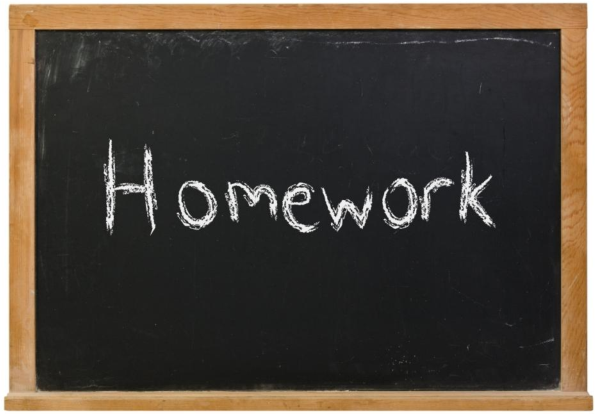
CSE 421: Introduction to Algorithms

Course Overview

Shayan Oveis Gharan

Administrativia Stuffs



HW1 is due Thursday April 09 at 11:59PM
Please submit to Canvas

Late Submission: Coordinate with me
How to submit?

- Submit a separate file for each problem
- Double check your submission before the deadline!!
- For hand written solutions, take a picture, turn it into pdf and submit

Guidelines:

- Always justify your answer
- You can collaborate, but you must write solutions on your own
- Your proofs should be clear, well-organized, and concise. Spell out main idea.
- Sanity Check: Make sure you use assumptions of the problem

Man Optimality Summary

Man-optimality: In version of GS where men propose, each man receives the best valid partner.

w is a valid partner of m if there exist some stable matching where m and w are paired

Q: Does man-optimality come at the expense of the women?

Example

Here

Valid-partner(w_1) $= \{m_1, m_2\}$

Valid-partner(w_2) $= \{m_1, m_2\}$

Valid-partner(w_3) $= \{m_3\}$.

Man-optimal matching $\{m_1, w_1\}, \{m_2, w_2\}, \{m_3, w_3\}$

	1st (favorite)	2nd	3rd (least favorite)
m_1	w_1	w_2	w_3
m_2	w_2	w_1	w_3
m_3	w_1	w_2	w_3

	1st (favorite)	2nd	3rd (least favorite)
w_1	m_2	m_1	m_3
w_2	m_1	m_2	m_3
w_3	m_1	m_2	m_3

Woman Pessimality

Woman-pessimal assignment: Each woman receives the worst valid partner.

Claim. GS finds woman-pessimal stable matching **S***.

Proof.

Suppose (m, w) matched in **S***, but m is not worst valid partner for w.

There exists stable matching **S** in which w is paired with a man, say m', whom she likes less than m.

Let w' be m partner in **S**.

m prefers w to w'. ← man-optimality of **S***

Thus, (m, w) is an unstable in **S**.

■

Summary

- Stable matching problem: Given **n** men and **n** women, and their preferences, find a stable matching if one exists.

- Gale-Shapley algorithm guarantees to find a stable matching for any problem instance.

- GS algorithm finds a stable matching in $O(n^2)$ time. ✓

- GS algorithm finds man-optimal woman pessimal matching ✓

- Q: How many stable matching are there?

How many stable Matchings?

We already show every instance has at least 1 stable matchings.

There are instances with about b^n stable matchings for $b > 2$

[Karlin-O-Weber'17]: Every instance has at most c^n stable matchings for some $c > 2$

[Open Problem]:

Is there an "efficient" algorithm that chooses a uniformly random stable matching of a given instance.

Extensions: Matching Residents to Hospitals

Men ≈ hospitals, Women ≈ med school residents.

- Variant 1: Some participants declare others as unacceptable.
- Variant 2: Unequal number of men and women.
- Variant 3: Limited polygamy.

e.g. A resident not interested in Cleveland

e.g. A hospital wants to hire **3** residents

Def: Matching **S** is unstable if there is hospital **h** and resident **r** s.t.

