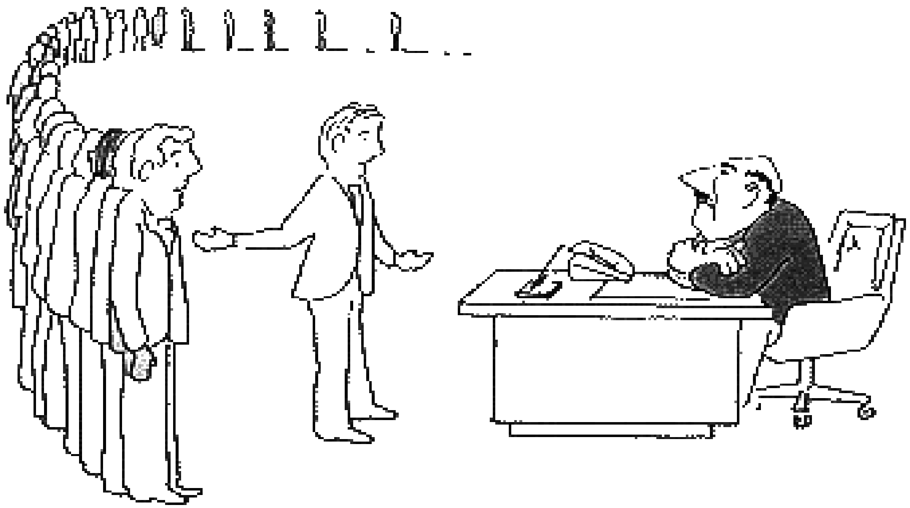
P = NP?

Projects, HW, and Tests!

- HW 5 Due Today – canvas
- Project 5 Assigned Thursday – Due December 8
  - Last Project
- Finals: 1 hour 15 minutes Long
  - 4:30pm Class: December 11, 3-4:15pm
  - 6:00pm Class: December 10, 5:45-7pm
  - Big-Oh, Sorting, Graphs, …
  - Practice Exams on Thursday
  -

Where are we?

- Dynamic Programming – Last week, today's hw that's due
- P = NP?
    - I couldn't find an efficient solution, but neither could all these famous people
    -
- Multi-threading – Rest of the semester!



What can we compute?

- We've focused on giving good algorithms for specific problems
  - General techniques that help do this
- Now shifting to focus to problems where *we think* this is impossible.
  - There are many...

Polynomial Time

- Many of the algorithms we studied are polynomial
-
-

Algorithm Review Runtimes – Big-Oh Expected Case

- Insertion Sort?
- Build Heap?
- Quick Sort?
- Merge Sort?
- Radix Sort?
- Prim's MST?
- Kruskal's MST?
- Dijkstra's?
- Topological Sort?

10 minute challenge to answe

Algorithm Review Runtimes – Big-Oh Expected Case

- Insertion Sort? $O(n^2)$
- Build Heap?$O(n^2)$
- Quick Sort? O(nlogn)
- Merge Sort? O(nlogn)
- Radix Sort? O(kn)
- Prim's MST? O(|E| log |V|)
- Kruskal's MST? O (|E| log |V|)
- Dijkstra's? O(|V| log |V| + |E| log |V|) or $|V|^2$
- Topological Sort? O(|E| + |V|)

The class P : Polynomial

- Definition: P = the set of (decision) problems solvable by computers in polynomial time
  - $T(n) = O(n^k)$ for some fixed k
  -
- Examples: sorting, shortest path, MST... most of the stuff we studied (exceptions: Towers of Hanoi)

