Advanced Algorithms

Lecture 25: Limits on efficient computation

Lower bounds on computation

- Can we search for an element x in a sorted, n-element array in time $< \log n$?
- Can we solve the shortest path problem in time $O(m+n)$ on all graphs?
- Can we multiply two n x n matrices in time $O(n^2)$?
- Can we factor an n digit number in poly(n) time?

open

Challenge in lower bounds: **must reason about an algorithm without knowing what it is!**

The model matters!

Problem. can we search for an element x in a sorted, n-element array in time < log n?

- What operations are "allowed"? [*if only comparisons, then there's a lower bound of log n*]
- What about randomness? [*makes things tricky*]

Yao's minimax principle.

Last class

- In limited models, can show **“unconditional”** lower bounds
- **Key question:** is there a model that captures “all computations”
- Turing machine / RAM model — [Church-Turing thesis]
- Move to decision problems (YES/NO answers)

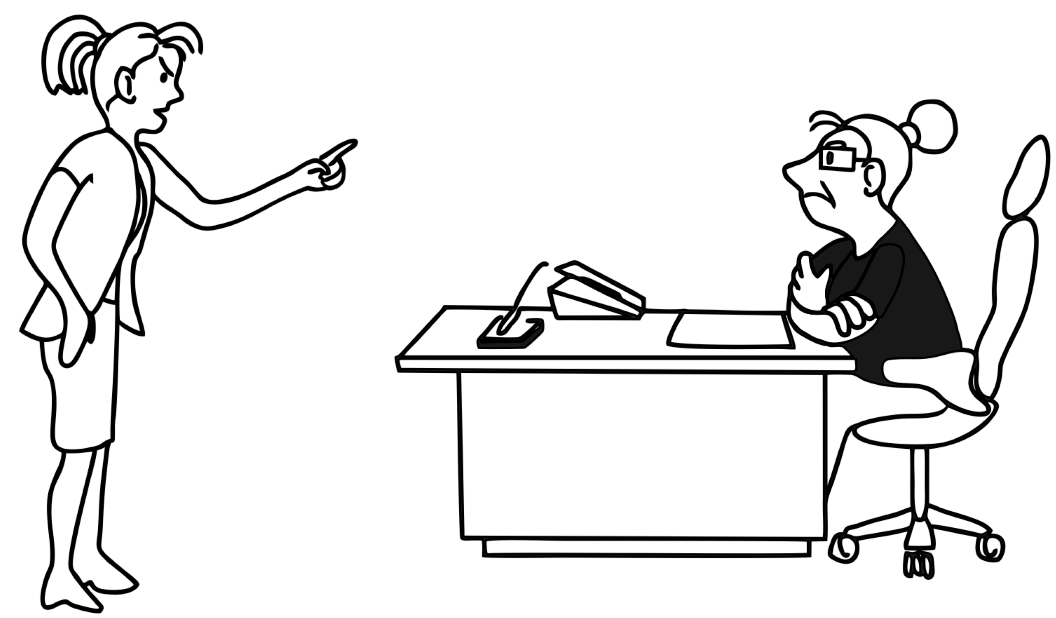
  – Fairly general way of moving to decision versions.

  (Ind Set)
- Lower bounds for TMs hard to prove!

