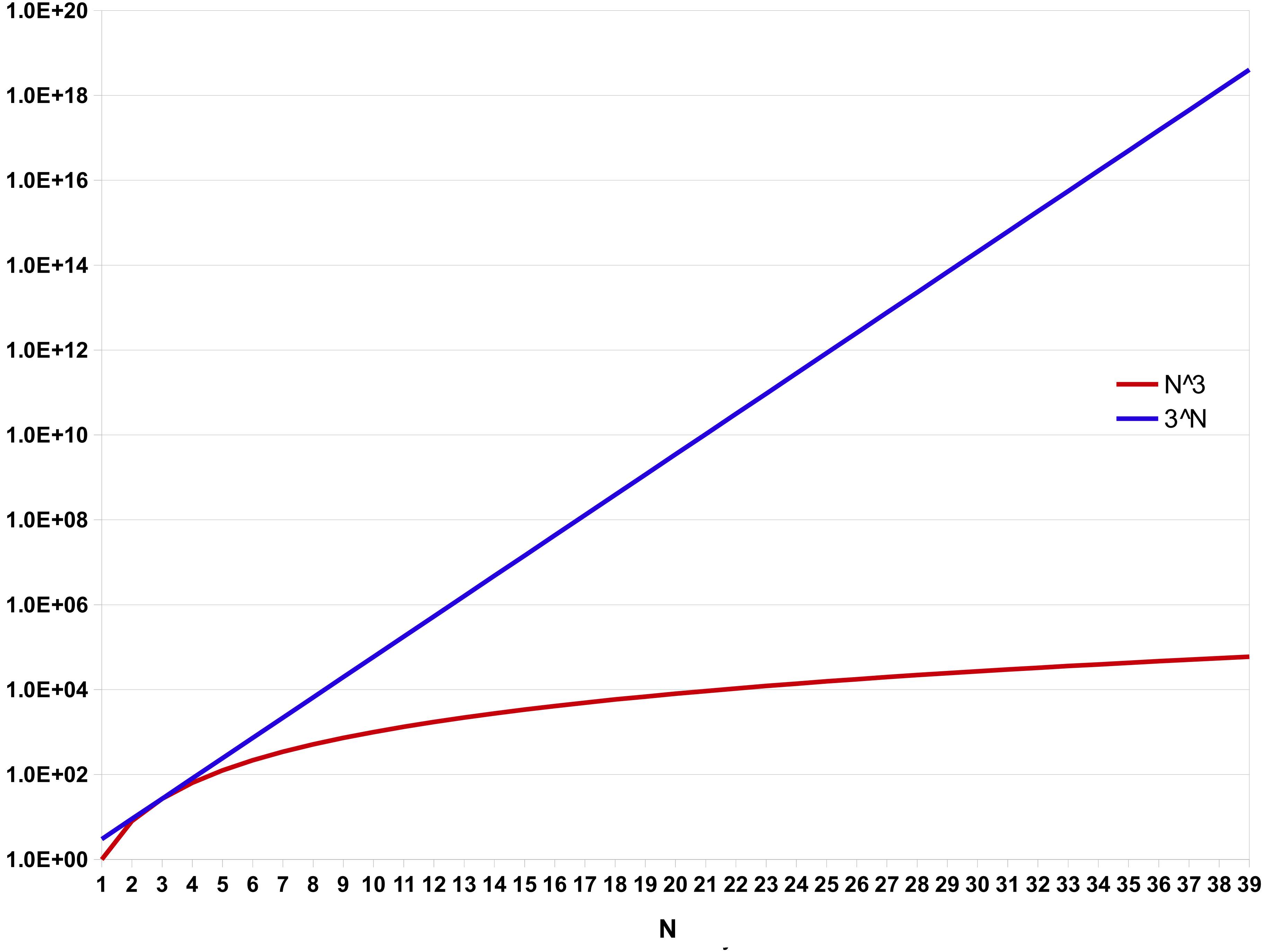
Section 7.2

November 28, 2014

Polynomial vs Exponential

- Polynomial: n^3 Exponential: 3^n




1.0E+20
1.0E+18
1.0E+16
1.0E+14
1.0E+12
1.0E+10
1.0E+08
1.0E+06
1.0E+04
1.0E+02
1.0E+00
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
N
N^3
3^N

Complexity relationships between models

- Theorem 7.8: let t(n) >= n, every t(n) time multi-tape TM has an equivalent $O((t(n)^2)$ time single-tape TM.
  - polynomial difference
- Theorem 7.9: Every t(n) >= n time ND single tape TM has an equivalent $2^{O(t(n))}$ time deterministic single tape TM
  - exponential difference

The class P

- P is the class of languages

- Problems in class P

RELPRIME Sipser, p 261

PROOF The Euclidean algorithm E is as follows.