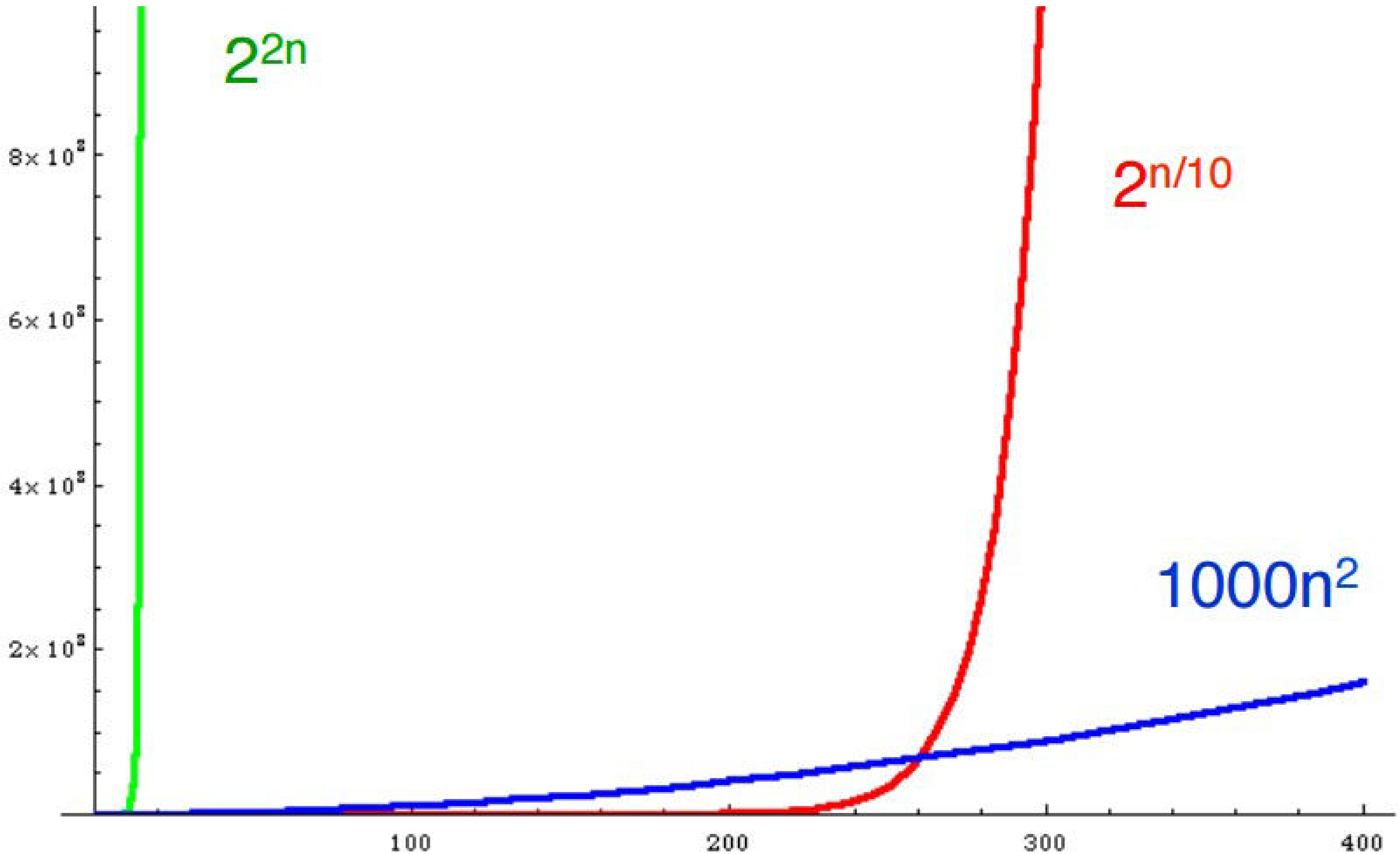
Easy vs Hard

- We usually call a problem “Easy” if it has polynomial time
- “Hard” problems have exponential running times
- Why do we care so much?
  - Polynomials have useful properties
-
-

Why “Polynomial”?

- Point is not that n^{2000} is a nice bound or that differences between n and 2n and n^2 are negligible
- Simple theoretical tools might not capture differences, but exponentials are qualitatively different from polynomials
  - “My problem is not in P” might mean you need to shift to a more feasible variant

Running Times



Another view of Poly vs Exp

- Next year's computer will be 2x faster. If I can solve a problem of size n today, how large a problem can I solve in the same time next year?