- **h** and **r** are acceptable to each other; and
- either **r** is unmatched, or **r** prefers **h** to her assigned hospital; and
- either **h** does not have all its places filled, or **h** prefers **r** to at least one of its assigned residents.

Lessons Learned

- Powerful ideas learned in course.
  - Isolate underlying structure of problem.
  - Look at the first occurrence of a bad event and get a contradiction.

- Potentially deep social ramifications. [legal disclaimer]
  - Historically, men propose to women. Why not vice versa?
  - Men: propose early and often.
  - Men: be more honest.
  - Women: ask out the guys.
  - Theory can be socially enriching and fun!

“The Match”: Doctors and Medical Residences

- Each medical school graduate submits a ranked list of hospital where he wants to do a residency
- Each hospital submits a ranked list of newly minted doctors
- A computer runs stable matching algorithm (extended to handle polygamy)
- Until recently, it was hospital-optimal.

History

1900

- Idea of hospital having residents (then called “interns”)

1900-1940s

- Intense competition among hospitals
  - Each hospital makes offers independently
  - Process degenerates into a race; hospitals advancing date at which they finalize binding contracts

1944

- Medical schools stop releasing info about students before a fixed date

1945-1949

- Hospitals started putting time limits on offers
  - Time limits down to 12 hours; lots of unhappy people

“The Match”

1950

- NICI run a centralized algorithm for a trial run
- The pairing was not stable, Oops!!

1952

- The algorithm was modified and adopted. It was called the Match.
- The first matching produced in April 1952