  $\boxed{10\text{-}n}$

Lower bounds

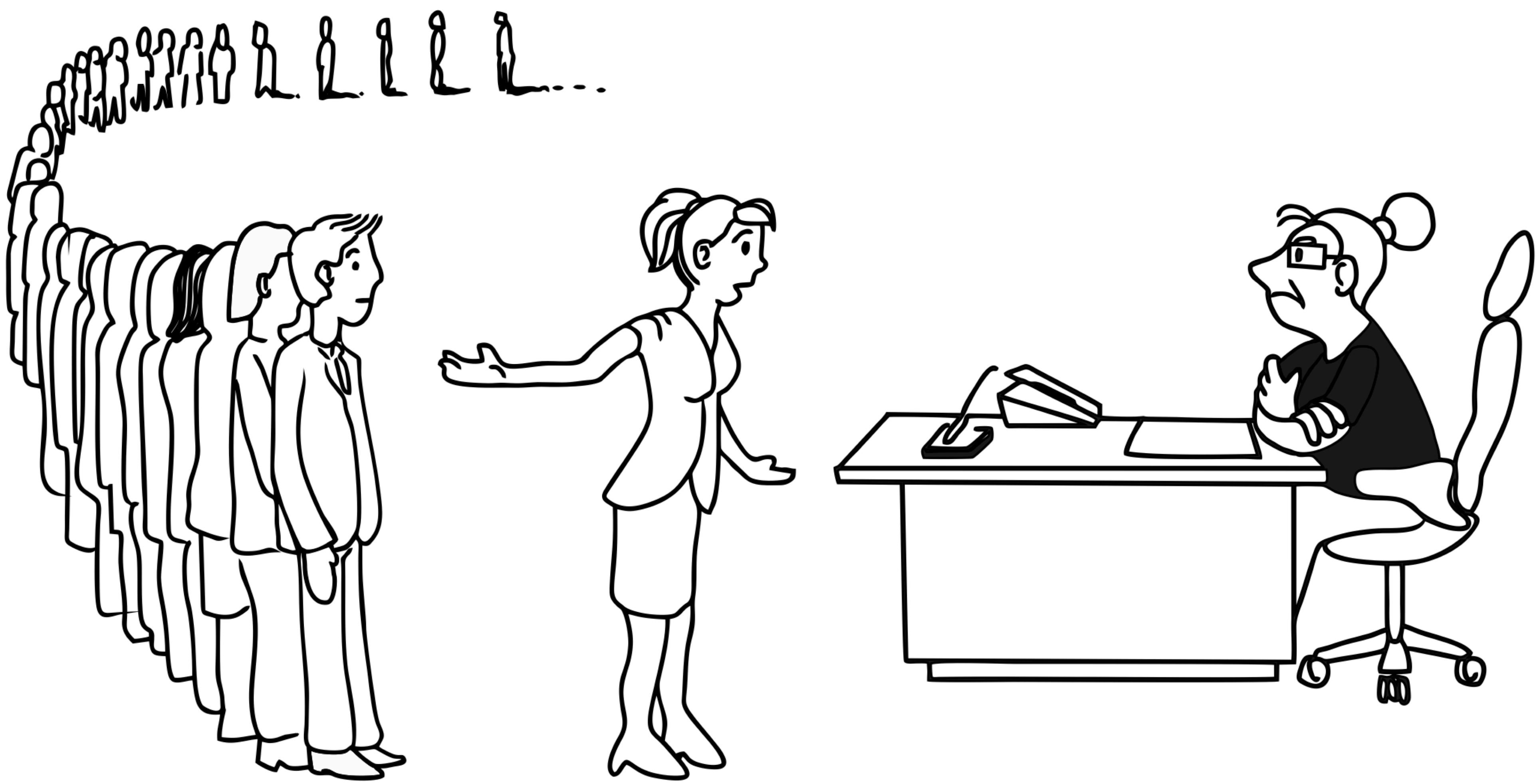
“I can’t find an efficient algorithm, I guess I’m just too dumb.”



“I can’t find an efficient algorithm, because no such algorithm is possible!”

Conditional lower bounds

Idea behind "reductions".



"I can't find an efficient algorithm, but neither can all these famous people."

Complexity classes

Group problems into “classes” — *equally difficult/easy*

The class P

Definition. set of all decision problems that can be solved in polynomial time (on a Turing machine).

[definition of the class does not allow randomness].

- Does graph have a path of length <= L between u and v?
- Does graph have a spanning tree of total cost <= C?

All problems solvable in poly time on a non-det. Turing machine.

The class NP

Non-deterministic polynomial time

Definition. the set of all decision problems for which there is a polynomial time "verification algorithm"

verification alg.

- **Recall (independent set).** does a graph have an independent set (set of vertices with no edges) of size k?

For any YES instance of problem,

- There is a "witness" that can convince someone that the answer is YES.

- Given a NO instance, there is no way to convince ... that answer is YES.

G, k.

witness: simply the independent set S.

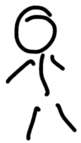
Verification alg:
- check that $|S| > k$.
- check that there are no edges between vertices of S.

$O(n^2)$ time algorithm.

- if both conditions are met, return YES
- else return NO.

Verification Procedure

Prover



generate S.

$G = (V, E)$, k

Verifier

$S \rightarrow$ $\boxed{\text{Alg.}}$ — runs in poly time.

- check if $S \subseteq V$, $|S| \geq k$
- checks that there are no edges.

* → If (G, k) is a YES instance then $\exists\, S$ that prover can use to make verifier output YES.

* → If (G, k) is actually a NO instance, there is NO choice of S that can make verifier output YES.

Witnesses and NP

S : <u>witness or certificate</u>.