Complexity	Increase	E.g. $T=10^{12}$	
$O(n)$	$n_0 \rightarrow 2n_0$	10^{12}	2×10^{12}
$O(n^2)$	$n_0 \rightarrow \sqrt{2}\, n_0$	10^6	1.4×10^6
$O(n^3)$	$n_0 \rightarrow \sqrt[3]{2}\, n_0$	10^4	1.25×10^4
$2^{n/10}$	$n_0 \rightarrow n_0+10$	400	410
2^n	$n_0 \rightarrow n_0+1$	40	41

Nice Properties of Polynomial Time

- Any algorithm that runs in polynomial time, on any other computer will run in polynomial time
- Polynomials are closed under addition and composition
  - You can compose two polynomial algorithms (have one call another), still will run in polynomial time
  - You can call one polynomial algorithm then another, then another – all polynomial time
-
-

Running Times

- For some problems, we don't know if an efficient non-exponential growth solution to problems exists
- Lots of computer science research devoted to this

Poly vs Exponential

- Find the shortest path from node A to node B in a weighted graph
  - ??
- Hard: Find the longest path (without cycles) from node A to node B in a weighted graph
  - ??

Poly vs Exponential

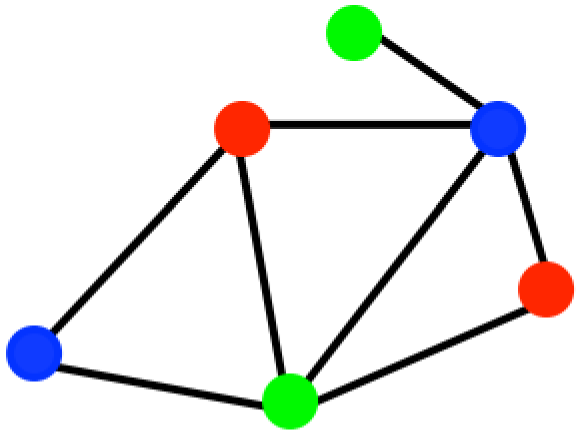
- Find the shortest path from node A to node B in a weighted graph
  - Dijkstra, polynomial
- Hard: Find the longest path (without cycles) from node A to node B in a weighted graph
  - All known algorithms are exponential

Decision Problem

- Ask these problems as a Yes-No problem
  - Is there a path from node A to node B with weight < M?
  - Is there a path from node A to node B with weight > M?
- Computation complexity usually analyzed using decision problems
  - Answer is just yes or no
  - Simpler to deal with
-

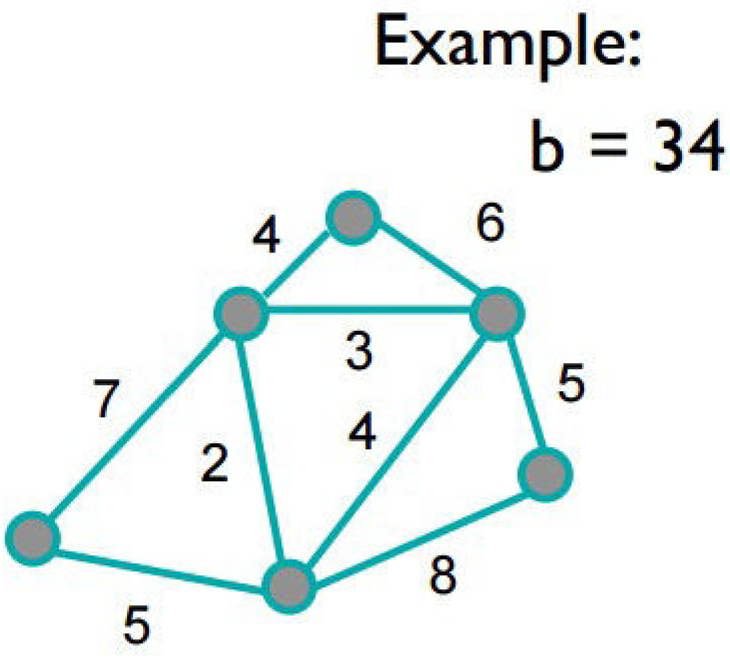
3-Coloring vs 2-Coloring

- 3-Coloring:
  - Graphs G=(V,E) for which there is an assignment of at most 3 colors to the vertices in G such that no two adjacent vertices have the same color
- How to pose as decision problem?



