Copyright Warning

COMMONWEALTH OF AUSTRALIA
Copyright Regulation 1969

WARNING

This material has been copied and communicated to you by or on behalf of **Curtin University of Technology** pursuant to Part VB of the *Copyright Act 1968* (**the Act**)

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice

Curtin
University of Technology

Theoretical Foundations of Computer Science 300/552

Lecture 8

Algorithm Complexity

Curtin University of Technology

Outline

- What is computational complexity?
- Time complexity
- How to measure time complexity?
- Time complexity relationships.
- Space complexity
- Complexity classes: P, NP.
- NP-completeness.

Curtin University of Technology

Unit Learning Outcomes

- Define and describe P, NP, and NP-complete computation time problems, and explain their relationship and applications.

Curtin University of Technology

Assessment Criteria

- **Evaluate** the Polynomial time order of an algorithm given an Implementation Level Description of the Turing Machine that solves it.

- **Define** and relate the P and NP classes.

- **Classify** problems according to complexity classes.

Curtin University of Technology

TIME COMPLEXITY

Purpose

Aims

Measurement

Curtin University of Technology

Computational Complexity

- Sometimes a problem can be Turing Decidable but still may not be solvable in practice.

- Decidability tells you that it can be solved, but gives no idea of what resources are required to achieve this solution.
  - ➢ Recall that decidability assumes that the computation finished eventually (unbounded time) and that the tape is infinite (unbounded memory).
  - ➢ Real systems have bounds; both in terms of time and memory.

Curtin University of Technology

Computational Complexity

- Time complexity gives a general measure of the time required to solve a problem.
  - ➢ Uses broad categories (*e.g.*, $O(n)$) that are suitable for classification.
- Space complexity gives a general measure of the memory (RAM, swap, *etc*.) required to solve a problem.
- We mainly consider time complexity; the same concepts apply to space complexity.

Curtin University of Technology

Measuring Complexity

- Consider the language $A=\{0^k1^k \mid k \geq 0\}$
  - ➢ A is obviously decidable

- How much time does a TM take to decide it ?

- In order to estimate this, we require the TM description at a low level, including head movements.

- A high level description may not uniquely identify head movements, and thus leave uncertainty as to the complexity.

Curtin University of Technology

Analyzing Algorithms

- The number of steps an algorithm uses depends on many parameters
  - ➢ If input is a graph, the parameters may include number of nodes, number of edges, max degree of the graph
  - ➢ For simplicity we compute running time as a function of the length of the input string
- In worst-case analysis, we consider the longest running time of all inputs of particular length
- In average-case analysis, it is the average of the running times of all inputs of a certain length

Curtin University of Technology

FORMAL DEFINITION

Time Complexity Definition

Big-O

Example

Curtin University of Technology

Definition of time complexity

- Let M be a Deterministic Turing Machine that halts on all inputs.

- The running time or (worst case) time complexity of M is the function $f: N \rightarrow N$, 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 M is an $f(n)$ time TM.

Curtin University of Technology

Big-O Notation

- Exact running time is usually a complex expression, so we just estimate it.

- Asymptotic analysis is when we estimate running time for large inputs; only consider the highest order term of the running time expression
  - ➢ the highest order term dominates the other terms
  - ➢ all coefficients are ignored

Big-O Notation

- Example:
  - $f(n) = 6n^3 + 2n^2 + 20n + 45$
  - Ignore all terms other than the one with the most rapid growth ($6n^3$).
  - Ignore the co-efficient of this term (n^3).
  - The asymptotic (or big-O) notation for describing this relationship is $f(n) = O(n^3)$

- Big-O notation
  - Is a way of saying that one function is asymptotically no more than another
  - The exact details of the implementation will impact the details of time complexity but shouldn't affect its asymptotic complexity

Curtin University of Technology

Example

- Consider the language $A=\{0^k1^k \mid k \geq 0\}$
  - ➢ Need a TM
  - ➢ For approximation, can use Implementation Level description
    - Gives basic movement of head
    - Omits minor detail
    - Simpler than Formal Description of TM

Curtin University of Technology

Analysing algorithms

- 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 the following if both 0s and 1s remain on the tape.
  3. Scan across, crossing off a single 0 and a 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."

