Nondeterministic Polynomial Time

Discrete Structures (CS 173) Fall 2016

Gul Agha

Slides based on Derek Hoiem, University of Illinois

2016 CS Alumni Awards

Sohaib Abbasi (BS ’78, MS ’80), Chairman & CEO (retired) of Informatica, speaks of the history of Illinois CS alums

https://youtu.be/3EVTs8tKi18

This class

- Multiplying large numbers (review)
- Algorithmic complexity
- P vs. NP

Example: multiplying large numbers

Multiplying small numbers in binary

$$\begin{array}{r} 101 \\ \times\ 011 \end{array}$$

Complexity:

Multiplying large numbers

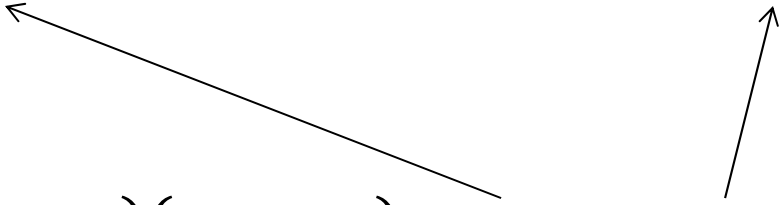
$$x = x_1 2^m + x_0$$
$$y = y_1 2^m + y_0$$
$$\begin{aligned} xy &= (x_1 2^m + x_0)(y_1 2^m + y_0) \\ &= x_1 y_1 2^{2m} + (x_0 y_1 + y_0 x_1) 2^m + x_0 y_0 \end{aligned}$$

Complexity:

Example: multiplying large numbers

Multiplying large numbers

$$x = x_1 2^m + x_0$$
$$y = y_1 2^m + y_0$$
$$xy = (x_1 2^m + x_0)(y_1 2^m + y_0)$$
$$= x_1 y_1 2^{2m} + (x_0 y_1 + y_0 x_1) 2^m + x_0 y_0$$



Trick by Anatolii Karatsuba

$$(x_0 y_1 + y_0 x_1) = (x_1 + x_0)(y_1 + y_0) - x_1 y_1 - x_0 y_0$$

Complexity:

P vs. NP

A problem is **class P** if a polynomial-time solution exists

A problem is **class NP** (non-deterministic polynomial time) if a solution can be checked in polynomial time, but no known algorithm can generate the solution in polynomial time

Examples

Boolean satisfiability: Determine if any assignment of n boolean variables can satisfy a set of logical expressions

E.g., $a \wedge (b \vee \neg c \vee d) \wedge (\neg a \vee d \vee \neg b)$ is a 3-SAT problem

Boolean Satisfiability

How fast can you find a solution to 3-SAT?

How fast can you check a solution to 3-SAT?

NP: finding solution takes exponential time, but checking solution is polynomial

Examples

Sorting a set of integers..

How fast can you find a solution?

P: sorting takes linearithmic time

How fast can you check a solution?

Checking takes linear time

Examples

(1) Determining if a graph of size n is k-colorable
   - k^n brute force search
   - $O(2^n n)$ algorithm exists, for any k

(2) Determining the chromatic number of a graph.

(1) is in **NP**, can be checked quickly. It is NP-complete.

(2) is in NP-hard (not necessarily in NP)

P = NP?

- If a problem can be checked efficiently (class P), then can we solve it efficiently?
  - Yes: $P = NP$
  - No: $P \neq NP$
- Introduced by Stephen Cook in 1971
  - Notion of NP-complete introduced in the same paper
- Proof is worth $1,000,000 (Millenium Prize Problem)

Problem Reduction

- Problem *P* can be *reduced to* Problem *Q* if we can transform an instance of *P* into an instance of *Q*. Thus:
  - Given an algorithm for *Q*, we can use it to find an algorithm for *P*
- Problem *P* can be *reduced to* Problem *Q* in *polynomial time* if we can transform an instance of *P* into an instance of *Q in polynomial time*
  - *Q* is at least as hard as *P*

Example Problem Reduction

We can reduce the problem of multiplication to the problem of squaring—assuming we have addition, subtraction and division operations—using the formula:

