Lecture 5: Hardness within P

- "Fine-Grained Hardness"
- The Assumptions
- Hardness of Diameter

Resources

P, NP, and all that…

P = efficient?

The Model Matters

You studied Turing Machines in CS240

The RAM Model

All basic arithmetic/logical operations on numbers in $\underbrace{[1, n^{100}]}_{O(\log n) \text{ bits}}$ in 1 step.

Aside: We've already been working in the RAM Model

What's the runtime of Dinitz-Edmonds-Karp really?

Aside: We've already been working in the RAM Model

What's the runtime of Dinitz-Edmonds-Karp really?

We said $O(m^2 n)$. But what's the dependence on the size of the input numbers?

Real time complexity: $O(m^2 n) \cdot \tilde{O}(\underbrace{B}_{\text{\# bits in any input number}} + \log n)$

What do we do when we fail to find a fast algorithm?

e.g., longest common subsequence

Input: A B C D   length
A C B A D   n each

What do we do when we fail to find a fast algorithm?

e.g., longest common subsequence

Input: A B C D length n each
A C B A D

(AB) (AC) (AD) (BD)
(CD)
(ABD) (ACD).

dynamic programming: $O(n^2)$ time

Is there a better algorithm? $O(n^{1.99})$?

How can we show there *doesn't* exist a better algorithm

Dream: $P \neq NP \overset{?}{\underset{??}{\Rightarrow}}$ LCS not in time $n^{1.99}$

Backup Dream

Make one or two fixed assumptions

Derive many hardness consequences.

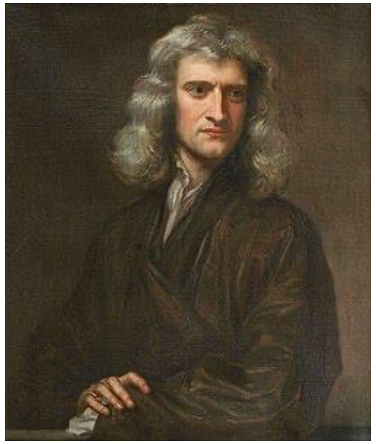
Backup Dream

Make one or two fixed assumptions

Derive <u>many</u> hardness consequences.

"We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances."

"Occam's Razor"



