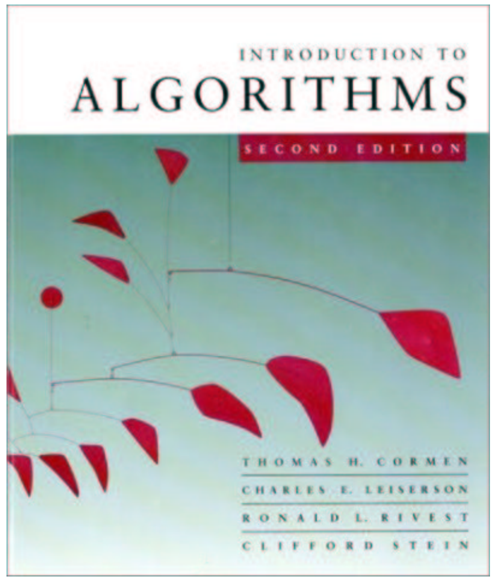
Introduction to Algorithms

6.046J/18.401

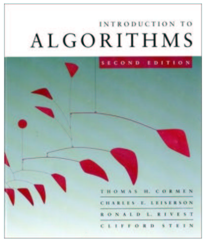


Lecture 21

Prof. Piotr Indyk

P vs NP (interconnectedness of all things)

- A whole course by itself
- We’ll do just two lectures
- More in 6.045, 6.840J, etc.

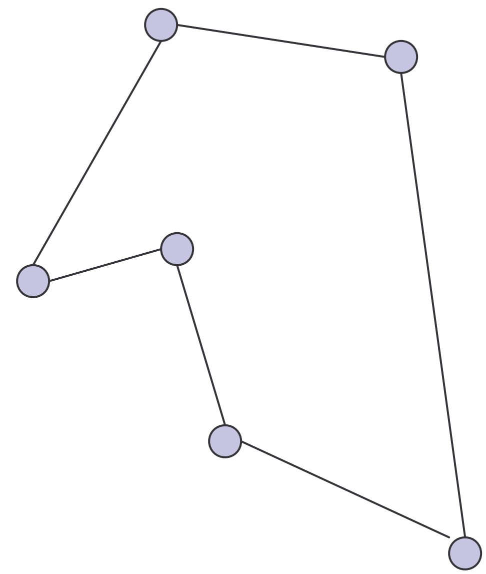


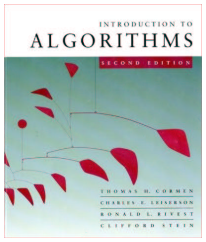
Have seen so far

- Algorithms for various problems
  - Running times $O(nm^2)$, $O(n^2)$, $O(n \log n)$, $O(n)$, etc.
  - I.e., polynomial in the input size
- Can we solve all (or most of) interesting problems in polynomial time ?
- Not really…

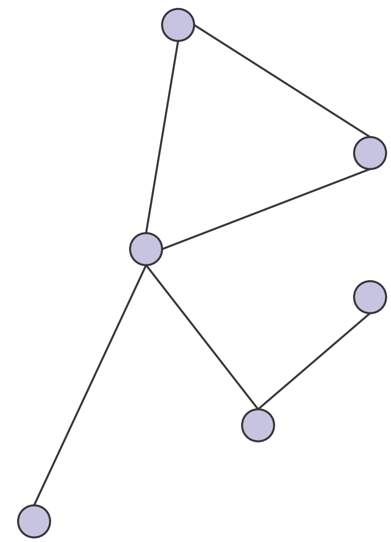
Example difficult problem

- Traveling Salesperson Problem (TSP)
  – Input: undirected graph with lengths on edges
  – Output: shortest tour that visits each vertex exactly once
- Best known algorithm: $O(n\,2^n)$ time.

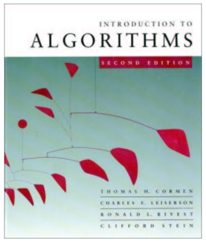




Another difficult problem

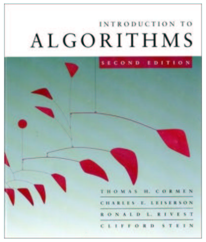


- Clique:
  - Input: undirected graph G=(V,E)
  - Output: largest subset C of V such that every pair of vertices in C has an edge between them
- Best known algorithm: $O(n\,2^n)$ time



What can we do ?

