Advanced Algorithms

Lecture 24: Complexity: limits on efficient computation

Announcements

- HW 5 due tomorrow (HW 6 will be released tomorrow.)
- Problem 3 clarification (purely experiment).
- HW 6 logistics

(4-day grace period.).

(Dec 4-6 $\longrightarrow$ Wednesday).

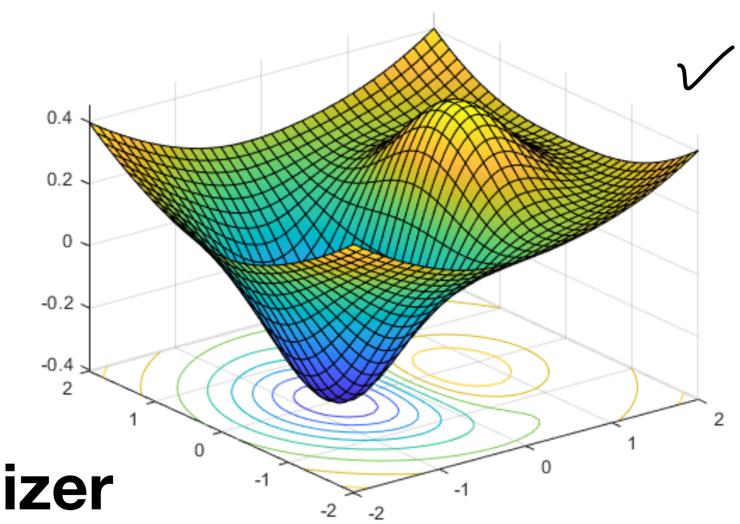
Last two weeks

Instance of problem

Optimization formulation

variables $x_1, x_2, \ldots$
constraints
objective

"relaxation"



Optimizer

opt solution:
$x_1 = 0, x_2 = 1, \ldots$

Solution to problem instance
"rounding"

Optimization formulations

Continuous approaches for discrete problems

- Can sometimes lead to polynomial time algorithms (weighted matching)
- Often used to obtain “approximation algorithms”
- Integrality gap

Lower bounds

- 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?

Lower bounds

- 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?

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

Computational model

Problem. can we search for an element x in a sorted, n-element array in time $< \log_2 n$?

$A[1], \; A[2], \; ---- \;, A[n]$.

- What operations are "allowed"? $\longrightarrow$ .. comparing gives an advantage ..
- What about randomness?

$\longrightarrow$ may / may not help.

A formal lower bound

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

- What operations are "allowed"?
- What about randomness?

queries are of the form:
is $x < A[i]$?

Theorem. consider any deterministic algorithm for "search" problem that can only access array via comparisons. Then algorithm must take at least $\log_2 n$ steps (comparisons).

(we don't know what the algorithm is!).

A formal lower bound

ALG:

query 1: $x < A[i]$?

query 2:

