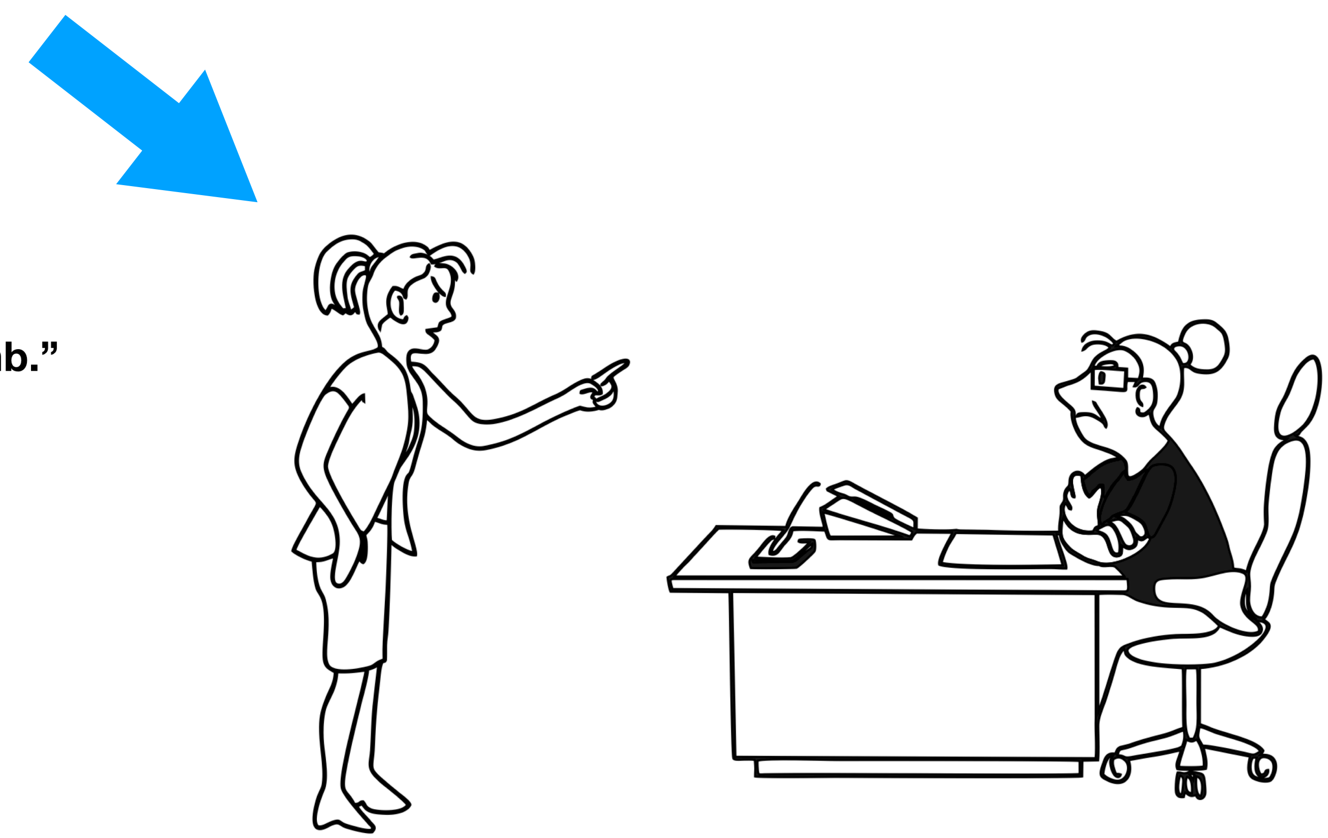
Advanced Algorithms

Lecture 26: Reductions between problems

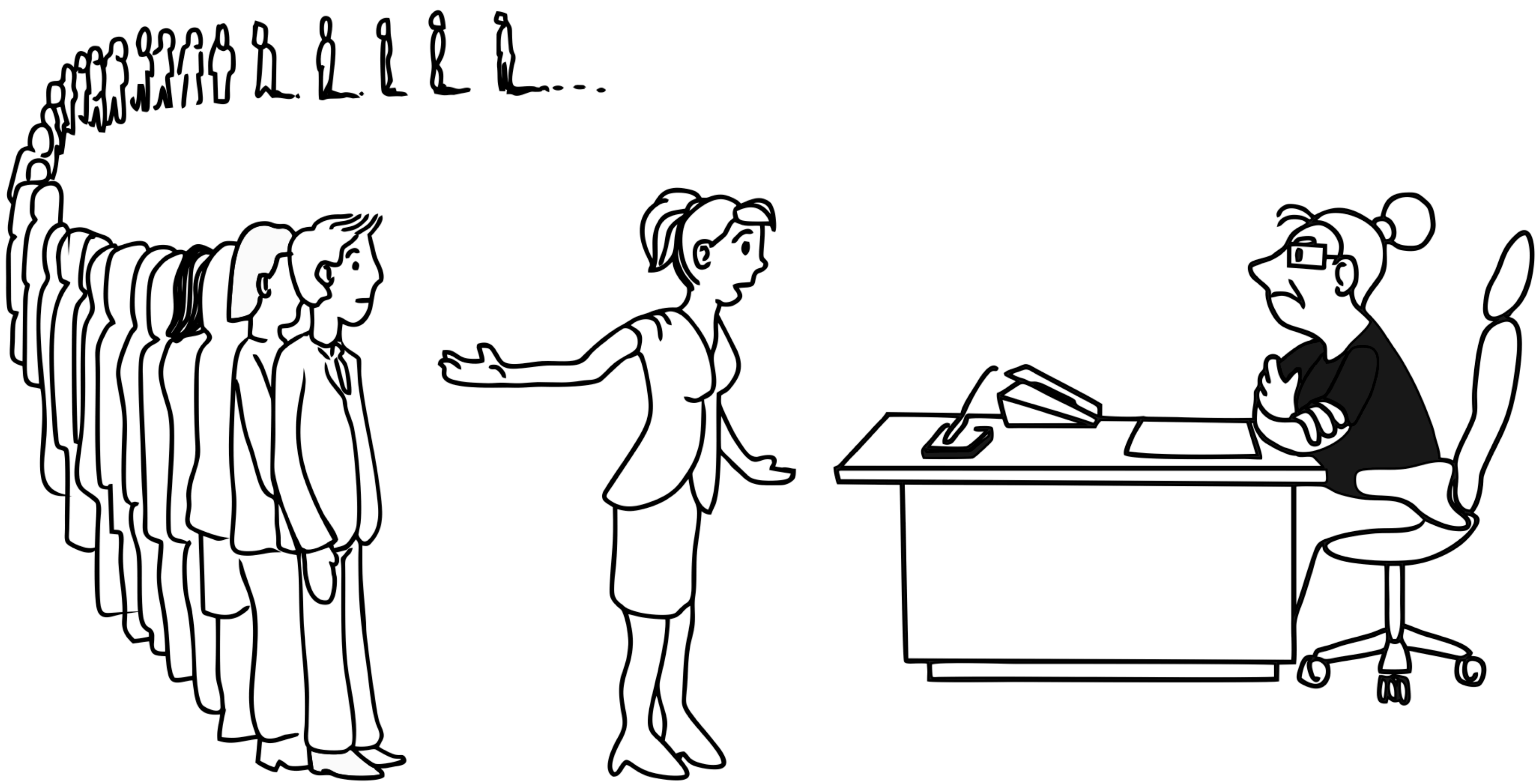
Lower bounds

“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



“I can’t find an efficient algorithm, but neither can all these famous people.”

The class P

Definition. set of all decision problems that can be solved in polynomial time

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

The class NP

Non-deterministic polynomial time

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

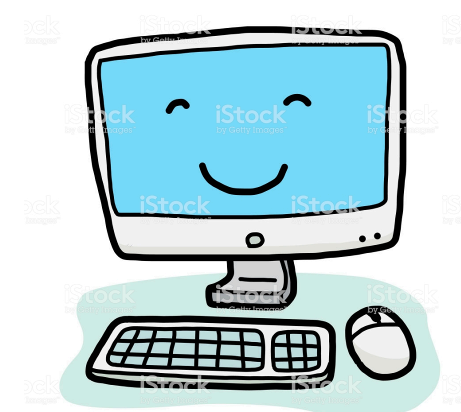
- **Recall (independent set).** does a graph have an independent set (set of vertices with no edges) of size k?