Backup Dream

The *SETH*

The 3-SUM Assumption

The APSP Assumption

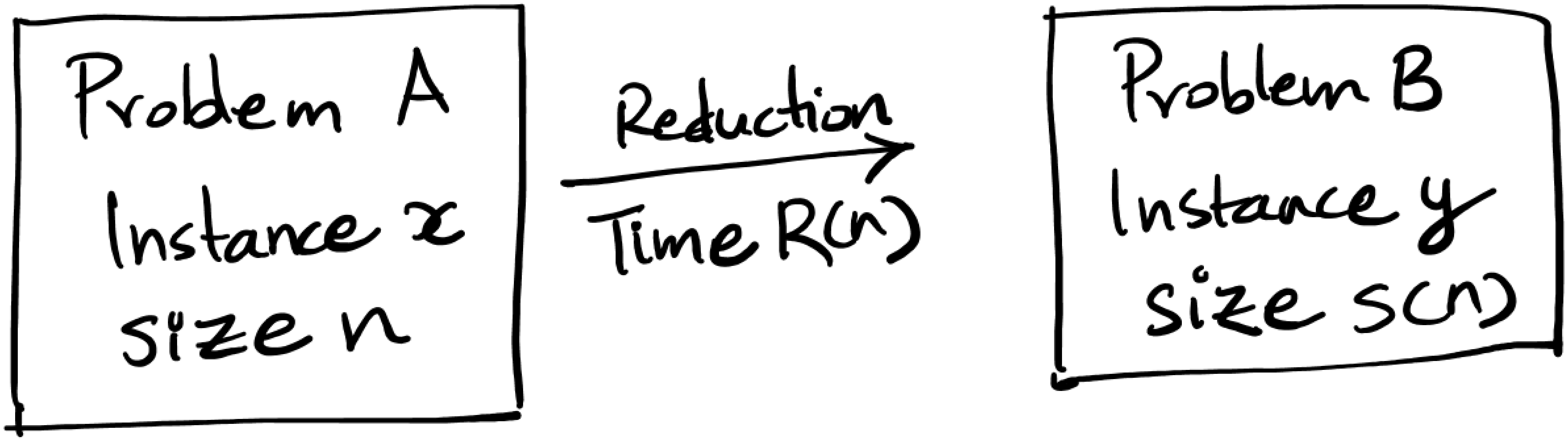
The k-Clique Assumption

Pioneers

FINE GRAINED COMPLEXITY



Reductions


Problem A
Instance x
size n
Reduction
Time R(n)
Problem B
Instance y
size s(n)

Reductions

Problem A
Instance x
size n

$\xrightarrow[\text{Time } R(n)]{\text{Reduction}}$

Problem B
Instance y
size $s(n)$

x is a YES instance of A $\iff$ y is a YES instance of B

Reductions

Problem A
Instance x
size n

Reduction
Time $R(n)$

Problem B
Instance y
size $s(n)$

x is a YES instance of A $\iff$ y is a YES instance of B

usually dominates

$T(n)$ time algo for B $\Rightarrow$ $T(s(n)) + R(n)$ algo for A.

Reductions

Prob A, size n $\xrightarrow[R(n) \text{ time}]{\text{reduction}}$ Prob B, size S(n)

x is a YES instance of A $\Leftrightarrow$ y is a YES instance of B

usually dominates

T(n) time algo for B $\Rightarrow$ T(S(n)) + R(n) algo for A.

no T(S(n)) + R(n) time algo for A $\Rightarrow$ No time T(n) time algo for B

Reductions

More generally, you could design an algo. that uses a subroutine for B to give an algorithm for A.

ETH, SETH

Input: A CNF-Formula (AND OF ORs) in n truth variables $(x_1 \vee \bar{x}_2 \vee x_{50} \vee \bar{x}_{21} \ldots) \wedge \ldots$

Goal: sat truth assignment?

ETH, SETH

Input: A CNF-Formula (AND OF ORs) in n truth variables $(x_1 \vee \bar{x}_2 \vee x_{50} \vee \bar{x}_{21} \ldots) \wedge \ldots$

Goal: sat truth assignment?

Brute Force: $O(2^n \cdot m)$. For 3SAT: best 1.31^n.

SETH: $\forall \varepsilon > 0$, CNF-SAT with m clauses & n variables requires $2^{(1-\varepsilon)n} \cdot \text{poly}(m)$ time

Hardness of Computing Diameter

Computing Diameter

<u>Input</u>: undirected graph G n vtcs, m edges

<u>Goal</u>: Compute diam(G)

$$\overset{\text{def}}{=} \max_{u,v} \text{dist}(u,v)$$

"length of a shortest path"

Computing Diameter

Computing Diameter

Input: undirected graph G — n vtcs, m edges

Goal: Compute $\mathrm{diam}(G) \overset{\text{def}}{=} \max_{u,v} \mathrm{dist}(u,v)$

"length of a shortest path"

Runtime: $O(mn)$

Computing Diameter

<u>OPEN Q:</u> FASTER THAN $O(mn)$ TIME?

Computing Diameter

OPEN Q: FASTER THAN $O(mn)$ TIME?

[ACIM'96], [Roddity-Vassilevska'13]

$\tilde{O}(m\sqrt{n})$ time algo. Factor $\frac{2}{3}$-approx.

Computing Diameter

OPEN Q: FASTER THAN $O(mn)$ TIME?

[ACIM'96], [Roddity-Vassilevska'13]

$\tilde{O}(m\sqrt{n})$ time algo. Factor $\frac{2}{3}$-approx.

