Computing Theory

COMP 147 (4 units)

Chapter 7: Time Complexity
Section 7.1: Measuring Complexity
Section 7.2: Class P

Course Segments

- Automata and Languages
  - How can we define abstract models of computers?
- Computability Theory
  - What can (or cannot) be computed?
- Complexity Theory
  - What makes some problems computationally difficult?

Complexity Theory

- Consider only decidable problems
    - Goal: What is the time and space complexity?

Time Complexity of TMs

- "Time Complexity" = Running Time of programs
- Consider input w, count the number of steps of TM
  - Only decidable problems (always halt!)
- Consider all inputs
    - What is the maximum number of steps of TM
    - Expressed as a function f(n) where n is the size of the input

Time Complexity of TMs

DEFINITION 7.1

Let M be a deterministic Turing machine that halts on all inputs. The ***running time*** or ***time complexity*** of M is the function $f: \mathcal{N} \longrightarrow \mathcal{N}$, where $f(n)$ is the maximum number of steps that M uses on any input of length n. If $f(n)$ is the running time of M, we say that M runs in time $f(n)$ and that M is an $f(n)$ time Turing machine. Customarily we use n to represent the length of the input.

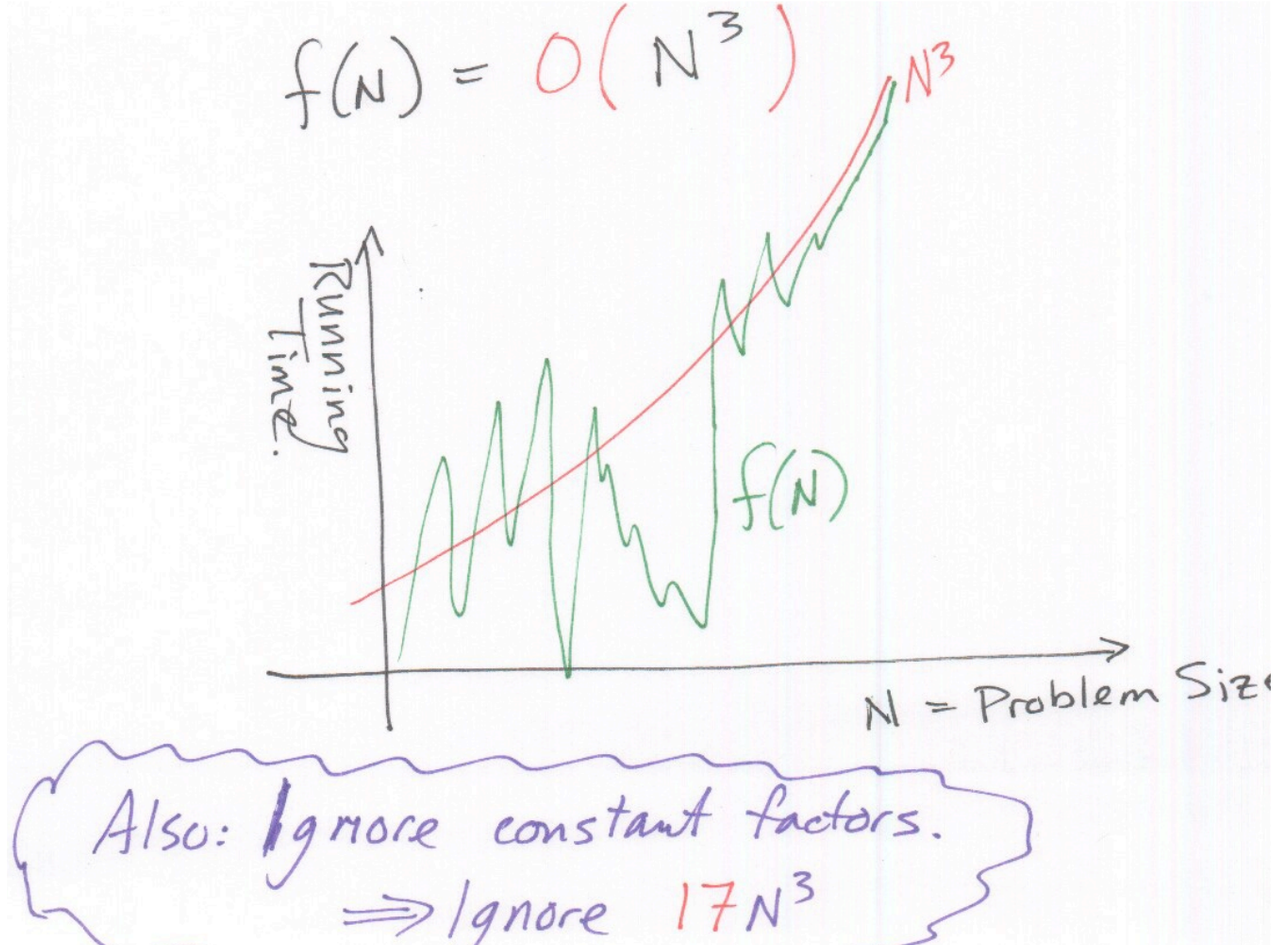
exact running time can be complex. Use Big-O Notation