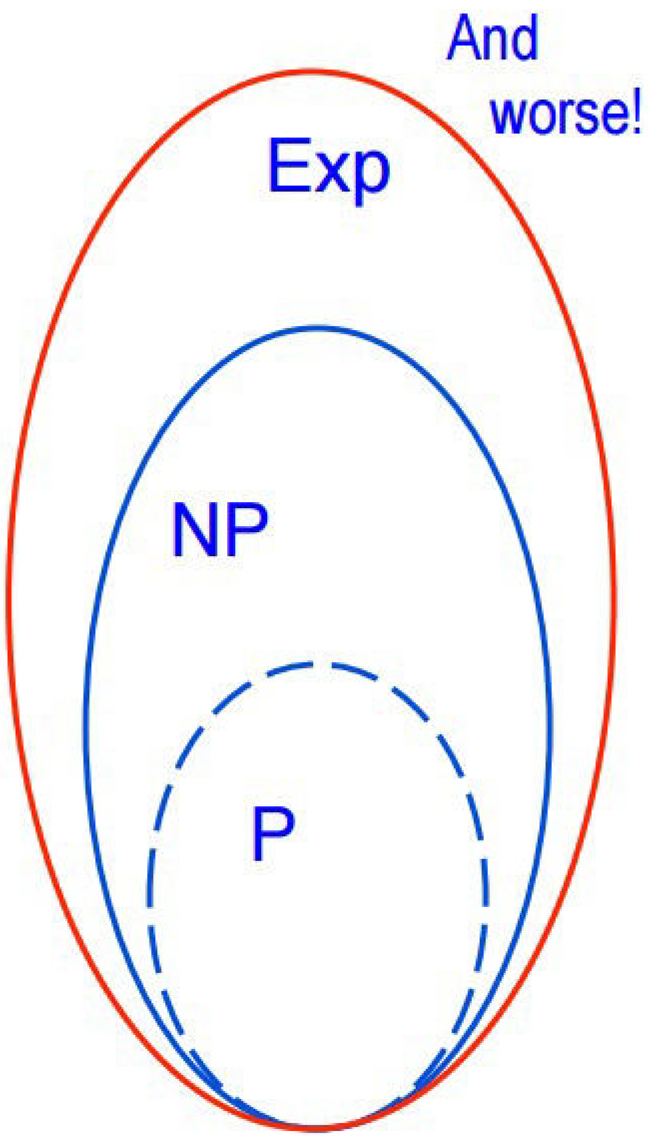
The Traveling Salesman Problem

- Find the shortest route for a salesman to visit every town exactly once



Not Every Problem is in P

- NP, commonly-seen class of problems, that appear to require exponential time (but unproven)



Common property of these problems

- “Answer” to a decision problem is just yes or no, but there's always are more elaborate “solution”
- Verifiable in polynomial time
- But the “solution” is buried in an exponentially large search space of potential solutions
-

NP : nondeterministic polynomial

- NP consists of all decision problems where
  - You can verify the YES answers in polynomial time, given a polynomial size solution
- And
  - No solution can trick your polynomial time verifier into saying YES for NO

Determinism vs. Nondeterminism

- What do we mean by non-deterministically polynomial?
-
-

.

Determinism vs. Nondeterminism

- What do we mean by non-deterministically polynomial?
- On a deterministic machine the next step to execute is unique based on current instruction
- On a *Theoretical* nondeterministic machine:
  - There is a choice of steps, the machine is free to choose any, and it will always choose the correct one leading to solution
  - Optimal guesser
-