- Spend more time designing algorithms for those problems
  - People tried for a few decades, no luck
- Prove there is no polynomial time algorithm for those problems
  - Would be great
  - Seems *really* difficult
  - Best lower bounds for “natural” problems:
    - $\Omega(n^2)$ for restricted computational models
    - $4.5n$ for unrestricted computational models

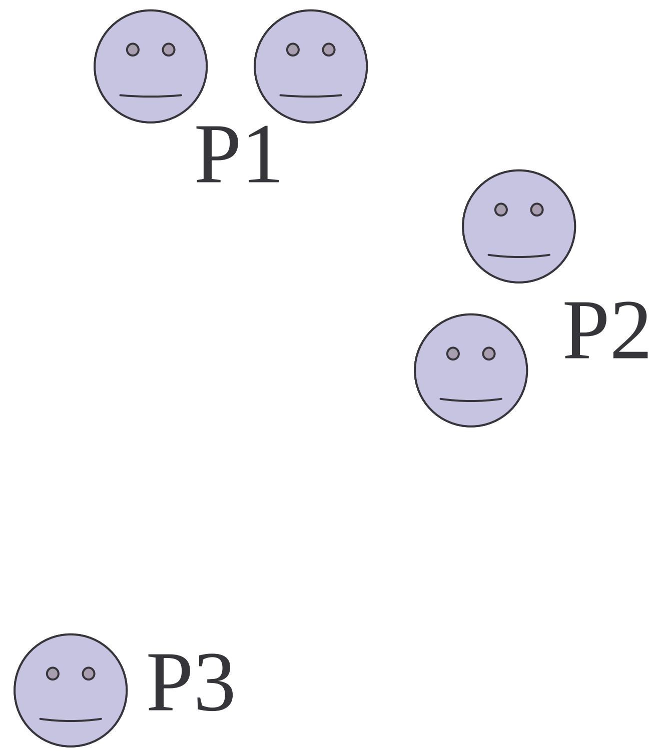


What else can we do ?

- Show that those hard problems are essentially equivalent. I.e., if we can solve one of them in poly time, then all others can be solved in poly time as well.
- Works for at least 10 000 hard problems

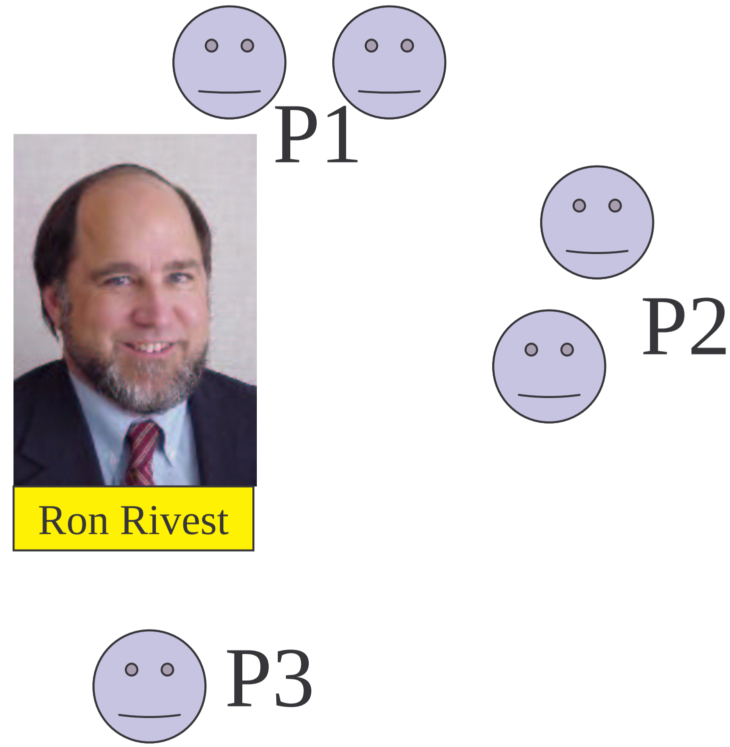
The benefits of equivalence

- Combines research efforts
- If one problem has polytime solution, then all of them do



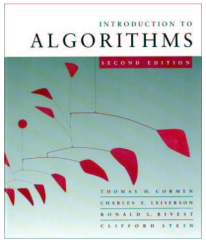
A more realistic scenario

- Once an exponential lower bound is shown for one problem, it holds for all of them
- But someone *is* happy…



Summing up

- If we show that a problem $\prod$ is equivalent to ten thousand other well studied problems without efficient algorithms, then we get a very strong evidence that $\prod$ is hard.
- We need to:
  - Identify the class of problems of interest
  - Define the notion of equivalence
  - Prove the equivalence(s)



Class of problems: NP

- Decision problems: answer YES or NO. E.g.,"is there a tour of length ≤ K" ?
- Solvable in *non-deterministic polynomial* time:
  - Intuitively: the solution can be verified in polynomial time
  - E.g., if someone gives as a tour T, we can verify if T is a tour of length ≤ K.
- Therefore, TSP is in NP.