Big-O Notation

- Function f(n) can be a complex expression

$$f(N) = 17N^3 + 5N^2 + 3\log N + 29$$

- For large value of N we only care about N^3



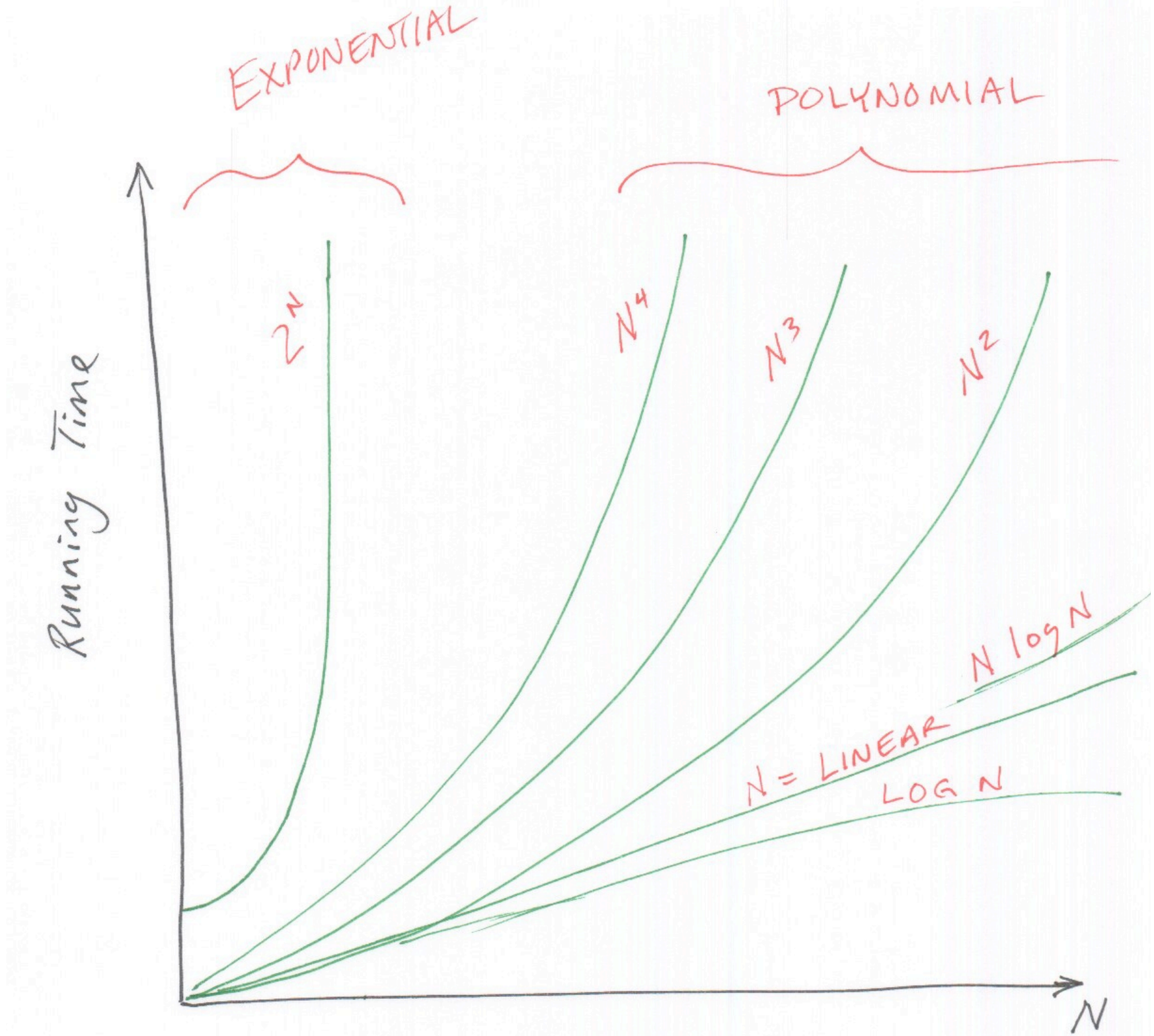
Big-O Notation

Number of operations	Big-O notation
n^2+5n	$O(n^2)$
$100n+n^2$	$O(n^2)$
$(n+7)(n-2)$	$O(n^2)$
$n+100$	$O(n)$
number of digits in $2n$	$O(\log n)$
$16n^3\log_8(10n^2) + 100n^2 + 1000$	$O(n^3 \log n)$