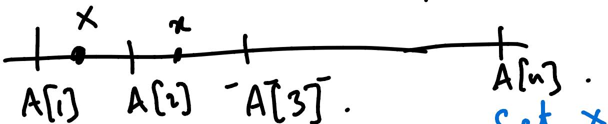
⋮

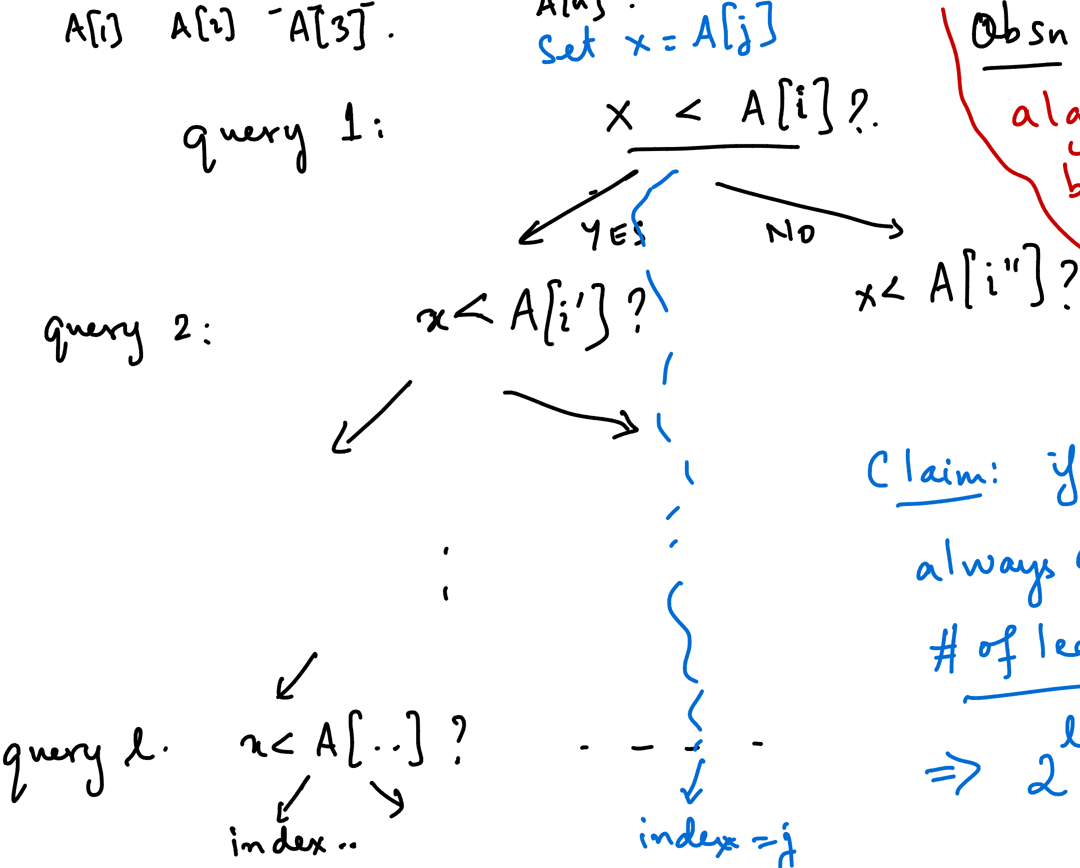
query ℓ: $x \leq \cdots$

→ needs to output index of x in the array.

what if algorithm needs to output only YES/NO (element $x \notin$ array or not.).

$A[1], \ldots, A[n]$. # to find $= x$.





Obsn: Any deterministic alg is characterized by this "decision tree".

Claim: if algorithm is always correct, then # of leaves $\geq n$.

$\Rightarrow 2^{l} \geq n \Rightarrow l \geq \log_2 n$

Lower bounds

- Lower bounds for "limited" models very useful in algorithm design

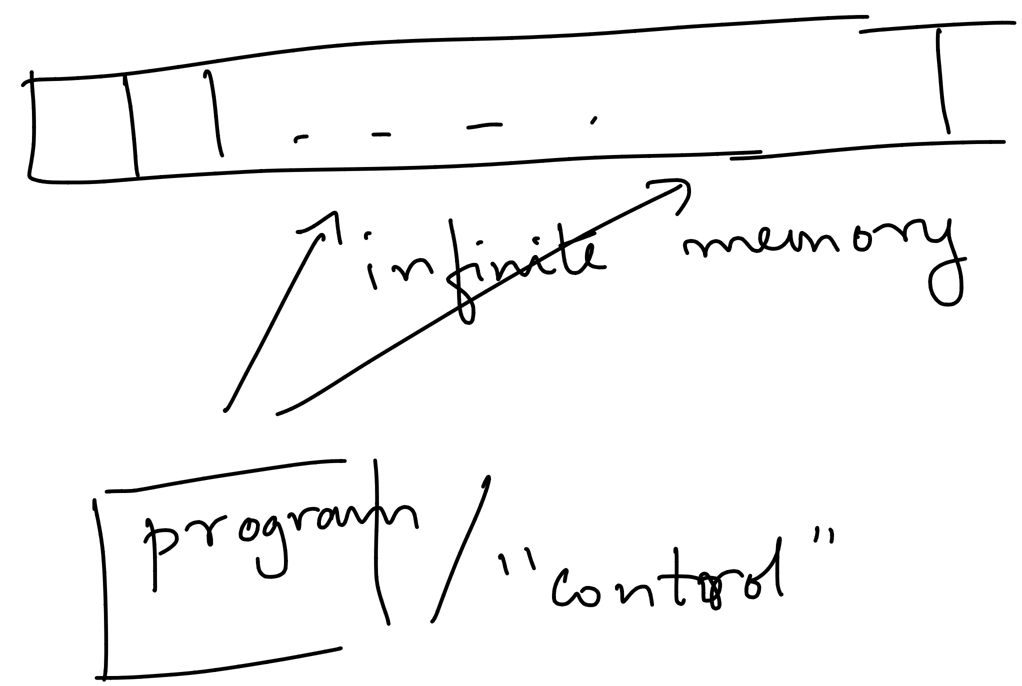
- **Key question:** is there a model that captures "all computations"

Candidate: There is no poly time algorithm for factorization.

Claim:

Universal models

- Turing machine
- Equiv., RAM model



... Quantum computers...

Church-Turing "thesis". any *reasonable* physical model of computation can be simulated "efficiently" (polynomial slowdown) on a Turing machine

Universal models

- Turing machine
- Equiv., RAM model

Church-Turing "thesis". any *reasonable* physical model of computation can be simulated "efficiently" (polynomial slowdown) on a Turing machine

Assuming it's true, need to only show lower bounds on a Turing machine

Reasoning about problems

[Can a # be factored efficiently?
Can we solve maximum Ind. set
in poly time?].