Formal definitions of P and NP

- A problem $\prod$ is solvable in poly time (or $\prod \in P$), if there is a poly time algorithm $V(.)$ such that for any input x:

$$\prod(x)=YES \text{ iff } V(x)=YES$$

- A problem $\prod$ is solvable in non-deterministic poly time (or $\prod \in NP$), if there is a poly time algorithm $V(. , .)$ such that for any input x:

$$\prod(x)=YES \text{ iff there exists a certificate } y \text{ of size } \mathrm{poly}(|x|) \text{ such that } V(x,y)=YES$$

Examples of problems in NP

- Is "Does there exist a clique in G of size ≥K" in NP ?

  Yes: V(x,y) interprets x as a graph G, y as a set C, and checks if all vertices in C are adjacent and if |C|≥K

- Is Sorting in NP ?

  No, not a decision problem.

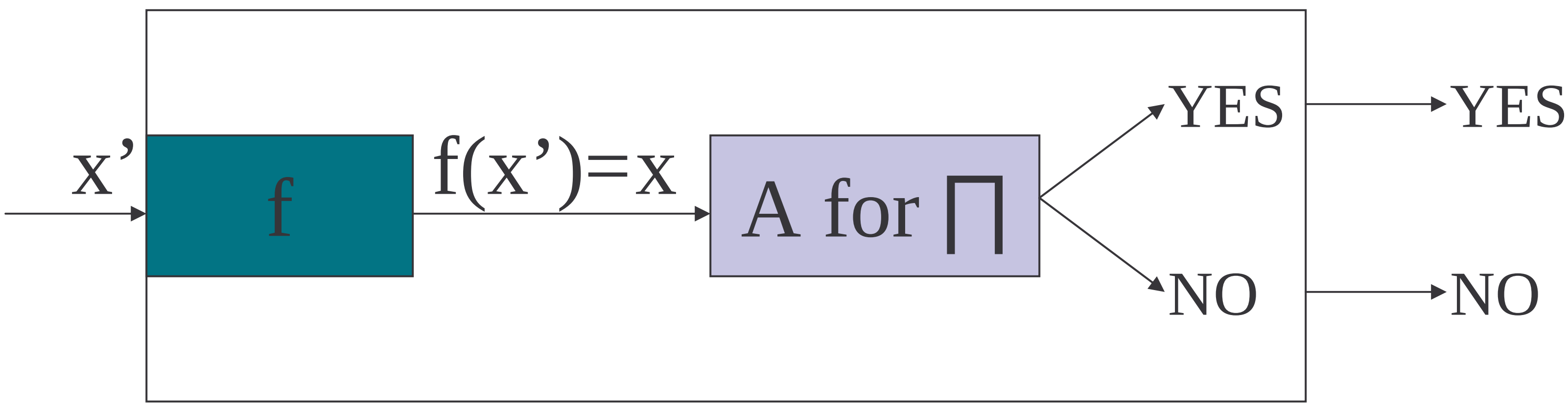
- Is "Sortedness" in NP ?

  Yes: ignore y, and check if the input x is sorted.

- Is Compositeness in NP ?

  Yes. In fact, for V as in Lecture 17, there are many certificates y.

Reductions: ∏’ to ∏



A’ for ∏’

Reductions

- $\prod'$ is poly time reducible to $\prod$ ($\prod' \le \prod$) iff there is a poly time function f that maps inputs x' to $\prod'$ into inputs x of $\prod$, such that for any x'

$$\prod'(x')=\prod(f(x'))$$

- Fact: if $\prod \in P$ and $\prod' \le \prod$ then $\prod' \in P$
- Fact 2: if $\prod \in NP$ and $\prod' \le \prod$ then $\prod' \in NP$
- Fact 3: if $\prod' \le \prod$ and $\prod'' \le \prod'$ then $\prod'' \le \prod$

Recap

- We defined a large class of interesting problems, namely NP
- We have a way of saying that one problem is not harder than another $(\Pi' \leq \Pi)$
- Our goal: show equivalence between hard problems