`http://www.cs.unh.edu/~ruml/cs758`

NP-Completeness

Problems, Not Algorithms

P vs NPC vs EXPTIME

- shortest path vs longest path
- Euler tour (each edge) vs hamiltonian cycle (each vertex)
- minimum spanning tree vs
  shortest total all-pairs path length spanning tree
- spanning tree vs vertex cover
- maximum flow vs minimum edge-cost flow (meeting demand)
- minimum cut vs maximum cut
- maximum bipartite matching vs minimum maximal matching
- addition vs subset sum
- 2-CNF satisfiability vs 3-CNF
- interval scheduling vs job shop scheduling
- value of move in checkers, Go

if 1 step $=$ 1 μsecond:

	20	40	60
n	.00002 sec	.00004 sec	.00006 sec
n^2	.0004 sec	.0016 sec	.0036 sec
n^3	.008 sec	.064 sec	.216 sec
n^5	3.2 sec	1.7 min	13 min
2^n	1.0 sec	12.7 days	366 cent
3^n	58 min	3855 cent	10^{13} cent

(non-)effect of CPU speed:

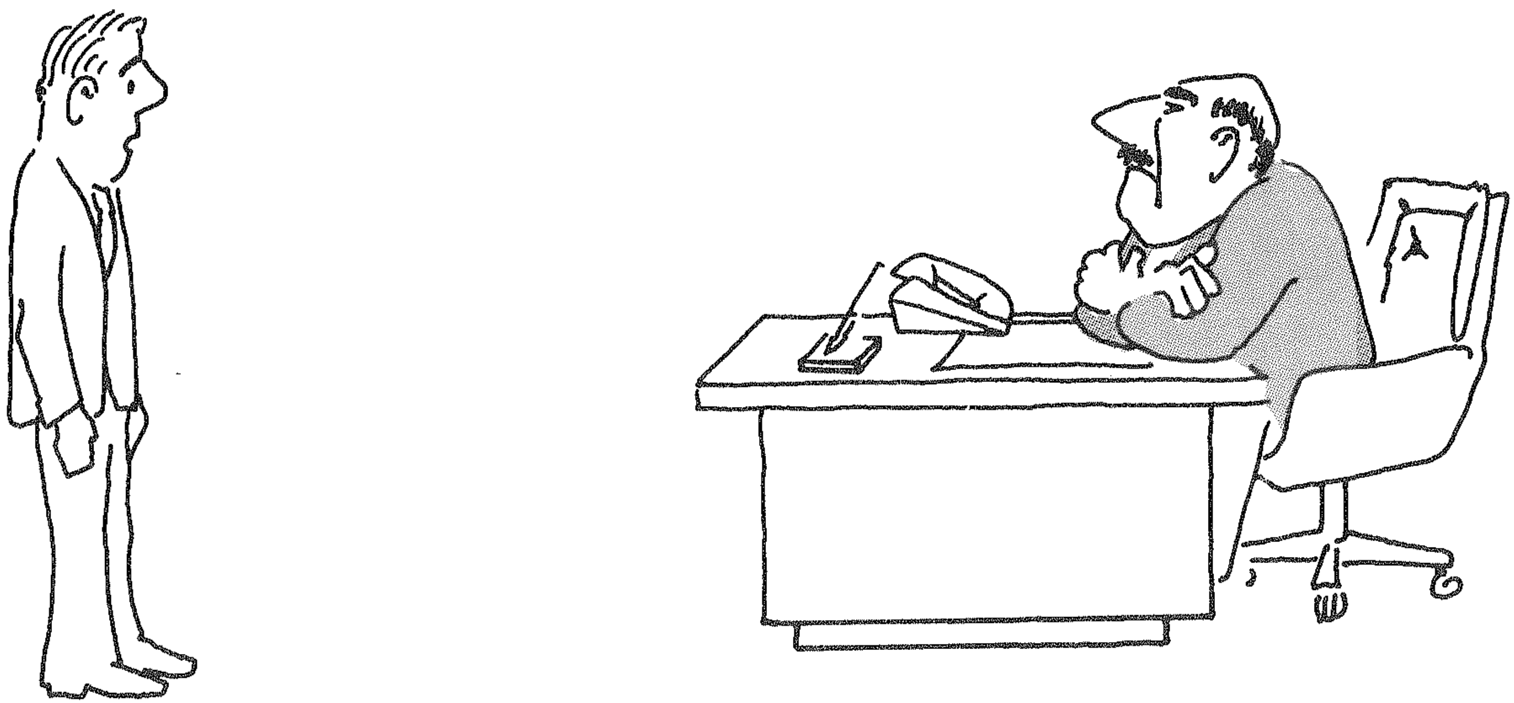
	curr size	$100\times$	$1000\times$
n	N	$100N$	$1000N$
n^2	N	$10N$	$31.6N$
n^3	N	$4.64N$	$10N$
n^5	N	$2.5N$	$3.98N$
2^n	N	$N + 6.64$	$N + 9.97$
3^n	N	$N + 4.19$	$N + 6.29$

tractable: polynomial in (non-unary) input
P: solvable in polynomial time
NP: verifiable in polynomial time (eg, blockchain, cloud computing)
NP-Hard: as hard as any problem in NP (via polytime reduction)
NP-Complete: NP-Hard and in NP

optimization vs decision: if opt were easy, decision would be too

reduce a to b: $a \to b$ in polytime, decide $b, \to$ decision for a

b hard by reduction from a: if $a \to b$ in polytime and b polytime, could solve a