Time Complexity Classes

Function	Common Name
N!	factorial
2^N	Exponential
N^d, d > 3	Polynomial
N^3	Cubic
N^2	Quadratic
N log N	Log linear
N	Linear
$\sqrt{N}$	Root - n
log N	Logarithmic
1	Constant

Polynomial Time


EXPONENTIAL
POLYNOMIAL
Time
Running
2^N
N^4
N^3
N^2
N log N
N = LINEAR
LOG N
N

Running Times

- Assume N = 100,000 and processor with 10^6 operations/second

Function	Running Time
2^N	over 100 years
N^3	31.7 years
N^2	2.8 hours
$N\sqrt{N}$	31.6 seconds
N log N	1.2 seconds
N	0.1 seconds
$\sqrt{N}$	3.2×10^{-4} seconds
log N	1.2×10^{-5} seconds

Q: Why aren't there many $O(\log N)$ algorithms?

A: The input has size n.

Just to read all the input requires $O(n)$

Time Complexity Classes

TIME(n)

The set of all languages/problems that can be DECIDED in $O(n)$ time.

$TIME(n^2)$ ... that can be DECIDED in $O(n^2)$ time.

$TIME(n^3)$ ... in $O(n^3)$

$TIME(2^N)$ ... in exponential time.

NOTE:

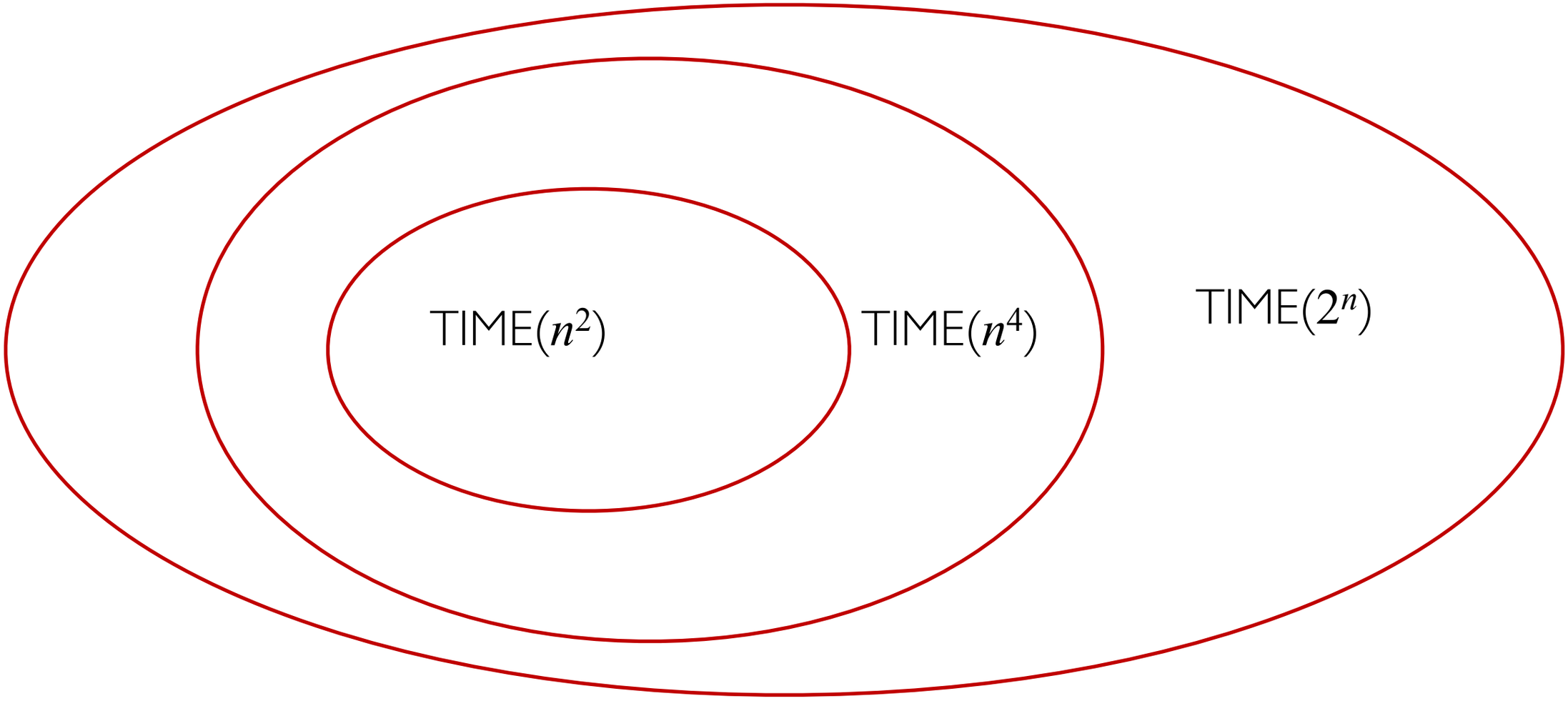
$TIME(n) \subset TIME(n \log n) \subset TIME(n^2) \subset$