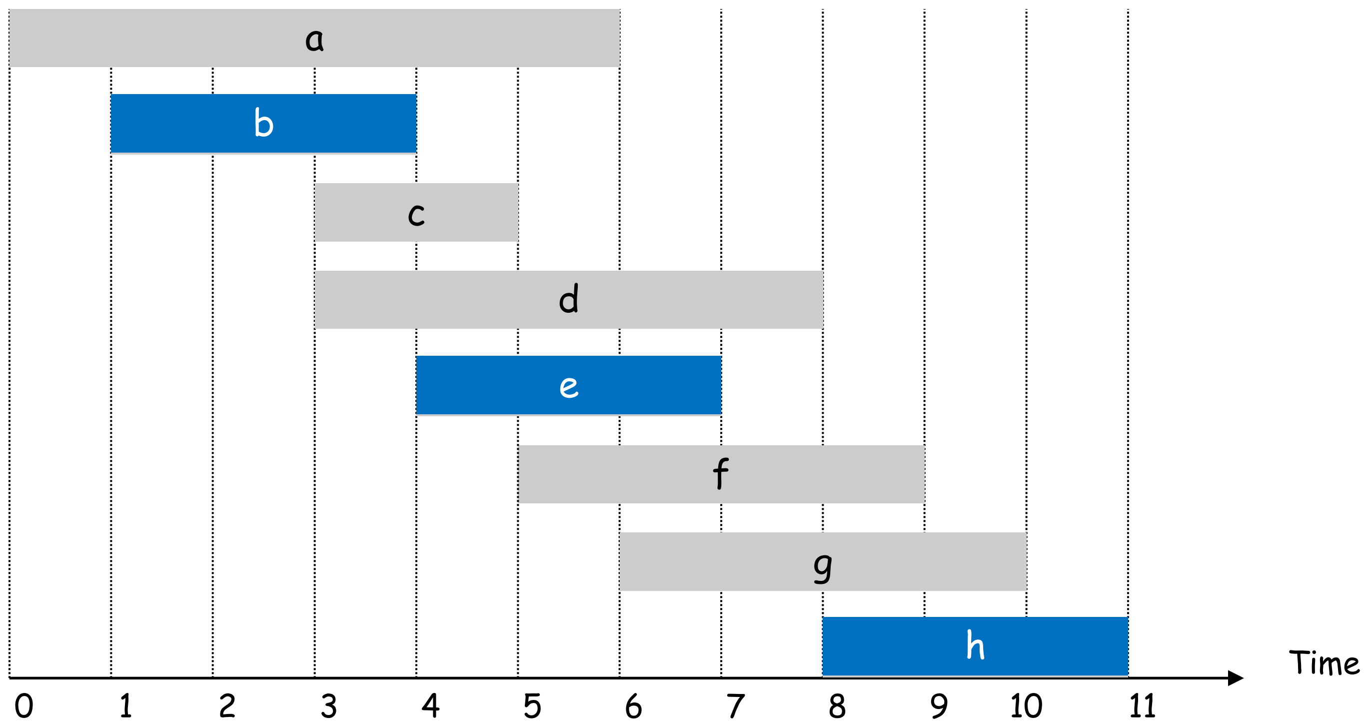
Five Representative Problems

1. Interval Scheduling
2. Weighted Interval Scheduling
3. Bipartite Matching
4. Independent Set Problem
5. Competitive Facility Location

Interval Scheduling

Input: Given a set of jobs with start/finish times

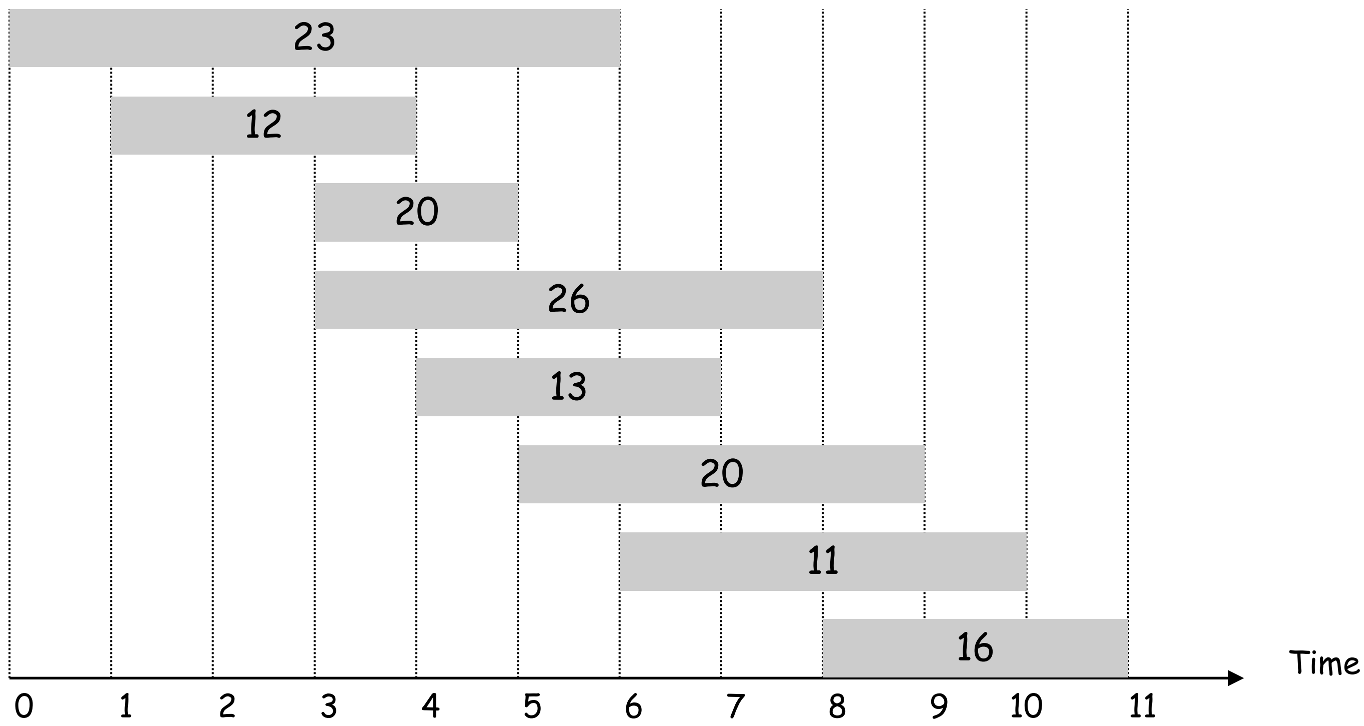
Goal: Find the maximum cardinality subset of jobs that can be run on a single machine.