Analysing algorithms

- Consider each stage
  - ➢ Stage 1: uses n steps to scan and another n steps to reposition the head at the left-hand end. O(n) steps.
  - ➢ Stages 2 &3 : Each scan uses O(n) steps and at most n/2 scans can occur. $O(n^2)$ steps.
  - ➢ Stage 4: Single scan, uses O(n) steps.
  - ➢ Total time: $O(n) + O(n^2) + O(n)$
  - ➢ This gives $O(n^2)$

COMPLEXITY RELATIONSHIPS

Single Tape v Multi-Tape

Deterministic v Non-Deterministic

Exponential v Polynomial Time

Curtin University of Technology

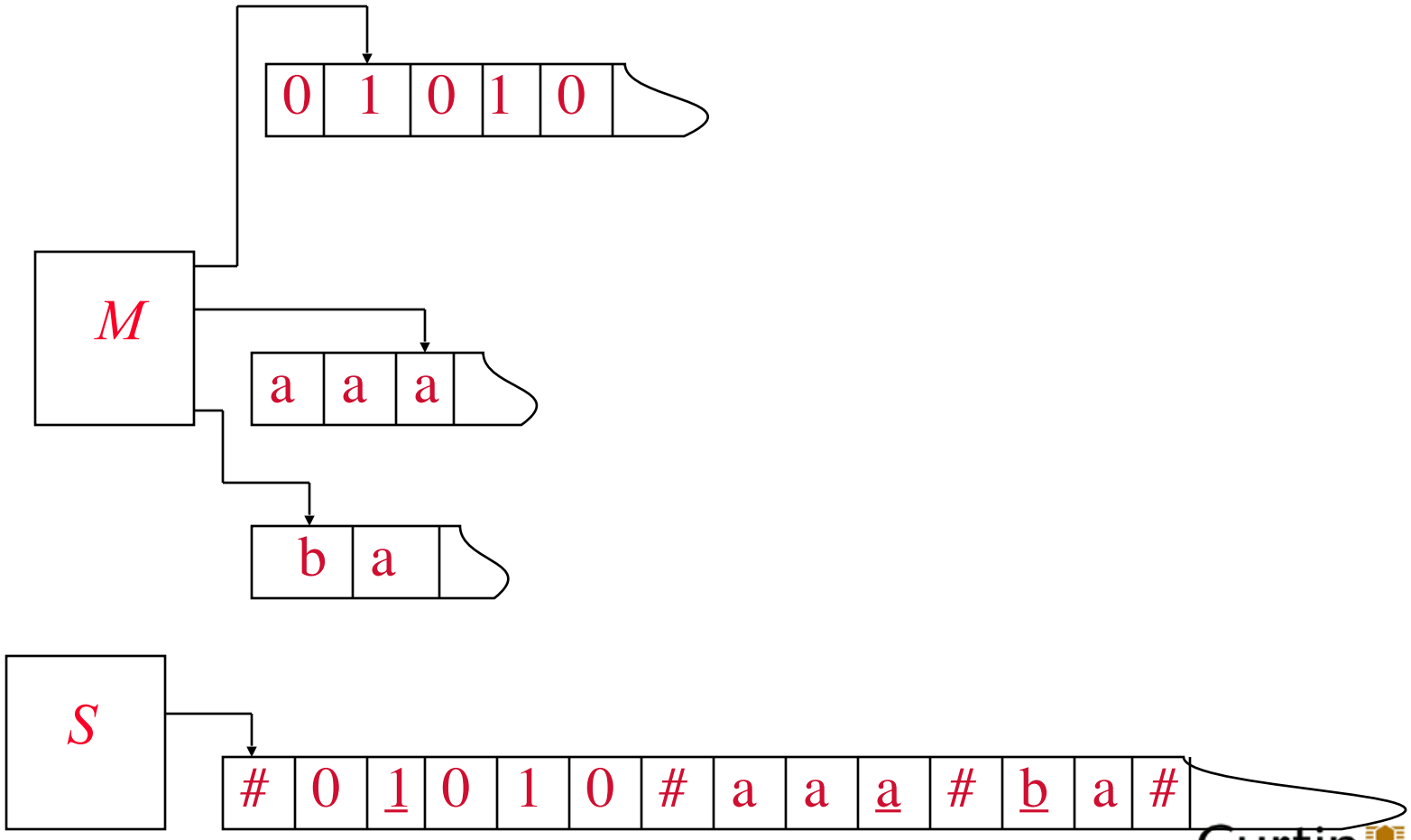
Complexity relationships among models

- In last week's lecture, we saw that different type of TMs all have the same expressive power; the same problems are decidable or undecidable for all models.
- The model **can** affect how long it takes to find the solution, and how much memory is used.