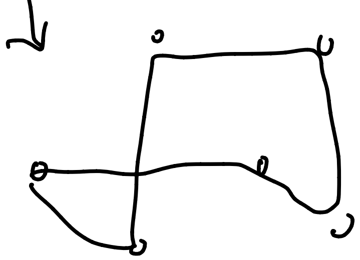
$\square$: verification algorithms.

Problems in NP

We don't know for sure if $P \neq NP$.

no need to look at certificate.

- All problems in P
- Most "puzzles" ⇝ $n \times n$ Sudoku
- Traveling salesman problem
- Boolean "satisfiability"



G, dists, target length of tour L.

Witness: Solution S to puzzle

Verifier: a procedure that checks validity of S.

Witness: Order in which to visit vertices.

Verifier: Verify that every vertex in G shows up exactly once & total len $\leq L$.

input size $= s = \log N$ ← polynomial in # bits used to write N.

Primality testing: given N, return YES if # is prime & NO if # is composite.

75431421

Certificate? list all primes $\leq \sqrt{N}$. | Verification algo?

$2^{s/2}$

→ Pratt '60s:

Other complexity classes

mxn chess.

games:

- PSPACE — polynomial "space"

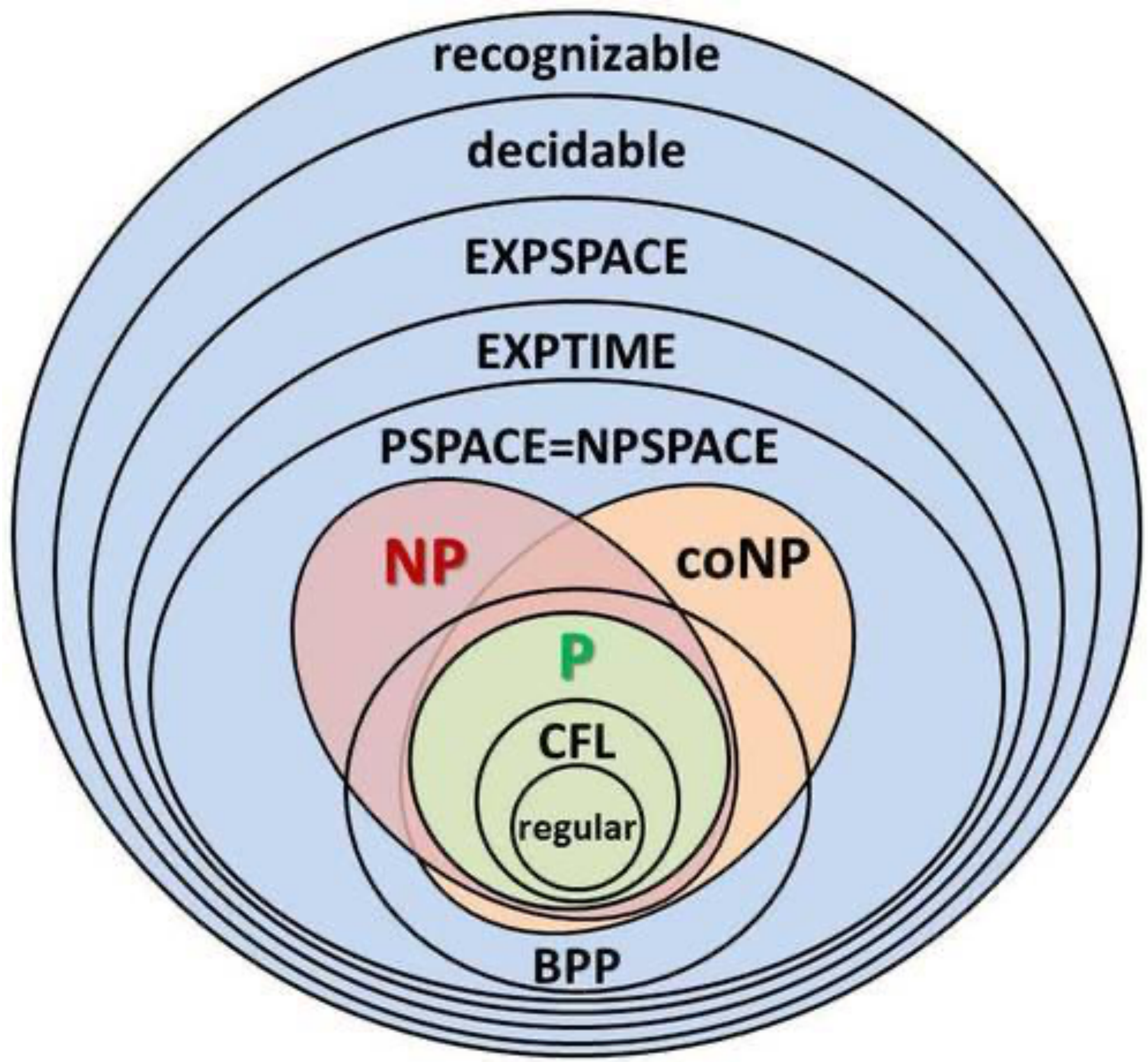
- BPP — randomized algorithms

(YES/NO problems for which there is a prob. polynomial time ALG that returns right answer with 90% prob.).