Interval Scheduling

Input: Given a set of jobs with start/finish times

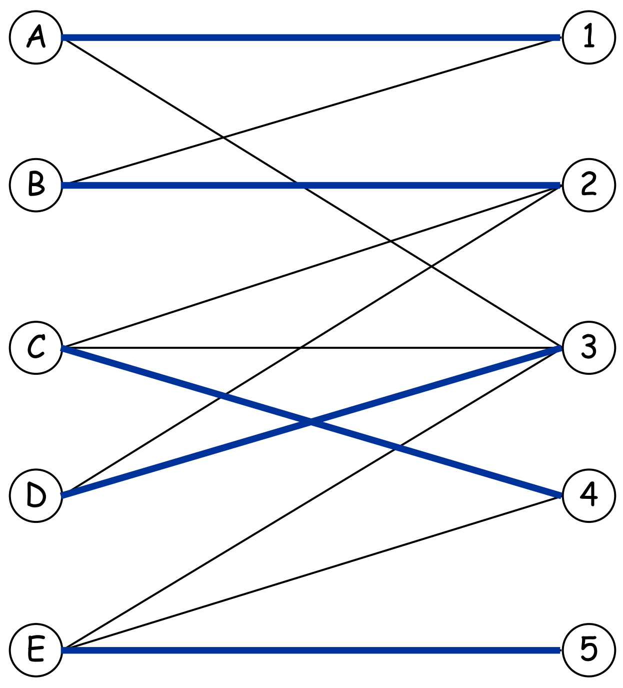
Goal: Find the maximum weight subset of jobs that can be run on a single machine.



Bipartite Matching

Input: Given a bipartite graph

Goal: Find the maximum cardinality matching

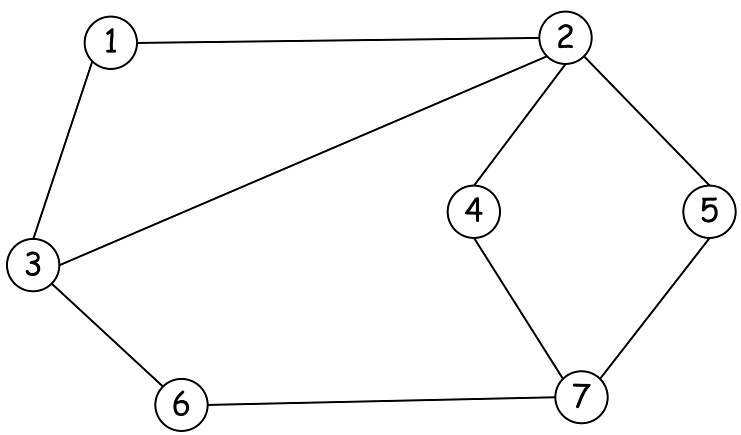


Independent Set

Input: A graph

Goal: Find the maximum independent set

Subset of nodes that no two joined by an edge

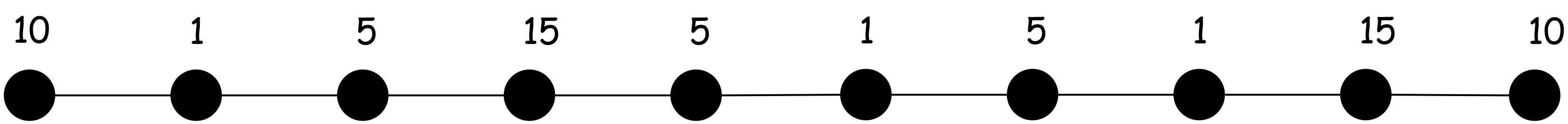


Competitive Facility Location

Input: Graph with weight on each node

Game: Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbors have been selected.

Goal. Does player 2 have a strategy which guarantees a total value of V no matter what player 1 does?



Second player can guarantee 20, but not 25.