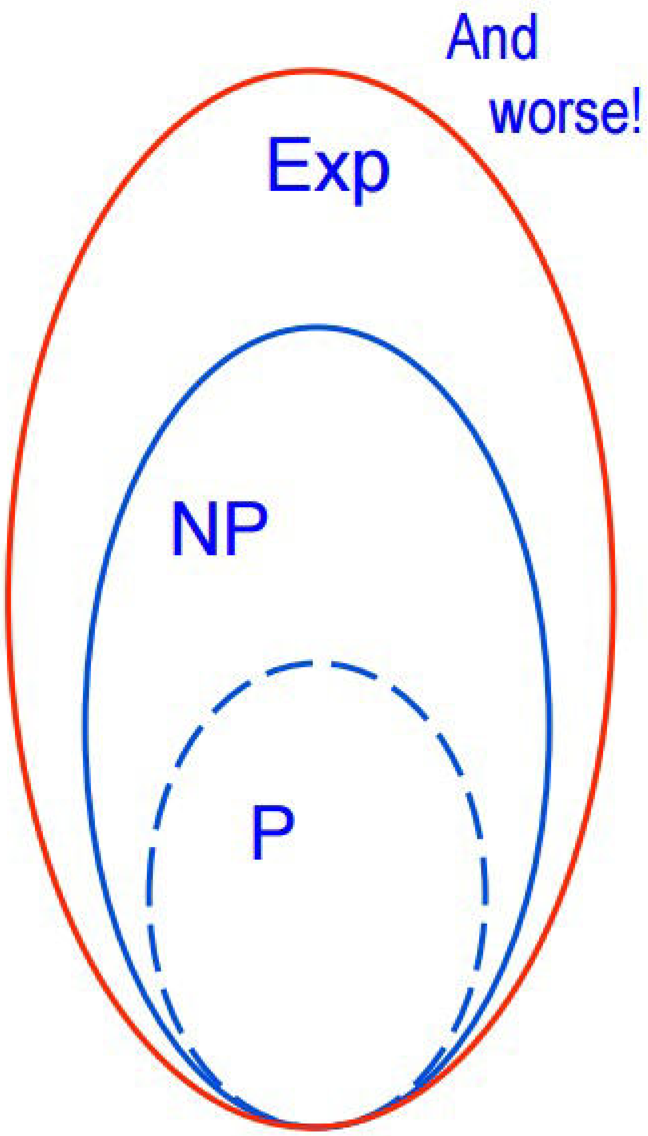
.

NP Definition Take 2

- Any problem that runs on a nondeterministic machine in polynomial time is in class NP
- Nondeterministic polynomial-time