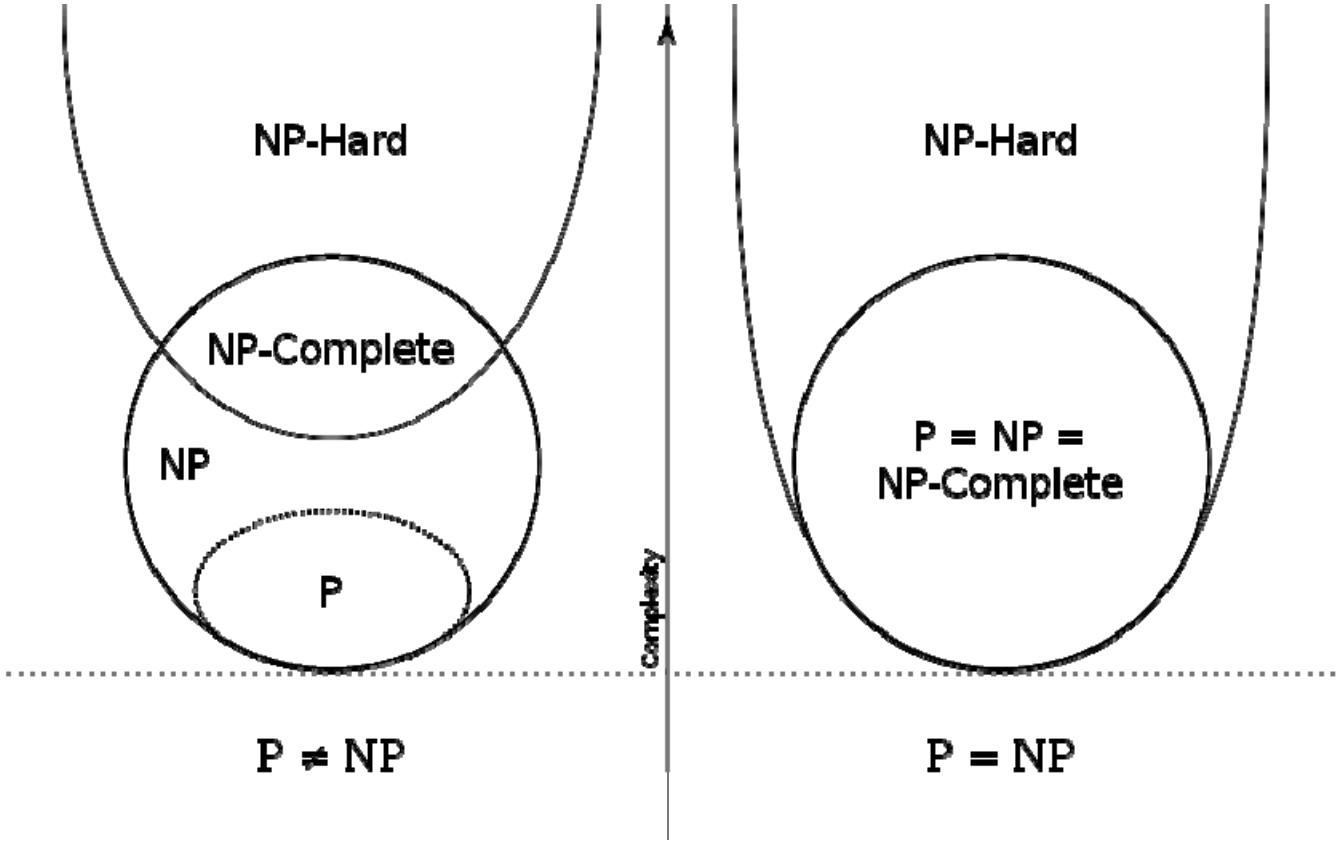
$$a \times b = \frac{(a+b)^2 - a^2 - b^2}{2}$$

NP-complete and NP-hard

- Any NP problem can be reduced in polynomial time to an **NP-complete** problem

  Example: satisfiability

- If any NP-complete problem can be solved in polynomial time, then all NP problems can be solved in polynomial time
- A problem is **NP-hard** if an NP-complete problem can be reduced to it
- If $P \neq NP$ then no NP-complete algorithm can be solved in polynomial time



Wikidpedia diagram

1. Everything inside the circle is in NP.
2. All NP problems can be reduced an NP-Complete problem in polynomial time.
3. A problem is NP-Hard if an NP problem can be reduced to it. An NP-Hard problem may or may not be in NP.