Five Representative Problems

Variation of a theme: Independent set Problem

1. Interval Scheduling
   $n \log n$ greedy algorithm

2. Weighted Interval Scheduling
   $n \log n$ dynamic programming algorithm

3. Bipartite Matching
   n^k maximum flow based algorithm

4. Independent Set Problem: NP-complete

5. Competitive Facility Location: PSPACE-complete

Main Objective: Design Efficient Algorithms
that finds optimum solutions in the Worst Case

Defining Efficiency

"Runs fast on typical real problem instances"

Pros:

- Sensible,
- Bottom-line oriented

Cons:

- Moving target (diff computers, programming languages)
- Highly subjective (how fast is "fast"? What is "typical"?)

Measuring Efficiency

Time ≈ # of instructions executed in a simple programming language

- only simple operations (+,*,-,=,if,call,…)
- each operation takes one time step
- each memory access takes one time step
- no fancy stuff (add these two matrices, copy this long string,…) built in; write it/charge for it as above

Time Complexity

Problem: An algorithm can have different running time on different inputs

Solution: The complexity of an algorithm associates a number **T(N)**, the "time" the algorithm takes on problem size **N**.

On which inputs of size **N**?

Mathematically,

T is a function that maps positive integers giving problem size to positive integers giving number of steps

Time Complexity (N)

Worst Case Complexity: max # steps algorithm takes on any input of size **N**

This Couse

Average Case Complexity: avg # steps algorithm takes on inputs of size **N**

Best Case Complexity: min # steps algorithm takes on any input of size **N**

Why Worst-case Inputs?

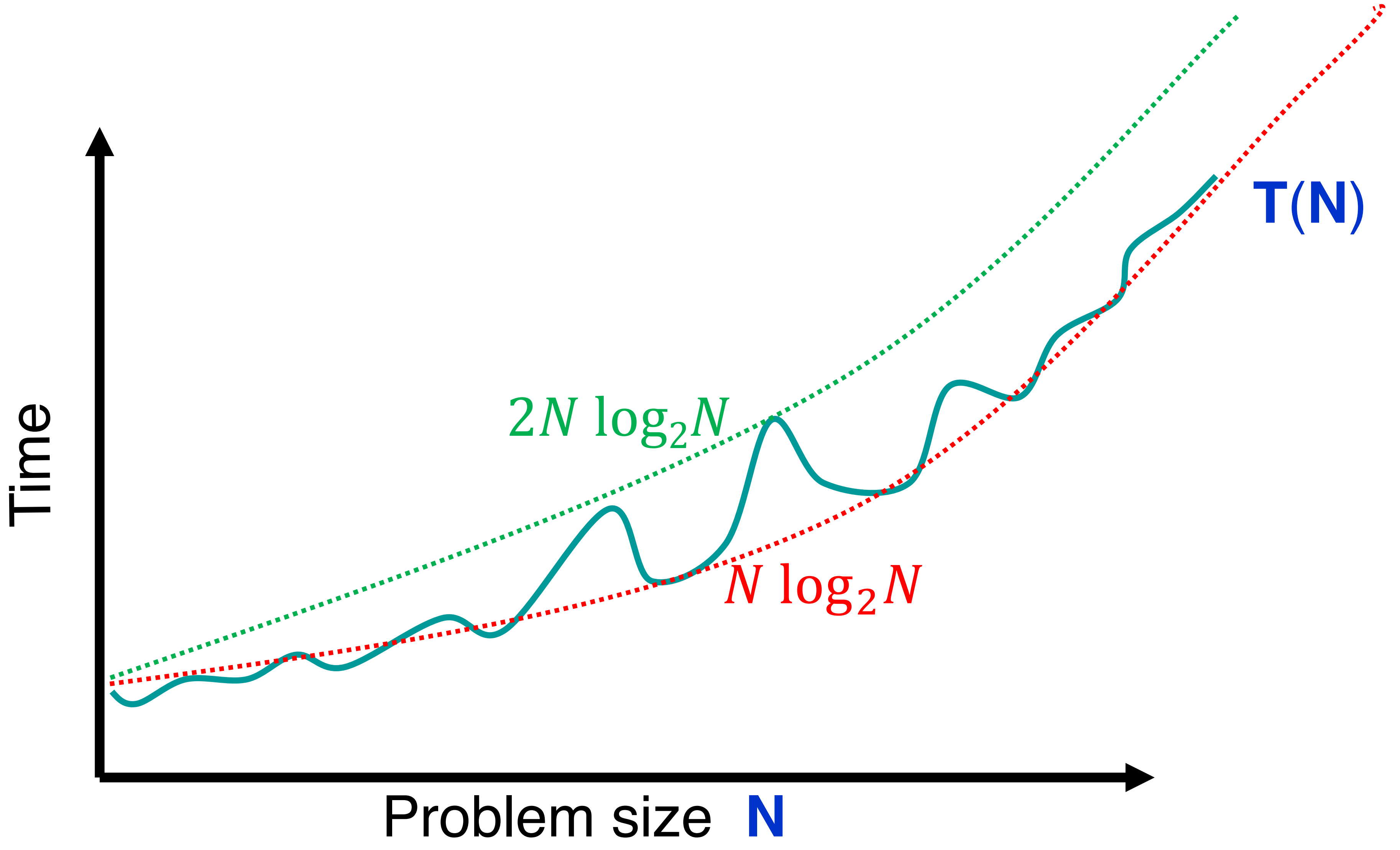
- Analysis is typically easier
- Useful in real-time applications