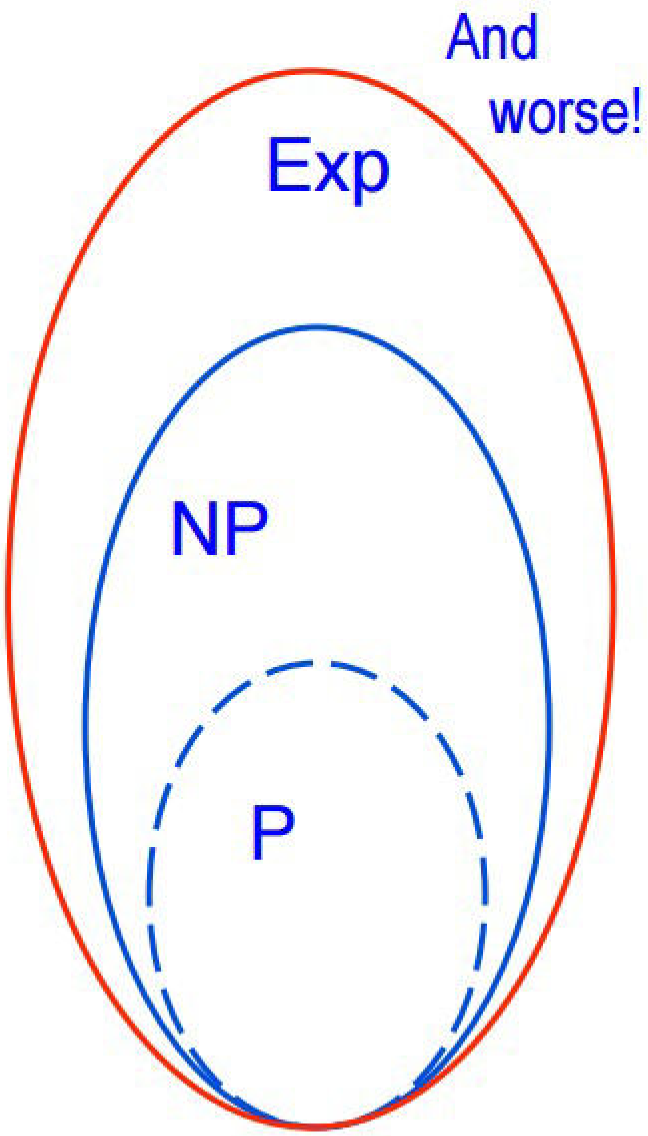
P and NP

- Every problem in P is in NP
-



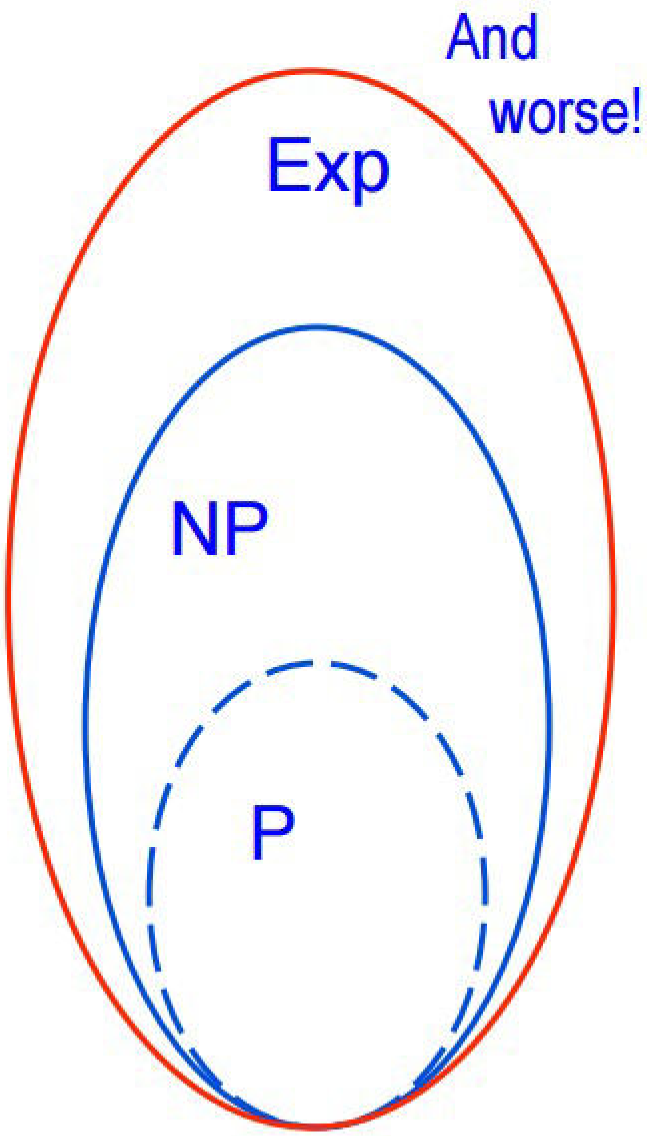
P and NP

- Every problem in P is in NP
- Every problem in NP is in exponential time
-



P and NP

- Every problem in P is in NP
- Every problem in NP is in exponential time
- $P \subseteq NP \subseteq Exp$
- We know $P \neq Exp$, so either
- $P \neq NP$, or $NP \neq Exp$



Does P = NP?

- This is the big open question!
- To show that P = NP, we have to show that every problem that belongs to NP can be solved by a polynomial time algorithm
- ...There are infinitely many problems in NP, do we need to pick them off one at a time???

Some Problem Pairs

- 2-Coloring vs 3-Coloring
- Shortest Path vs Longest Path
- Which ones are similar computationally?

P vs NP

- Theory
  - P = NP?
  - Open problem.
- Practice
  - Many useful, interesting problems known to be NP-complete
- NP-Completeness: the “hardest” problems in NP.
- Most problems in NP-Complete are equivalent – reductions key to showing this!