Prover, verifier

Prover

Instance *I*

Verifier

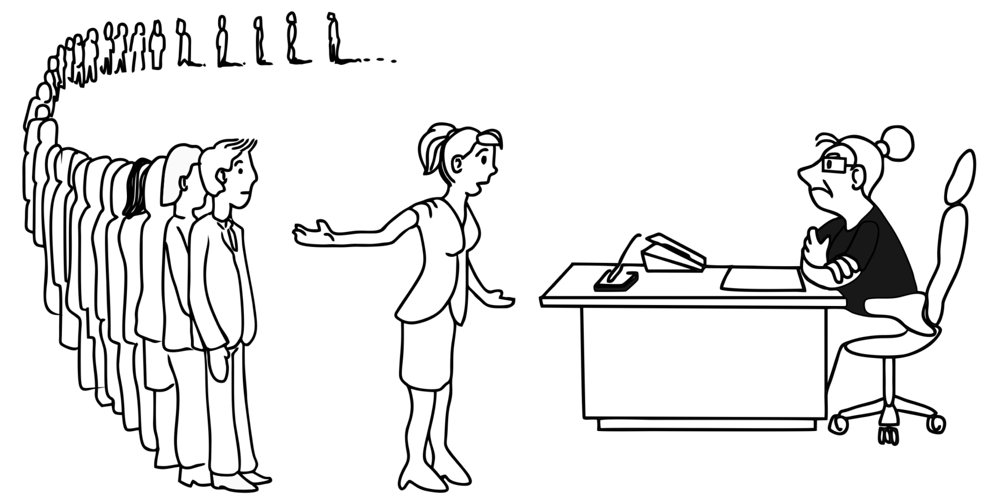


Runs ALG(Instance I, witness/cert w)

- If I is a YES instance, there exists a witness w such that ALG outputs YES
- If I is a NO instance, for *any* witness w, ALG outputs NO

Reductions

Informal. problem A said to reduce to problem B if a polynomial time algorithm for B can be used to obtain a polynomial time alg for A