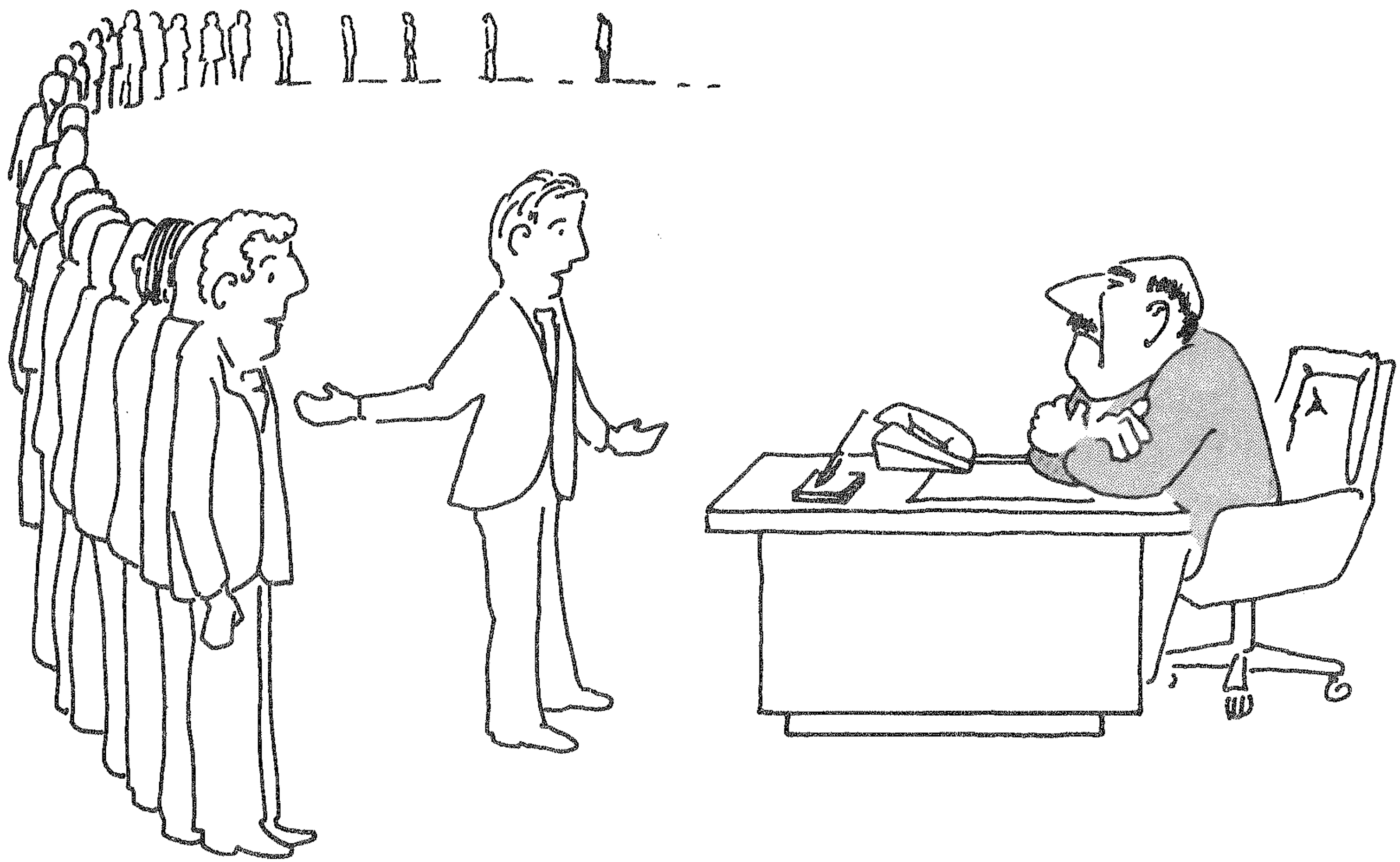


"I can't find an efficient algorithm, I guess I'm just too dumb."

Why

"I can't find an efficient algorithm, because no such algorithm is possible!"

Why



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

Break

- asst 12
- final exam confirmed for Wed Dec 11 3:30-5:30pm in N121
- wildcard vote!

NP

Definitions

P$= \{L \subseteq \{0,1\}^* : \exists$ algorithm that decides L in poly time $\}$

$A(x,y)$ verifies L iff for any input $x \in L$ $\exists$ certificate y that proves $x \in L$ and $\nexists$ certificate iff $x \notin L$

NP$= \{L \subseteq \{0,1\}^* : \exists$ algorithm $A(x,y)$ that can use certificate y with $|y| = O(|x|^c)$ to verify L in polynomial time $\}$

P $\neq$ NP?

co-NP$= \{L \subseteq \{0,1\}^* : \overline{L} \in$ NP $\}$.

NP $\neq$ co-NP? eg $L \in NP \Rightarrow \overline{L} \in NP$?

NP-Completeness

polynomial-time reducible: $L_1 \leq_P L_2$ iff $\exists$

polynomial-time computable function $f : \{0,1\}^* \rightarrow \{0,1\}^*$ such that for all $\{0,1\}^*$, $x \in L_1$ iff $f(x) \in L_2$.

L is NP-Complete iff $L \in$ NP and $\forall L' \in$ NP, $L' \leq_P L$

EOLQs

For example:

- What's still confusing?
- What question didn't you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.
Thanks!