Reductions: a useful tool

- Definition: To “reduce A to B” means to solve A, given a subroutine solving B.
- Reduce Median to Sort
  - ?
- Reduce Sort to Find_Max
  - ?
- Reduce Median to Find_Max
  - ?

10 minutes challenge

Reductions: a useful tool

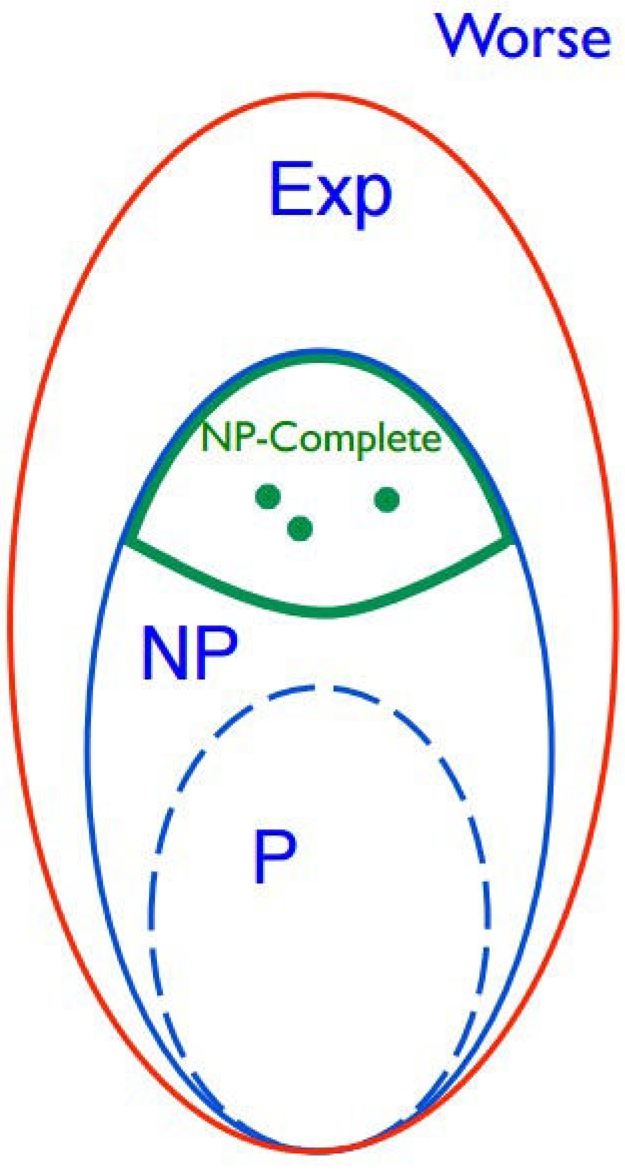
- Definition: To “reduce A to B” means to solve A, given a subroutine solving B.
- Reduce Median to Sort
  - Solution: sort, then select $(n/2)^{nd}$
- Reduce Sort to Find_Max
  - FIND_MAX, remove it, repeat
- Reduce Median to Find_Max
  - Transitivity: compose solutions above

P-time Reductions

- To reduce A to B means to solve A, given a subroutine to solve B
- Fast algorithms for B implies a fast algorithm for A
- If every algorithm for A is slow, then no algorithm for B can be fast

NP-Completeness

- Problem B is NP-complete if: B belongs to NP and every problem in NP is polynomially reducible to B
- These are the hardest problems
- Solving any solves them all!