$TIME(n^3) \subset TIME(n^K) \subset TIME(2^N)$

slide from Harry Porter

Time Complexity Classes

- TIME($t(n)$) is the set of all languages that are decidable by a TM whose time complexity is $O(t(n))$



Analyzing Algorithms

ALGORITHM TO DECIDE $\{0^k 1^k \mid k \geq 0\}$

M_1 = "On input string w:

1. Scan across the tape and *reject* if a 0 is found to the right of a 1.
2. Repeat if both 0s and 1s remain on the tape:
3. Scan across the tape, crossing off a single 0 and a single 1.
4. If 0s still remain after all the 1s have been crossed off, or if 1s still remain after all the 0s have been crossed off, *reject*. Otherwise, if neither 0s nor 1s remain on the tape, *accept*."

Algorithm to Decide $\{0^k 1^k \mid k \geq 0\}$

- Scan input to make sure it is in the form $0^* 1^*$.
  - n steps to scan
  - n steps to reposition to left end
  - $2n$ steps $O(n)$
- Repeat while tape contains at least one 0 and at least one 1...
  - Scan across tape and change a 0 to X and a 1 to X.
    - $2n$ steps $O(n)$
- End loop
  - $n/2$ repetitions
  - Whole loop take $\frac{N}{2} \cdot O(n)$ $O(n^2)$
- If tape contains all Xs, then accept else reject.
  - n steps $O(n)$