Examples

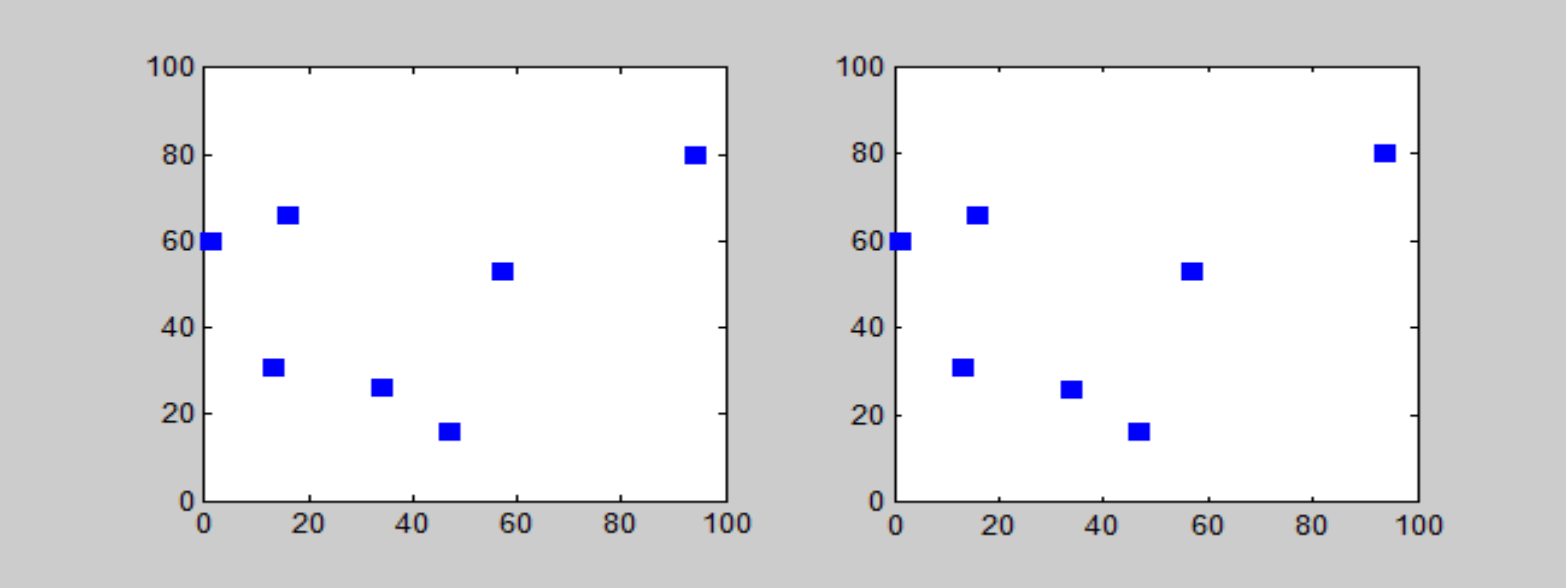
Traveling salesman problem: determine an order of cities to visit that minimizes total travel time without visiting the same city twice.

mathworld.wolfram.com

Find the most efficient (with the least weight) Hamiltonian cycle (circuit) through a graph

Traveling Salesman

- *Brute force:* branch on all $(n-1)!/2$ possibilities.
  Example: $n = 8$, 7! / 2 = 360 combinations.
- *Branch and Bound:* use best known solution thus far to *prune* the search (bound).



By Saurabh.harsh - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=23385823

- Traveling Salesman Problem (TSP) is *NP-Hard*

Traveling Salesman Problem variants

- Is their a tour covering all the cities whose cost is less than a given bound B.
- This problem is in NP: A solution can be checked in nondeterministic polynomial time:
  1. just walk through the tour and add the cost as you go.
  2. At the end check if every city is covered and that the total cost is less than the bound B.

(Occassionally, some textbooks call this problem TSP)

How large is $n!$

$$\ln n! = \ln 1 + \ln 2 + \cdots + \ln n$$

$$\ln 1 + \ln 2 + \cdots + \ln n - \frac{1}{2}(\ln 1 + \ln n) = \frac{1}{2}\ln n$$

$$\ln n! - \frac{1}{2}\ln n \approx \int_1^n \ln n \, dx = n \ln n - n + 1$$

(Trapezoid rule)