Thm [Roddity-Vassilevska'13]

CNF-SETH $\Rightarrow$ No $O(MN^{1-\varepsilon})$ time exact algo

#edges

#vertices

SETH Implies no better algorithm for exact diameter

THM:

CNF-SETH $\Rightarrow$ No $O(MN^{1-\varepsilon})$ time exact algo

SETH Implies no better algorithm for exact diameter

THM:
CNF-SETH $\Rightarrow$ No $O(MN^{1-\varepsilon})$ time exact algo

PROOF: "reduction"

CNF-SAT $\xrightarrow[\text{reduction}]{\text{TIME: } 2^{\frac{n}{2}} \cdot \text{poly}(m)}$ GRAPH G

$\phi = (x_1 \vee x_{10} \vee \bar{x}_{50} \vee x_{20}) \wedge \ldots$

n vars, m clauses

$N = 2^{n/2} + m + 2$

$M = O(m) \cdot 2^{n/2}$

ϕ sat $\Rightarrow$ diam(G) = 3, ϕ unsat $\Rightarrow$ diam(G) = 2

SETH Implies no better algorithm for exact diameter

SETH Implies no better algorithm for exact diameter

THM: CNF-SETH $\Rightarrow$ No $O(MN^{1-\varepsilon})$ time exact algo

PROOF: CNF-SAT $\xrightarrow{\text{TIME: } 2^{\frac{n}{2}}\cdot\text{poly}(m) \text{ reduction}}$ GRAPH G

n vars, m clauses

$N = 2\cdot 2^{n/2} + m$

$M = O(m)\cdot 2^{n/2}$

Φ sat $\Rightarrow$ diam(G) = 3,

Φ unsat $\Rightarrow$ diam(G) = 3

$O(MN^{1-\varepsilon})$ time algo

$\Rightarrow$ CNFSAT in time $2^{n/2}\cdot\text{poly}(m) + \text{poly}(m)\cdot 2^{n/2}\cdot 2^{\frac{n}{2}(1-\varepsilon)}$

Vertices: "Clause vertices": $C_1, C_2, \ldots, C_m$

"partial assignment vertices": $\alpha \in \{0,1\}^{n/2}$

$\beta \in \{0,1\}^{n/2}$

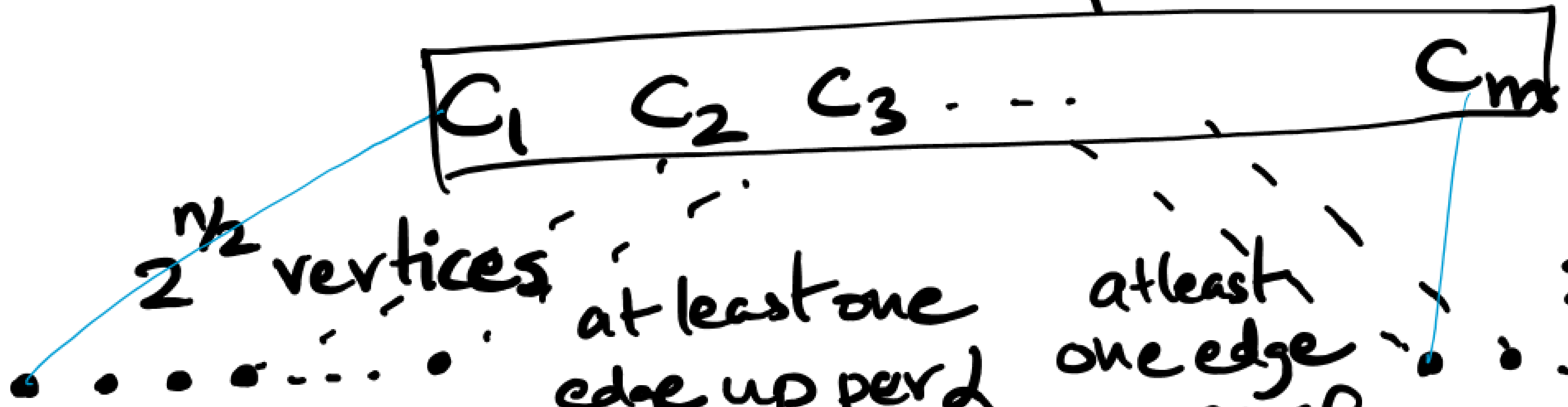
Edges: all possible edges betⁿ C_is.

no edge betⁿ any α & any β.