$O(n) + O(n^2) + O(n) \Longrightarrow O(n^2)$

slide from Harry Porter

Algorithm to Decide $\{0^k 1^k \mid k \geq 0\}$

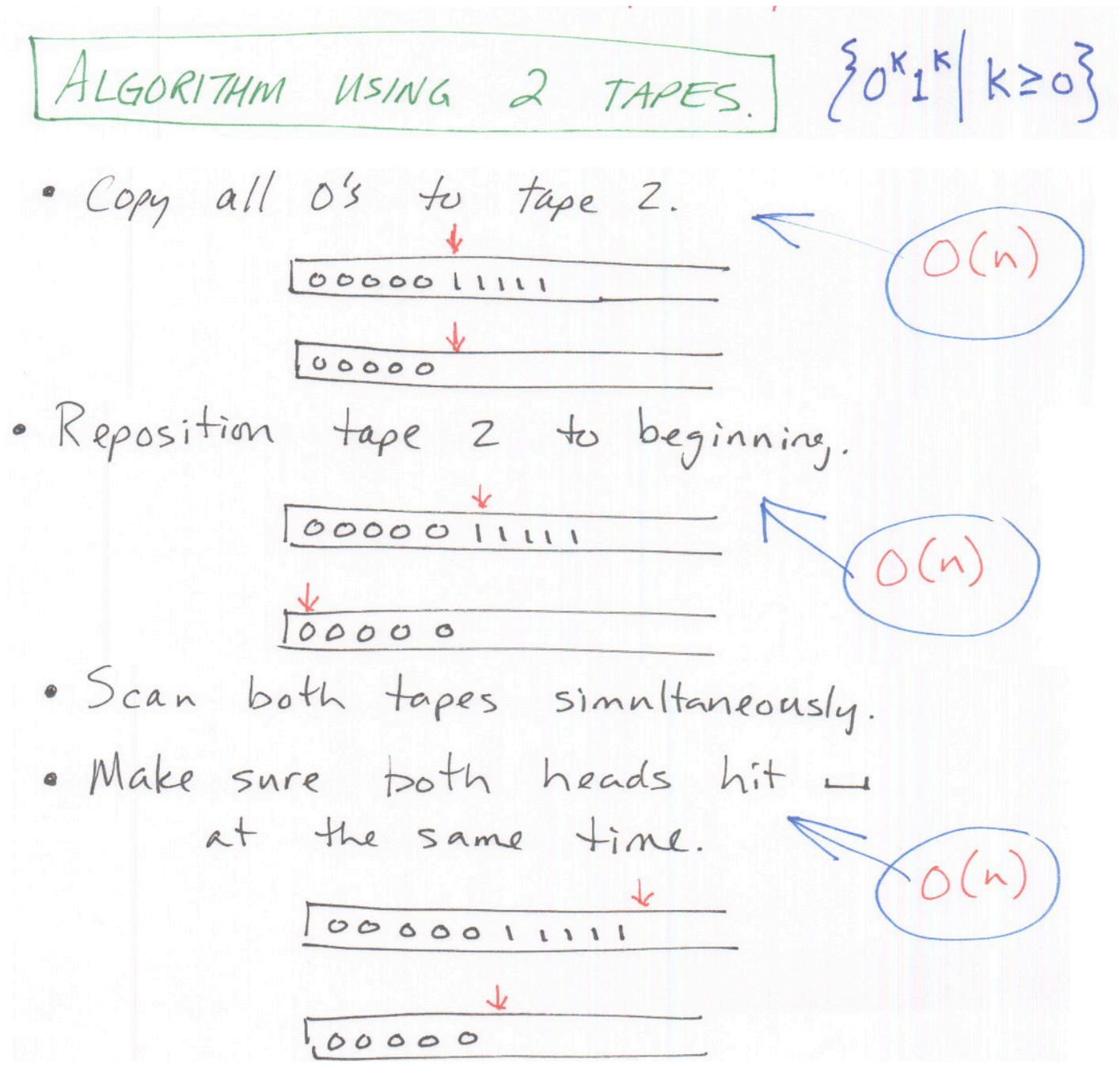
So: $\{0^k 1^k \mid k \geq 0\} \in TIME(n^2)$

Better algorithm may exist
Example of algorithm that take O(n log(n)) steps (check Harry Porter Video)

So: $\{0^k 1^k \mid k \geq 0\} \in TIME(n \log n)$

Different Computational Models

- Assume we have a 2 tape Turing Machine



slide from Harry Porter

Multi-tape vs. Single-tape TMs

- Theorem 7.8
  Every $t(n)$ multi-tape TM has an equivalent $O(t^2(n))$ single-tape TM

- Proof Idea:
  Every step in the multi-tape TM can be simulated by $O(t(n))$ steps of the single-tape TM

For $A = \{ 0^k1^k \mid k \geq 0 \}$, we saw:
Single-tape: $A \in$ TIME(n^2)
Single-tape: $A \in$ TIME($n \log n$)
Two-tape: $A \in$ TIME (n)