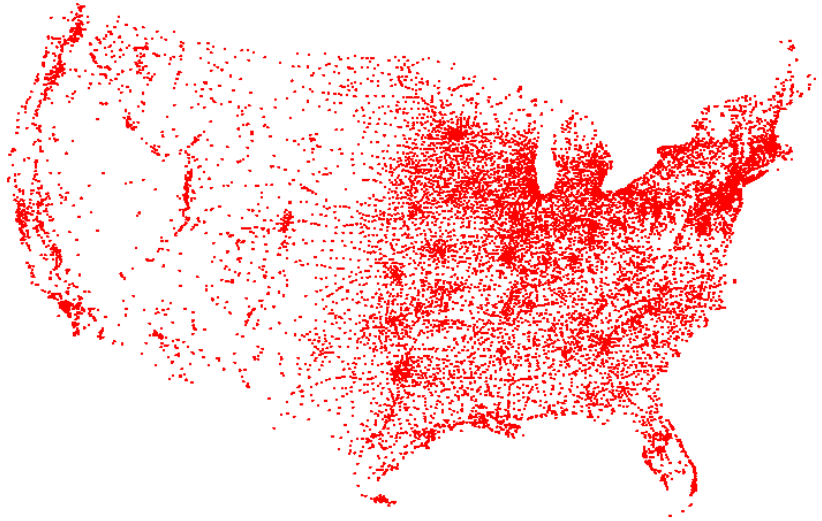
$$\ln n! = n \ln n - n + O(\ln n)$$

Stirling's Formula:

$$n! \sim \sqrt{2\pi n}(n/e)^n$$

Exact TSP solutions

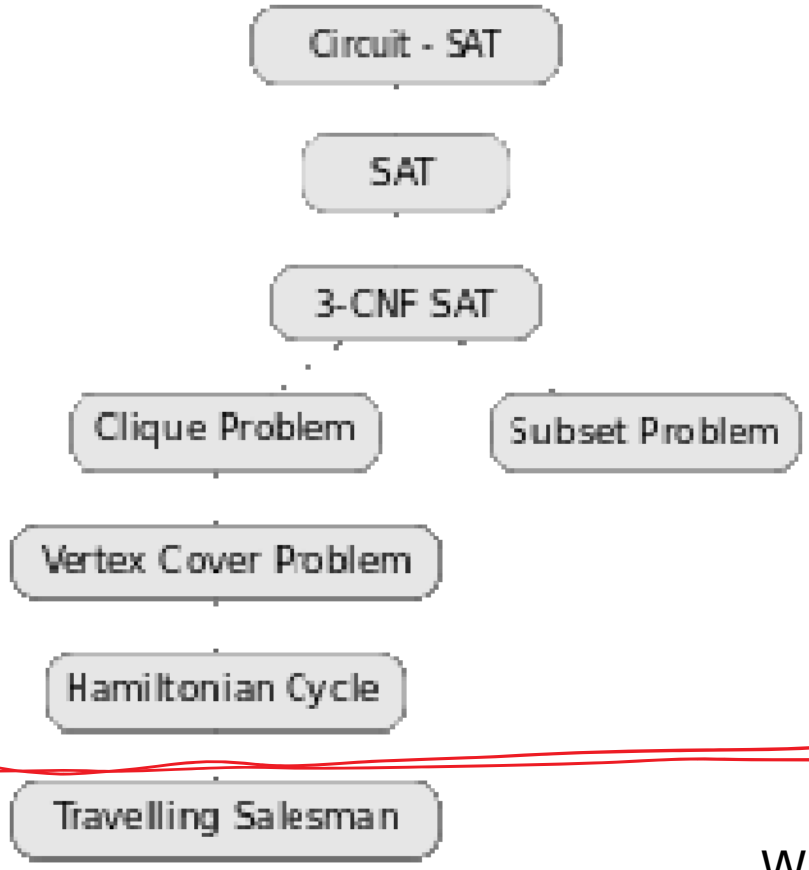
- 1998: A tour of 13,509 US cities (population > 500). There are 19,358 US cities and towns according the Census Bureau).



- 2004: A tour of 24,978 cities in Sweden.
- 2006: Hamiltonian circuit 85,900 locations in a VLSI circuit were solved.

NP Complete Problems

- Some NP-complete problems and typical reductions
- Reductions are transitive
- TSP is NP-hard



Wikipedia

NP-hard

Alternatives to Exact Solutions

- Heuristics: e.g. Monte Carlo Methods
- Approximations.
  - *TSP* in planar graphs is still NP-hard
  - Can find approximations (in this case, twice the length)

http://courses.csail.mit.edu/6.889/fall11/lectures/L15.html

Other NP-complete examples

- Lemmings
- Candy Crush Saga
- Battleship
- Set cover

Things to remember

- Be able to analyze code for computational cost
  - Tools: finding loops and recursive calls, using recursion trees
  - Sometimes need to know inner-workings of a library to determine (e.g., factorialSeries)
- Be able to convert to big-O or big-Theta and be familiar with basic complexity terms
  - E.g., linear, nlogn, polynomial, exponential
- Problems in NP can be checked in polynomial time but probably not solved in polynomial time
  - P=NP is open problem, most think not