- "Simplification": move to **decision version**
- ☆ Does graph have an independent set of size *k*?
- Does graph have a path of length <= L between *u* and *v*?

problems
where
answers
is YES/NO.

Max-independent set:

G

– find the largest

set of vertices without any edges between them.

Decision vs optimization

Theorem. suppose we can solve the decision version of the independent set problem in poly time, we can actually solve max-IS in poly time

(useful because LOWER BOUNDS for decision problem

$\Updownarrow$

Lower bounds for optimization version.)

Decision vs optimization

Proof: – first find k, the size of max indep. set.

– ~~fst~~ pick any vertex u & consider $G \setminus u$

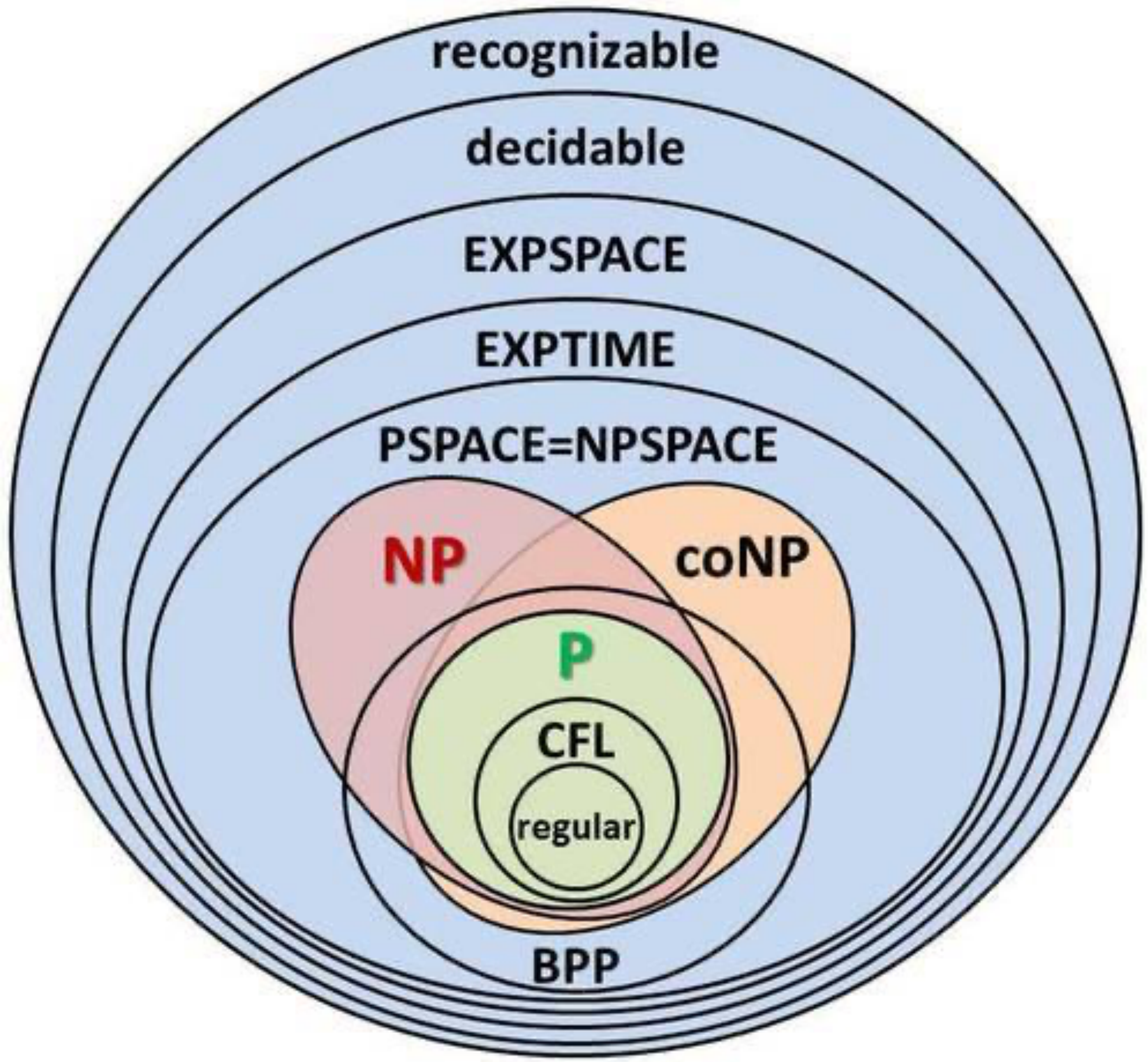
– if $G \setminus u$ has indep set of size $\geq k$, then recurse into $G \setminus u$.

– if not, add u to the IS and recurse into $G \setminus u$.

Complexity

Question. does the decision version of IS have a polynomial time algorithm?

Complexity "classes"



The class P

The class NP

Question. does the graph G have an independent set of size k?

The class NP

Most “puzzles” belong to NP

Boolean satisfiability

“Captures” the essence of NP