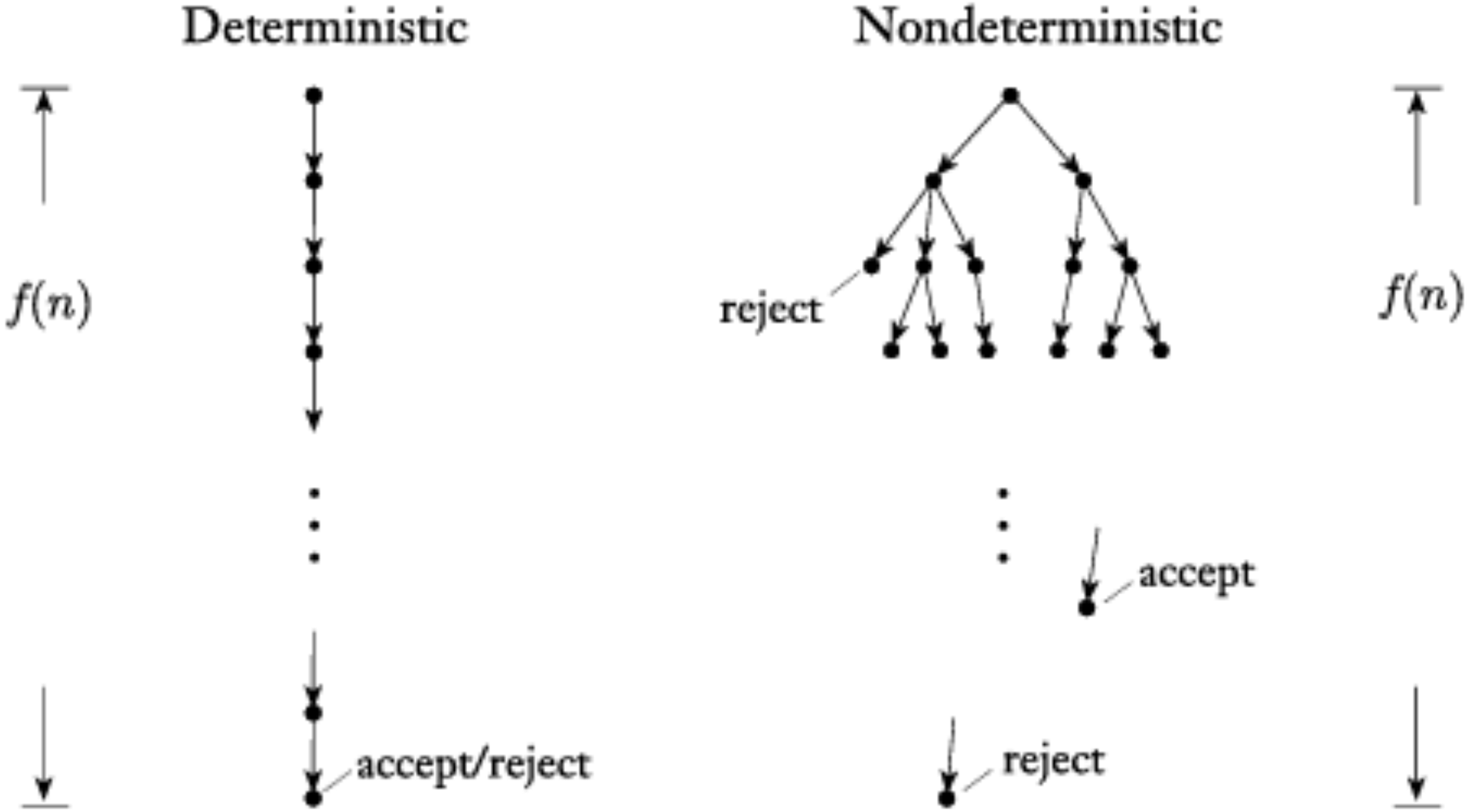
Summary

- The computation model matters

- However a polynomial time algorithm remains polynomial regardless on the model

As long as TMs deterministic!!

Time Complexity: NTM vs. DTM



Running Time of NTM:
The number of steps of the TM on the longest branch of computation

EVERY NONDETERMINISTIC TM CAN BE SIMULATED ON A DETERMINISTIC TM, USING EXPONENTIALLY MANY MORE STEPS.

NONDET TM

TAKES 419 STEPS ON INPUT w

DET SIMULATION

CAN BE DONE IN 2^{419} STEPS

NONDET TM

TAKES $O(N^2)$ TIME

DET SIMULATION

CAN BE DONE IN 2^{N^2} TIME

Class P

- P = class of languages decidable in polynomial time on a deterministic single-tape TM