- Three models
  - ➢ single tape TM
  - ➢ multi-tape TM
  - ➢ Non-deterministic TM

Curtin University of Technology

Representing 3 tapes with one



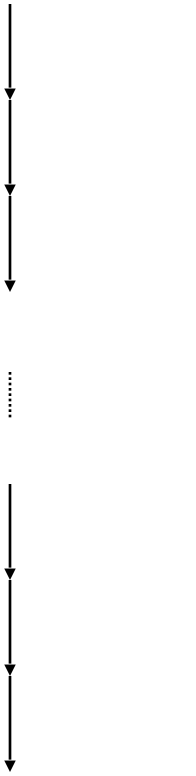
Complexity relationships among models

- Theorem:
  - Let $t(n)$ be a function, where $t(n) \geq n$. Every $t(n)$ time multi-tape TM has an equivalent $O(t^2(n))$ time single tape TM.
- Proof idea:
  - Simulation of multi-tape TM M on a single tape TM S
  - Each of the $t(n)$ steps of M takes $O(t(n))$ steps on S, giving $O(t^2(n))$ steps.

Curtin University of Technology

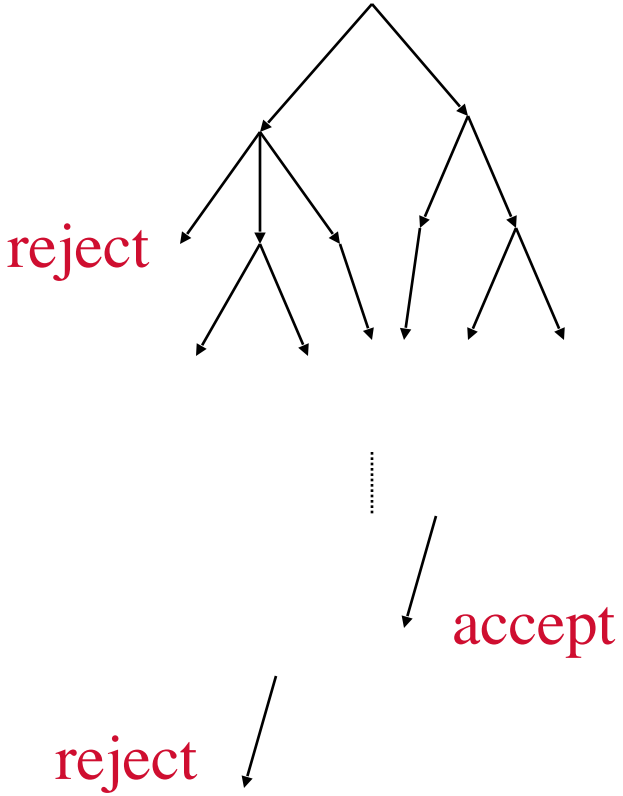
Deterministic and Nondeterministic computation

Deterministic



accept/reject

Nondeterministic



Curtin University of Technology

Time Complexity of NTM and DTM

- Theorem:
  - Let t(n) be a function, where $t(n) \geq n$. Every t(n) time single tape NTM has an equivalent $2^{O(t(n))}$ time single tape DTM.

- Proof idea:
  - Simulation of single tape NTM N on a single tape DTM D
  - On an input length of n, every branch of N's nondeterministic computation tree has a length of at most t(n)

Time Complexity of NTM and DTM

- Proof idea (contd.):
  - ➢ Every node in the tree can have at most b children
    - where b is the max number of choices given by N's transition function
  - ➢ Total number of leaves in the tree is at most $b^{t(n)}$, and total number of nodes, $O(b^{t(n)})$
  - ➢ Time for travelling from root to a node is O(t(n))
  - ➢ Running time for D is $O(t(n)\, b^{t(n)}) = 2^{O(t(n))}$

Curtin University of Technology

Exponential and Polynomial time

- Difference in time complexity between problems
  - ➢ Polynomial between complexity of problems on single tape and multi-tape TMs
  - ➢ Exponential between DTM and NTM
  - ➢ Polynomial time difference is considered small
    - *e.g.*, n=1000, n^3=1,000,000,000
  - ➢ Exponential time difference is large
    - *e.g.*, n=1000, 2^n is larger than the number of atoms in the universe