  e.g., space shuttle, nuclear reactors)
- Worst-case instances kick in when an algorithm is run as a module many times

  e.g., geometry or linear algebra library
- Useful when running competitions

  e.g., airline prices
- Unlike average-case no debate about the right definition

Time Complexity on Worst Case Inputs



O-Notation

Given two positive functions **f** and **g**

- **f(N)** is **O(g(N))** iff there is a constant **c**>**0** s.t.,
  f(N) is eventually always $\leq$ **c g(N)**

- **f(N)** is **Ω(g(N))** iff there is a constant **ε**>**0** s.t.,
  f(N) is $\geq$ **ε g(N)** for infinitely

- **f(N)** is **Θ(g(N))** iff there are constants c_1, c_2>0 so that
  eventually always $c_1 g(N) \leq f(N) \leq c_2 g(N)$

Asymptotic Bounds for common fns

- Polynomials:

  $a_0 + a_1 n + \cdots + a_d n^d$ is $O\left(n^d\right)$

- Logarithms:

  $\log_a n = O(\log_b n)$ for all constants $a, b > 0$

- Logarithms: log grows slower than every polynomial

  For all $x > 0$, $\log n = O(n^k)$

- $n \log n = O(n^{1.01})$

Efficient = Polynomial Time

An algorithm runs in polynomial time if $T(n)=O(n^d)$ for some constant d independent of the input size n.

Why Polynomial time?

If problem size grows by at most a constant factor then so does the running time

- E.g. $\mathbf{T(2N) \leq c(2N)^k \leq 2^k(cN^k)}$
- Polynomial-time is exactly the set of running times that have this property

Typical running times are small degree polynomials, mostly less than $\mathbf{N^3}$, at worst $\mathbf{N^6}$, not $\mathbf{N^{100}}$

Why it matters?

- #atoms in universe < 2^{240}
- Life of the universe < 2^{54} seconds
- A CPU does < 2^{30} operations a second

If every atom is a CPU, a 2^n time ALG cannot solve n=350 if we start at Big-Bang.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

not only get very big, but do so *abruptly*, which likely yields errati
performance on small instances

Why “Polynomial”?

Point is not that n^{2000} is a practical bound, or that the differences among n and 2n and n^2 are negligible.

Rather, simple theoretical tools may not easily capture such differences, whereas exponentials are qualitatively different from polynomials, so more amenable to theoretical analysis.

- “My problem is in P” is a starting point for a more detailed analysis
- “My problem is not in P” may suggest that you need to shift to a more tractable variant