CS 331, Fall 2025
Lecture 25 (12/3)

Today: - Beyond time complexity
- Exponential-time algorithms
- Approximation algorithms

# Beyond time complexity (Part VIII, Section 5.3)

We've defined many time-based classes:

P, NP, coNP, NP-hard, NP-complete
TIME(f(n)), NP-intermediate, PH

What other resources are interesting to measure?

- Space          - randomness          (today)

- parallelism, communication, samples, ...

# Space

$SPACE(f(n))$: All languages $L$ decidable using $f(n)$ space (disregard time)

Fact: $TIME(f(n)) \subseteq SPACE(f(n)) \subseteq TIME(2^{f(n)})$

Can't write enough bits

Should avoid looping state

Main interesting classes: $L = SPACE(O(\log(n)))$

$PSPACE = SPACE(poly(n))$

Containment:

$$L \underset{?}{\subseteq} NL = coNL \underset{?}{\subseteq} P \underset{?}{\subseteq} PH \underset{?}{\subseteq} PSPACE$$

Immerman-Szelepcsenyi

e.g. $NP$, $coNP$, ...

unequal, space hierarchy theorem

# Randomness

BPP : decidable w.p. $\gtrsim \frac{2}{3}$ using

    —poly$(n)$ time    —poly$(n)$ coin tosses

Big question: is BPP = P?

What we know... 
- BPP $\subseteq \Sigma_2 P \cap \Pi_2 P$

              (polynomial hierarchy)

- If $\exists L \in TIME(\exp(O(n)))$

      that requires <u>circuits</u> of $\exp(\Omega(n))$

      size, then BPP = P

Experts @ UT: David Zuckerman, Dana Moshkovitz

BQP : Computational model can use quantum

Experts @ UT: Scott Aaronson, Will Kretschmer, Nick Hunter-Jones

Rest of today: coping with NP-hardness

Sometimes, you just have to do it.

Option 1: Prove $P = NP$

Option 2: Sacrifice <u>runtime</u>

Option 3: Sacrifice <u>correctness</u>     } today

Some problems today... prove they're NP-complete

MinMakespan: Given list of job lengths, $k \in \mathbb{N}$
how to assign jobs to $k$ machines
w/ min {largest total length on machine}

Max3SAT: Given 3CNF formula $\Phi$,
how many clauses can we satisfy?

TSP: Compute Ham Cycle of min weight
(stay tuned...)

# Exponential-time algorithms (Part VIII, Section 6)

3SAT: $m$ clauses, $n$ Boolean variables

e.g. $\left(x_1 \vee x_3 \vee \neg x_4\right) \underset{\text{AND}}{\wedge} \left(x_2 \vee \neg x_3 \vee x_5\right)\ldots$

$\quad\quad\quad\quad\underset{\text{OR}}{}$

Naive: $O(2^n m)$

Idea 1: Divide-and-conquer

Recurse:

- Find any unsatisfied clause $C$
- Try all 7 assignments to $C$
- Eliminate all satisfied clauses

$$T(n) \leq 7\, T(n-3) + O(m)$$

(an check: $T(n) = O\left(\left(7^{1/3}\right)^n m\right)$

$$\underbrace{\quad\quad}_{\simeq 1.913^n}$$

Idea 2: Divide-and-conquer, redux

let $C = \underbrace{l_1 \vee l_2 \vee l_3}_{\text{literals}}$ be satisfied

Case 1: $l_1 = $ TRUE

2: $l_1 = $ FALSE, $l_2 = $ TRUE

3: $l_1 = l_2 = $ FALSE, $l_3 = $ TRUE

$$T(n) \leq T(n-1) + T(n-2) + T(n-3) + O(m)$$

Works out to $T(n) = O(1.84^n m)$

$$\nearrow$$

root of $r^3 = r^2 + r + 1$

SOTA (theory):

3SAT in $\approx 1.31^n \cdot m$ time      "PPSZ algo"
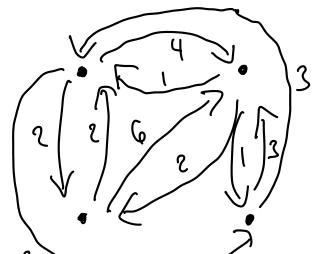
kSAT in $\approx 2^{(1-O(\frac{1}{k}))n} \cdot m$ time

SETH: For large constant $k$, kSAT
"strong ETH"    not soluble in time $\ll 2^n$

SOTA (practice):

SAT solvers work reasonably well. "CDCL"

TSP: Input complete weighted directed $G$
                   $> 0$



What is least-weight HamCycle?