- #P — counting class

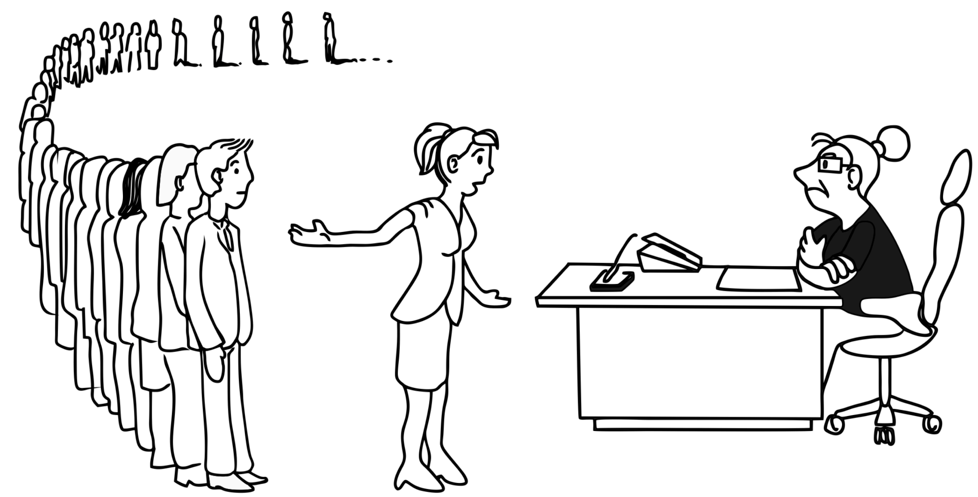
RP: lop-sided version.

Complexity “classes”



Reductions

Informal vs formal…



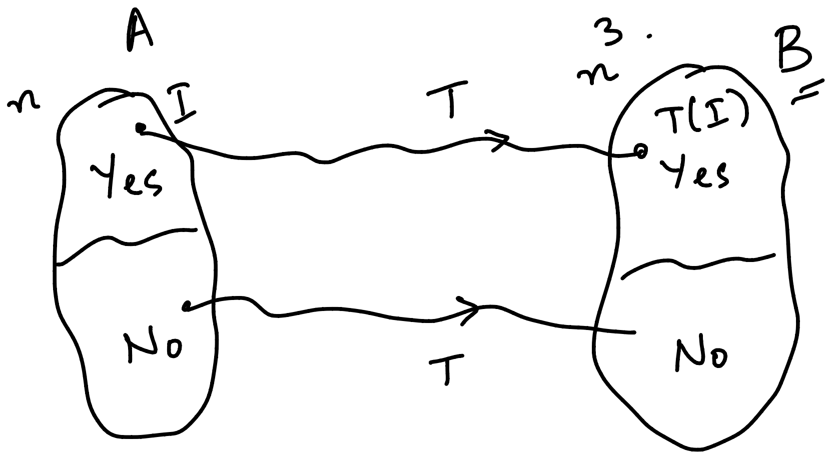
"I can't find an efficient algorithm, but neither can all these famous people."

Informal: suppose we have problems A, B. Then we say that A reduces to B if given an algorithm for B, we can find an alg. for A of "roughly same" complexity.

Formal defn: [Polynomial time reduction].

Let A & B be two decision problems. Then we say that $A \leq_p B$ if there exists a poly time procedure T that transforms any instance I of $A \rightsquigarrow$ instance $T(I)$ of B, such that

- if I is a YES inst. for A, $T(I)$ is a YES for B
- " NO " " NO "



Alg for B
$|\text{Input}|^2$

$\Downarrow$

Alg for A,
$|\text{Input}|^6$.

- T <u>operates in polynomial</u> time

★ T does not know if I is a YES or a NO instance

NP-hard and NP-complete

Boolean satisfiability

Cook-Levin theorem