Each α, β will have an edge to some C_i except in a corner case

If some α has no edges $\rightarrow$ output any graph of diam 2

add a clique on C_is. (connect all pairs)



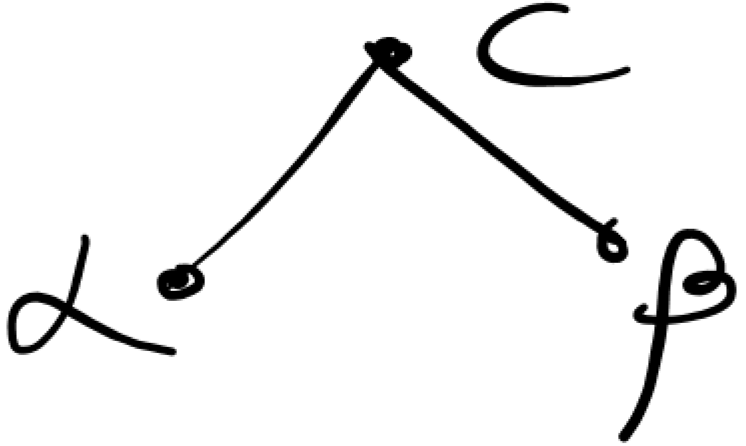
one vertex for each truth assignment α to $x_1, \dots, x_{\underline{n}}$

Connect α to C_i iff C_i is NOT sat by α (same for β)

one vertex for each partial assignment β to $x_{\frac{n}{2}+1}, \dots, x_n$

(α, C) edge: Connect α & C if α does <u>NOT</u> satisfy C. That is α does not set some literal in C to true

(β, C) edge: Connect β & C if β does <u>NOT</u> satisfy C.



Fix α. There is no (α, C) edge for all C iff α satisfies every constraint

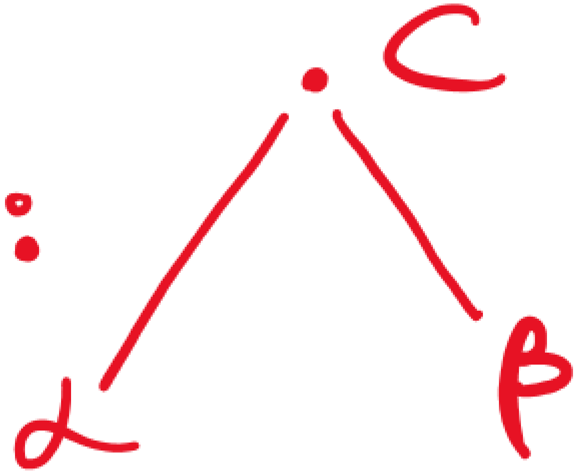
In that case we've discovered that the input formula is sat. In this corner case, output any graph with diam 3. (Similarly for β).

So far, time $\approx 2^{n/2} \left(\text{poly}(m) \cdot 2^{n/2} \right)$

$\#$ vertices, $\#$ edges ✓

diam(G) ≤ 3

It will be 2 iff $\forall\, \alpha, \beta, \exists\, c$:



Analysis

If Φ is sat, $diam(G) = 3$. If not $diam = 2$.

Analysis

If Φ is sat, $\text{diam}(G)=3$. If not $\text{diam}=2$.

$\boxed{\text{Key:}}$ (α,β) have dist 3 $\Leftrightarrow$ $\forall C_i$ don't have 

$\Leftrightarrow$ either α or β sat C_i $\forall i$

$\Leftrightarrow$ Combined assignment $\alpha+\beta$ sat C_i $\forall i$

$\Leftrightarrow$ Combined assignment (α,β) sat Φ.

So. Φ is sat $\Leftrightarrow$ $\exists(\alpha,\beta)$ at distance 3 $\Leftrightarrow$ $\text{diam}(G)=3$