• Visit every vertex once

• Return to start

Naive algo: try every permutation (cycle)

$\approx n!$ time. $\ddot{\frown}$

Bellman-Held-Karp: $\approx 2^n$ time via DP.

e.g. $30! = 2.65 \times 10^{32}$

$2^{30} = 1.07 \times 10^9$

Key Claim: let $C_{S,ij} = $ min cost of

cycle-free path that starts @ 1, ends @ j,

uses exactly intermediate vertices $S \subseteq V$

Then, $C_{S,ij} = \min_{k \in S} C_{S\setminus\{k\}, k} + W_{(k,j)}$

Idea: exponential-sized DP

$$DP[S][j] := C_{Sij}$$

$\le 2^n \, n$ subprobs $\times \, O(n)$ time / subprob

If we know all $C_{Sij}$, just try all

$S = V \setminus \{1, j\}$ to find optimal



total cost: $C_{Sij} + W_{(j,1)}$

# Approximation algorithms (Part VIII, Section 7)

Usually our goal is to compute

$$OPT = \min_{(or\ max)} (\text{Some problem, over choices})$$

If the problem is too hard, maybe we're ok w/ a solution between $OPT$ and $C \cdot OPT$.

↳ approximation factor, hopefully small...

## Min Makespan $(\{1, 2, 3, 4, 12, 13\}, 3)$

$k=3$ machines



makespan = max machine load

Idea: greedy assignment

For job $i \in [n]$:

    Assign job $i$ to machine $j \in [k]$
    with current smallest load. (arbitrary tiebreak)

Quiz: what is runtime?

Naive: $O(nk)$. Better: $O(n \log(k))$ (heap)

> Greedy achieves $C = 2$-approx!

Let $OPT = $ min makespan

$l_1, l_2, \ldots l_n$ be lengths

Claim 1: $\forall i \in (n)$, $OPT \geq l_i$

Claim 2: $OPT \geq \frac{1}{k} \sum_{i \in (n)} l_i$

Suppose greedy produces max load ALG



1

2

3

$l_{i^*}$

Most loaded machine

$L_{before}$: Smallest @ time $n-1$.

$ALG = L_{before} + l_{i^*}$   (Claim 2)   (Claim 1)

$$\leq \frac{1}{k} \sum_{i \in (n)} l_i + l_{i^*} \leq OPT + OPT$$

$$\leq 2 \cdot OPT \quad \blacksquare \quad (\text{in fact } C = 1.5$$
$$\text{w/ more careful analysis...)}$$

# Max 3SAT

try to satisfy as many clauses

$$\Phi = \bigwedge_{i \in (m)} \phi_i, \quad \phi_i = \ell_{i_1} \lor \ell_{i_2} \lor \ell_{i_3}$$

Idea: randomness. Each $\phi_i$ satisfied w.p. $\frac{7}{8}$

Pick uniformly random assignment

$$\mathbb{E}\left[ \# \text{ Clauses satisfied} \right]$$

$$= \mathbb{E}\left[ \sum_{i \in (m)} \mathbb{1}(\phi_i \text{ satisfied}) \right]$$

$$\underset{(\text{lin. ex.})}{=} \sum_{i \in (m)} \underbrace{\mathbb{E}\left[ \mathbb{1}(\phi_i \text{ satisfied}) \right]}_{= \Pr[\phi_i \text{ satisfied}] = \frac{7}{8}} = \frac{7}{8} m$$

We have randomized $C = \frac{8}{7}$ - approximation

Can be made deterministic in time $O(mn)$

Proof sketch: Assign one @ a time, best $\mathbb{E}$.

Suppose $X_1, \ldots X_k$ assigned

$$\mathbb{E}_{\substack{X_{k+1} \\ \vdots \\ X_n}} \left[ \# \text{ clauses satisfied} \mid X_1, \ldots X_k \right]$$

$$= \frac{1}{2} \mathbb{E}_{\substack{X_{k+2} \\ \vdots \\ X_n}} \left[ \# \text{ clauses satisfied} \mid \begin{matrix} X_1, \ldots X_k \\ X_{k+1} = \text{TRUE} \end{matrix} \right]$$

$$+ \frac{1}{2} \mathbb{E}_{\substack{X_{k+2} \\ \vdots \\ X_n}} \left[ \# \text{ clauses satisfied} \mid \begin{matrix} X_1, \ldots X_k \\ X_{k+1} = \text{FALSE} \end{matrix} \right]$$

We can determine which is bigger. (lin. ex. again)

Fun fact:

It's NP-hard to get $\left(\frac{8}{7} - o(1)\right)$-approx.

for Max3SAT (Moshkovitz-Raz '08)

Based on smarter reduction: (Fractan)

Normally NP-hard to distinguish m | $\qquad$

vs. m-1. $1 - \frac{1}{m}$

Using __amplification__ can boost $1 - \frac{1}{m} \Rightarrow$ smaller

at same cost in problem size.

Turns out to be equivalent to "PCP theorem"

Probabilistically checkable proof: randomly examine

$O(1)$ bits of witness string $w$