$E =$ "On input $\langle x, y\rangle$, where x and y are natural numbers in binary:

1. Repeat until $y = 0$:
2. Assign $x \leftarrow x \bmod y$.
3. Exchange x and y.
4. Output x."

Algorithm R solves *RELPRIME*, using E as a subroutine.

$R =$ "On input $\langle x, y\rangle$, where x and y are natural numbers in binary:

1. Run E on $\langle x, y\rangle$.
2. If the result is 1, *accept*. Otherwise, *reject*."

CFG Parsing   Sipser p 263

$D =$ "On input $w = w_1 \cdots w_n$:

1. If $w = \varepsilon$ and $S \to \varepsilon$ is a rule, *accept*. 〚 handle $w = \varepsilon$ case 〛
2. For $i = 1$ to n: 〚 examine each substring of length 1 〛
3. For each variable A:
4. Test whether $A \to \mathtt{b}$ is a rule, where $\mathtt{b} = w_i$.
5. If so, place A in $table(i, i)$.
6. For $l = 2$ to n: 〚 l is the length of the substring 〛
7. For $i = 1$ to $n - l + 1$: 〚 i is the start position of the substring 〛
8. Let $j = i + l - 1$, 〚 j is the end position of the substring 〛
9. For $k = i$ to $j - 1$: 〚 k is the split position 〛
10. For each rule $A \to BC$:
11. If $table(i, k)$ contains B and $table(k + 1, j)$ contains C, put A in $table(i, j)$.
12. If S is in $table(1, n)$, *accept*. Otherwise, *reject*."

Real Life

- Problems in class P are usually manageable on a real computer
  - n^K
  - though k=100 may introduce some practical problems

The class NP

- NP is the class of languages

  – Problems in class NP

Verifier

- A verifier of a language, A, is an algorithm, V, such that

  A = { w | V accepts <w, c> for some string c}

  where c is a certificate

  |c| is polynomial in terms of |w|

Clique, Sipser p 268

PROOF The following is a verifier V for *CLIQUE*.

$V =$ "On input $\langle\langle G, k\rangle, c\rangle$:

1. Test whether c is a set of k nodes in G
2. Test whether G contains all edges connecting nodes in c.
3. If both pass, *accept*; otherwise, *reject*."

ALTERNATIVE PROOF If you prefer to think of NP in terms of nondeterministic polynomial time Turing machines, you may prove this theorem by giving one that decides *CLIQUE*. Observe the similarity between the two proofs.

$N =$ "On input $\langle G, k\rangle$, where G is a graph:

1. Nondeterministically select a subset c of k nodes of G.
2. Test whether G contains all edges connecting nodes in c.
3. If yes, *accept*; otherwise, *reject*."

Subset-Sum Sipser p 269

PROOF The following is a verifier V for *SUBSET-SUM*.

$V =$ "On input $\langle\langle S, t\rangle, c\rangle$:

1. Test whether c is a collection of numbers that sum to t.
2. Test whether S contains all the numbers in c.
3. If both pass, *accept*; otherwise, *reject*."

ALTERNATIVE PROOF We can also prove this theorem by giving a nondeterministic polynomial time Turing machine for *SUBSET-SUM* as follows.

$N =$ "On input $\langle S, t\rangle$:

1. Nondeterministically select a subset c of the numbers in S.
2. Test whether c is a collection of numbers that sum to t.
3. If the test passes, *accept*; otherwise, *reject*."