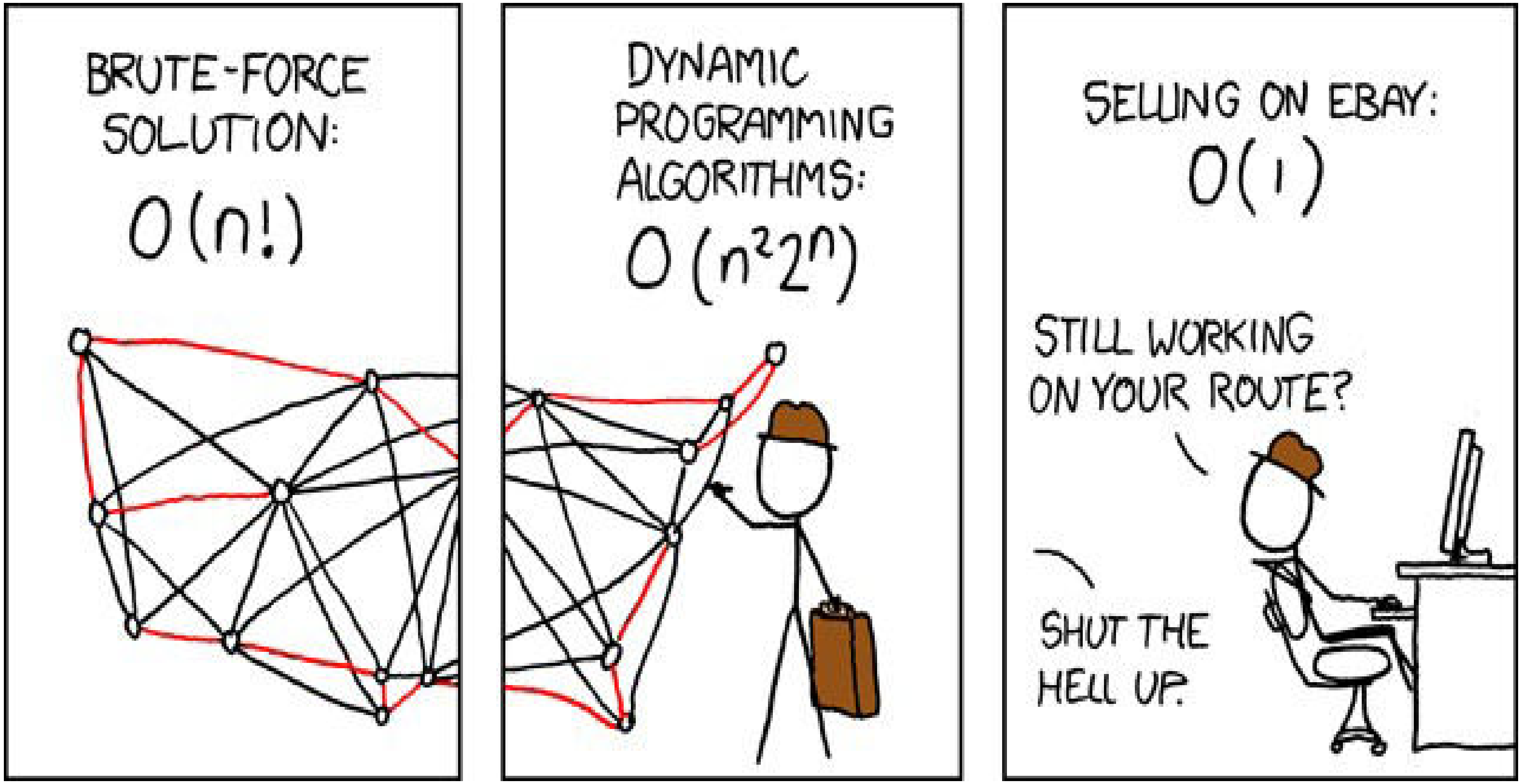
NP-completeness

- The general belief is that there is no efficient algorithm for any NP-complete problem, but no proof of that belief is known
- $1 million dollar prize to prove P=NP?
-

Summary

- Big-O – good
- P – good
- Exp – bad
- Exp, but solution helps? NP
- NP-complete – bad
- To show NP-complete – reductions
- Is NP-complete hopeless?
-

Traveling Saleman Problem: O(1)



NP-complete = hopeless?

- No, but you need to lower you expectations with heuristics, approximations, small instances...
- Guaranteed approximations good enough?
  - Traveling Saleman Problem: NN Heuristic
- Fast enough in practice if n is small
  - Clever exhaustive dynamic programming, backtracking, pruning
- Heuristics to find a fast good aproximation

Is Towers of Hanoi in NP? NP-Complete?

- ?