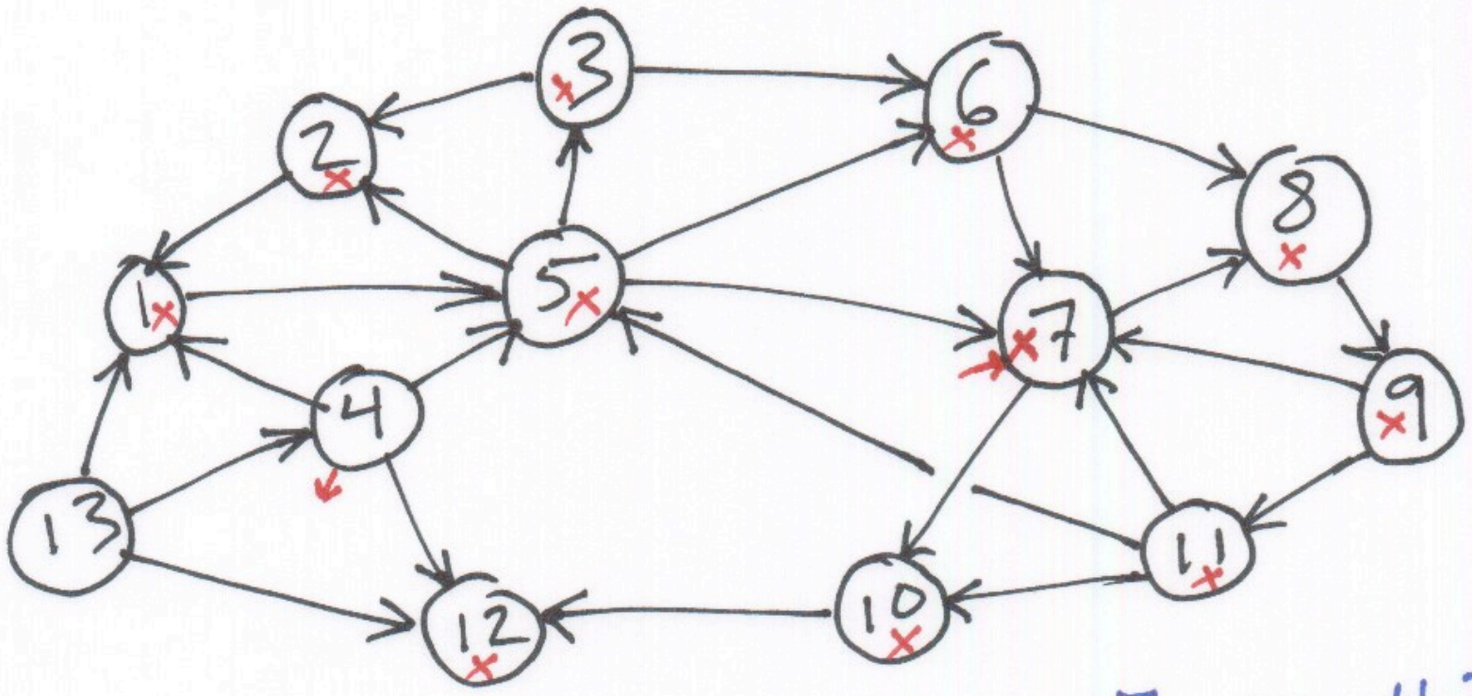
$$\mathrm{P} = \bigcup_k \mathrm{TIME}(n^k)$$

- P roughly corresponds to the class of problems that realistically solvable on a computer

THE "PATH" PROBLEM

INPUT: A DIRECTED GRAPH G AND TWO NODES, s and t...

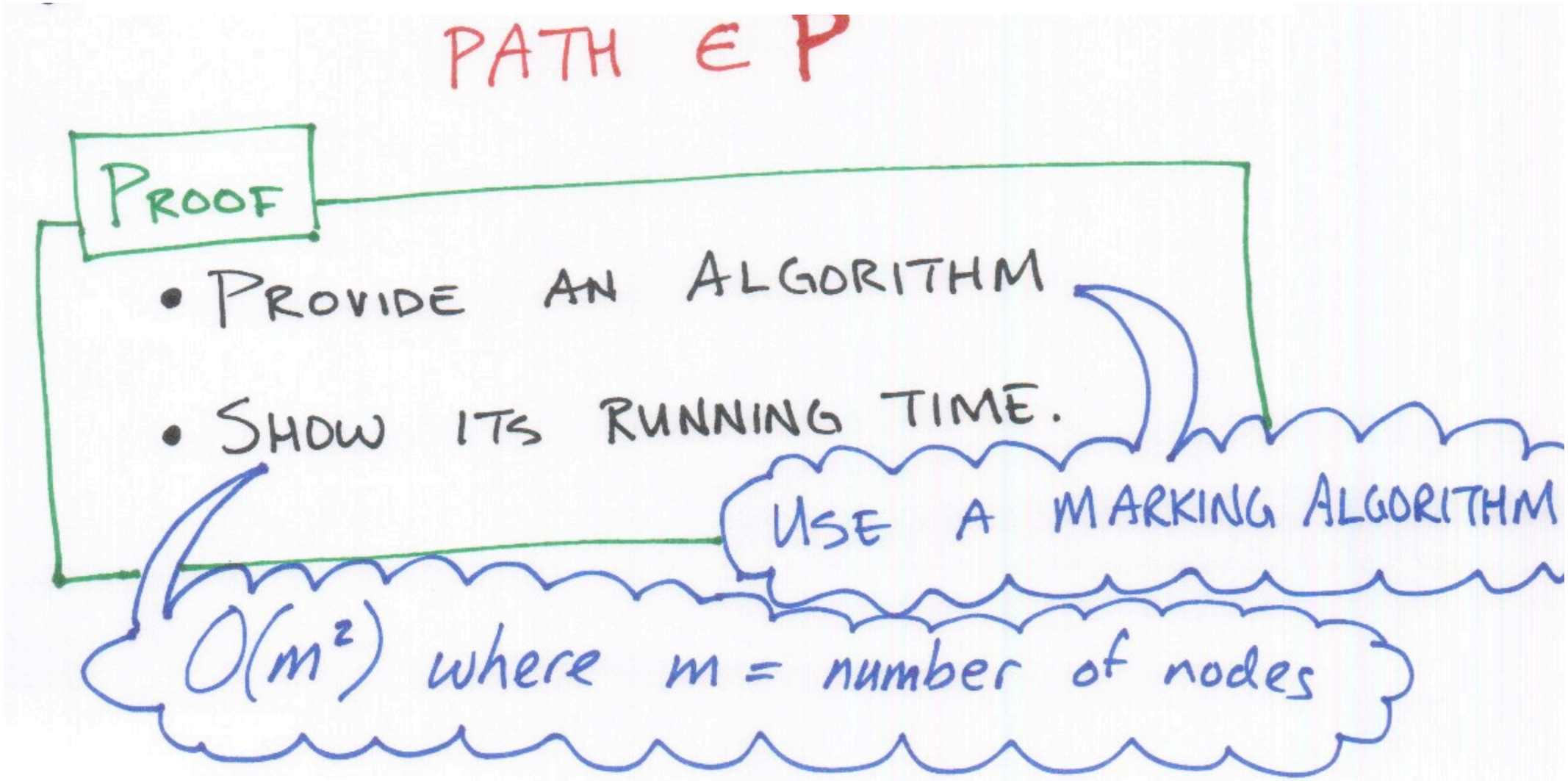
IS THERE A PATH FROM s TO t?