P vs NP

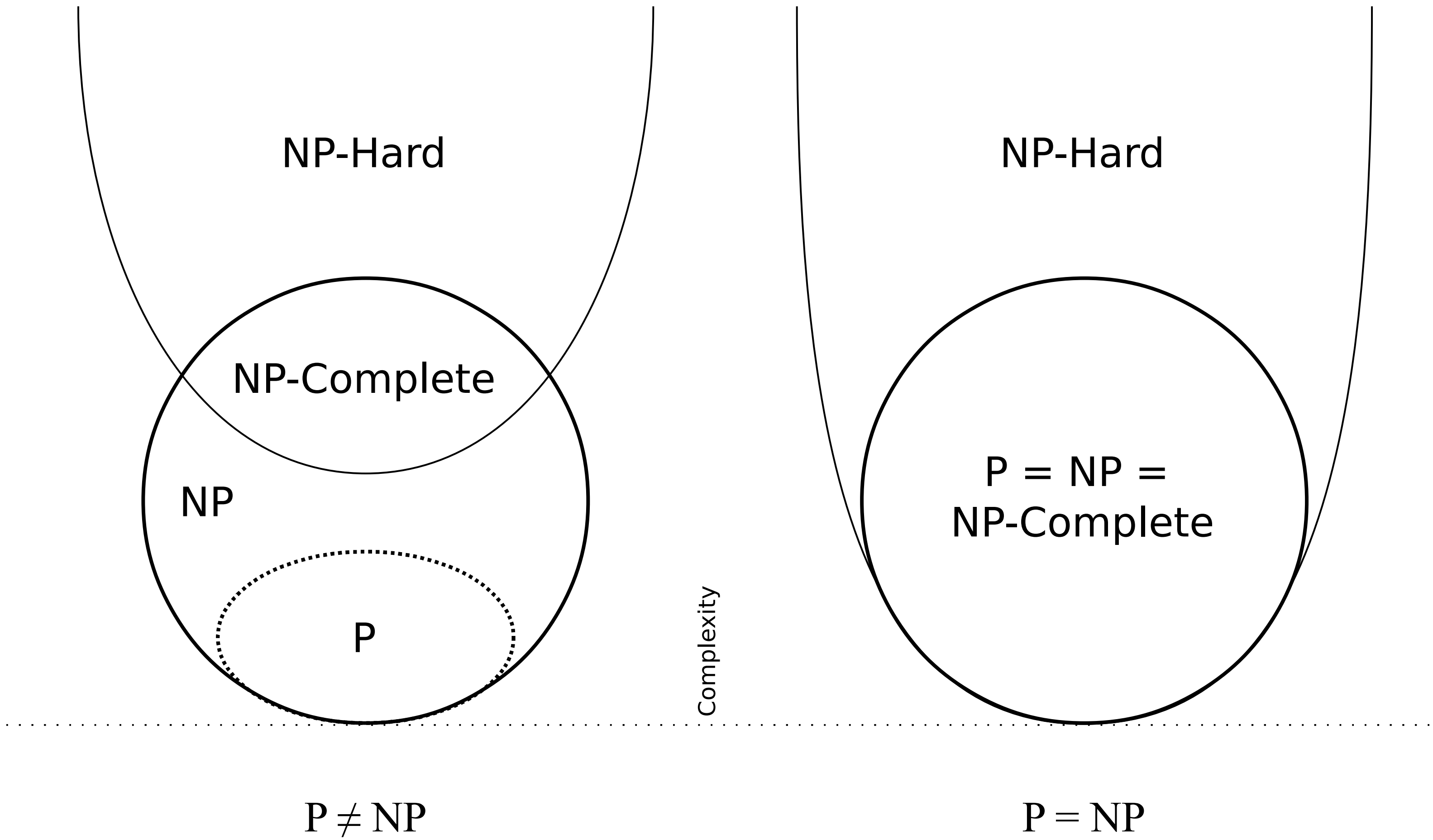
- $P \subseteq NP$
  - unknown if the classes are unequal
- If P = NP, then all problems in NP can be solved in polynomial time, if we are clever enough to find the right algorithm

NP-Complete

- NP-Completeness
  - set of problems in NP whose complexity is related to all problems in NP
  - if an NP-Complete problem can be shown to be in P, then P=NP
  - boolean satisfiability, for example
  - vertex-cover
  - clique
  - Hamilton Path

NP-Hard p 298, 7.33

-



http://en.wikipedia.org/wiki/File:P_np_np-complete_np-hard.svg

Recent Work

- Claim by Vinay Deolalikar (from HP Labs) that N != NP

- https://rjlipton.wordpress.com/2010/08/08/a-proof-that-p-is-not-equal-to-np/
  - Link to Deolalikar's paper
  - Much discussion

- http://en.wikipedia.org/wiki/P_versus_NP_problem#Claimed_solutions

- https://rjlipton.wordpress.com/2010/08/12/fatal-flaws-in-deolalikars-proof/
  - Fatal flaws?