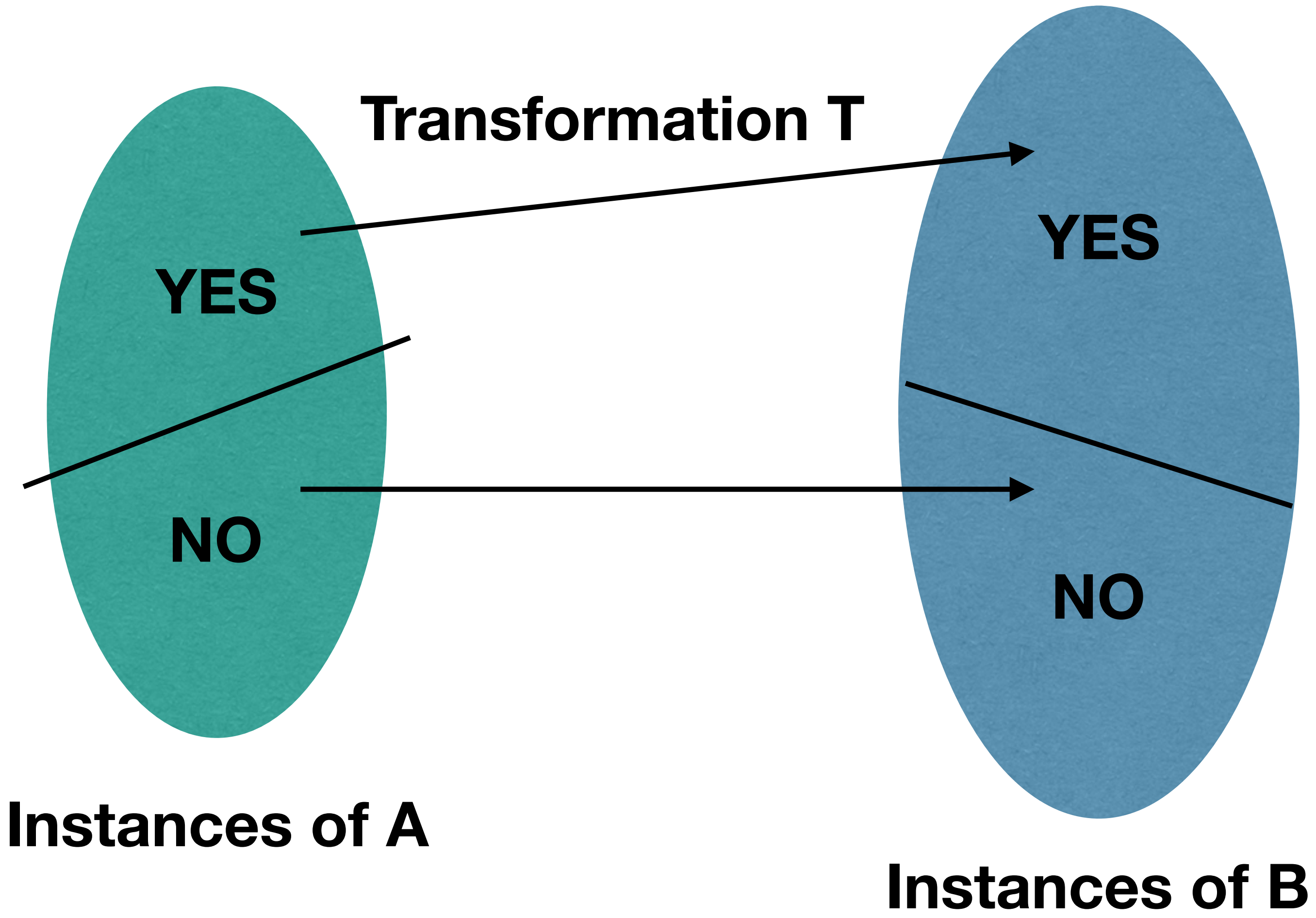


"I can't find an efficient algorithm, but neither can all these famous people."

Definition. a poly-time reduction from problem A to problem B is a transformation that maps instances of A to instances of B s.t. YES instances map to YES ones and NO instances map to NO ones.

$$A \leq_p B$$

A* is "easier" than *B



- T runs in polynomial time
- T has no idea if instance is YES or NO while transforming
- T can be applied to any instance of A, obtaining an instance of B

NP-hard and NP-complete

- **NP-hard:** a problem A is said to be NP-hard if Boolean SAT reduces to A (can replace with Independent Set)

- **NP-complete:** a problem A is said to be NP-complete if Boolean SAT reduces to A **and** A reduces to Boolean SAT

Boolean satisfiability

Cook-Levin theorem

Theorem. any problem in NP (i.e., any problem with a poly time verification oracle) has a poly time reduction to Boolean SAT.

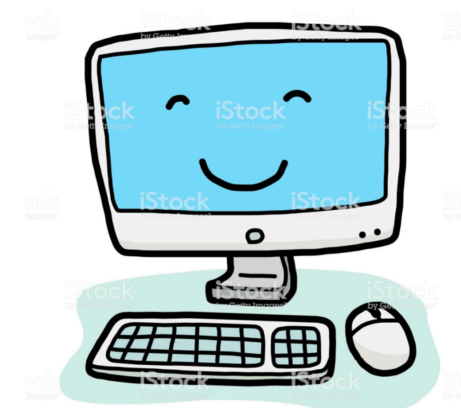
Boolean SAT is the "hardest" problem in NP

Prover, verifier

Prover

Instance *I*

Verifier



Runs ALG(Instance I, witness/cert w)

- If I is a YES instance, there exists a witness w such that ALG outputs YES
- If I is a NO instance, for *any* witness w, ALG outputs NO

Cook-Levin theorem

Theorem. any problem in NP (i.e., any problem with a poly time verification oracle) has a poly time reduction to Boolean SAT.

Hint of proof. bits of the "witness" are the variables; ALG(instance, witness) can be encoded as a boolean circuit!

3-SAT

Independent Set

Hardness of "approximation"