IS THERE A PATH FROM 7 TO 4?

M = "On input $\langle G, s, t \rangle$, where G is a directed graph with nodes s and t:

1. Place a mark on node s.
2. Repeat the following until no additional nodes are marked:
3. Scan all the edges of G. If an edge (a, b) is found going from a marked node a to an unmarked node b, mark node b.
4. If t is marked, *accept*. Otherwise, *reject*."



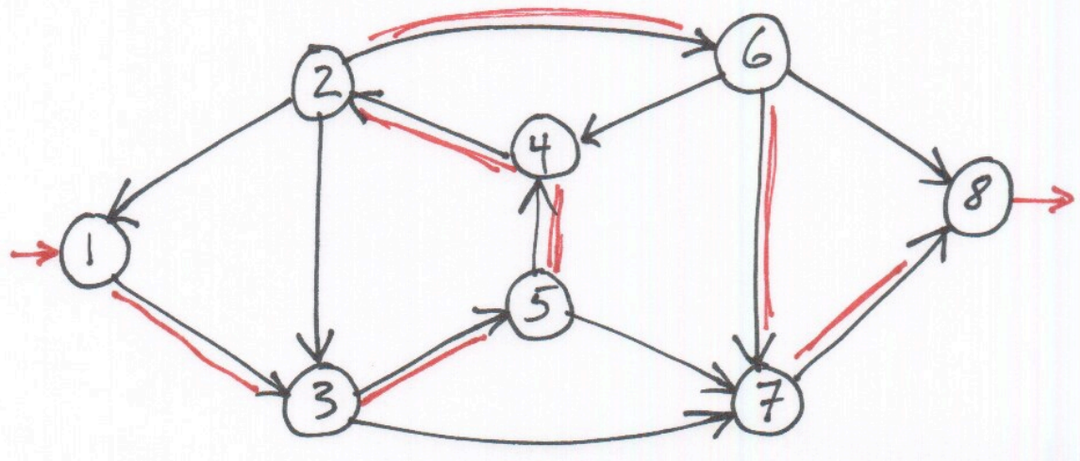
HAMPATH: THE "HAMILTONIAN" PATH PROBLEM

GIVEN A DIRECTED GRAPH, IS THERE A PATH THAT GOES THROUGH EVERY NODE EXACTLY ONCE?

$$HAMPATH = \{ \langle G, s, t \rangle \mid \begin{array}{l} G \text{ is a directed graph and there is} \\ \text{a "HAMILTONIAN" path from } s \text{ to } t \end{array} \}$$

WE ARE GIVEN THE STARTING AND ENDING NODES.

EXAMPLE: G, 1, 8



13542678