Curtin University of Technology

Time Comparisons

Each step of algorithm takes one tenth of a microsecond.
Total time needed by algorithms that take n, n^2, n^3, 2^n, 3^n steps

	10	20	30	40	50	60
n	.000001 sec	.000002 sec	.000003 sec	.000004 sec	.000005sec	.000006sec
n^2	.00001 sec	.00004 sec	.00009 sec	.00016 sec	.00025 sec	.00036 sec
n^3	.0001 sec	.0008 sec	.0027 sec	.0064 sec	.0125 sec	.0216 sec
2^n	.0001 sec	.1 sec	1.8 min	1.2 days	3.5 years	36.6 centuries
3^n	0.0059 sec	5.8 min	0.6 years	385 centuries	$2x\ 10^7$ centuries	$1.3x\ 10^{12}$ centuries

Curtin University of Technology

Exponential and Polynomial time

- Exponential time algorithms
  - ➢ arise in brute-force search of some solution spaces
  - ➢ *e.g.*, factoring numbers into primes by searching through all potential divisors
  - ➢ deeper understanding of a problem may reveal a polynomial time algorithm

- Polynomial equivalence
  - ➢ deterministic computational models with only polynomial time difference

Curtin University of Technology

COMPLEXITY CLASSES

Class P

Class NP

Curtin University of Technology

The Class P

- P is the class of languages that are decidable in polynomial time on a single tape DTM
- Class P is important because
  - ➢ P is invariant for all models of computation that are polynomially equivalent to single tape DTM
  - ➢ P roughly corresponds to class of problems that can be realistically solved on a computer

Curtin University of Technology

The Class P

- P is a robust class
  - ➢ not affected by the particulars of the model of computation
- P is relevant in practice
  - ➢ problems in P have a solution that takes time $O(n^k)$ for some k
  - ➢ threshold of practical solvability
  - ➢ Once a polynomial algorithm is found for a problem that was thought to require exponential time, some key insight is gained
    - further reductions in complexity usually follow

Curtin University of Technology

An example of problems in P

- *PATH*= {<G, s,t>|G is a directed graph that has a directed path from s to t}
  - ➢ For a given graph G, is there a path from s to t ?
- Theorem: *PATH* ∈P
- Proof idea:
  - ➢ Present a polynomial time algorithm that decides PATH.
  - ➢ Use a graph search method *e.g.*, breadth-first search
  - ➢ Successively mark all nodes reachable from s by paths of length 1, then 2, then 3, through m.

Curtin University of Technology

An example of problems in P

- *M*= "On input <*G*, *s*, *t*> where *G* is a directed graph with nodes *s* and *t*:
  1. Place a mark on *s*.
  2. Repeat the following until no additional nodes are marked
  3. Scan all edges of *G*. If an edge <*a*, *b*> goes from marked node *a* to unmarked node *b*, mark *b*.
  4. If *t* is marked, accept. Otherwise reject."

Curtin University of Technology

An example of problems in P

- M= "On input <G, s, t> where G is a directed graph with nodes s and t:
  1. Place a mark on s.
  2. Repeat the following stage until no additional nodes are marked
  3. Scan all edges of G. If an edge <a, b> goes from marked node a to unmarked node b, mark b.
  4. If t is marked, accept. Otherwise reject."

- Analysis
  - ➢ Stages 1,2 and 4 are executed only once, stage 3 at most m times (with $m<n$). There are at most |E| edges with $|E|<n^2$
  - ➢ Each stage is easily implemented in polynomial time
  - ➢ Hence M is a polynomial time algorithm

Curtin University of Technology

The Class NP

- NP is the class of languages that have polynomial time verifiers
  - ➢ Check if a proposed solution is a solution takes Polynomial time
  - ➢ Algorithm can be to check all solutions
- NP contains many problems of practical importance
  - ➢ NP time is *nondeterministic polynomial time*
  - ➢ TM runs in *nondeterministic polynomial time*
  - ➢ time of an NTM is the time used in its longest computation branch

Curtin University of Technology

The Class NP

- Avoiding brute-force search
  - ➢ for many interesting and useful problems not yet possible
  - ➢ reason for lack of success is not known
    - Are there better algorithms yet to be discovered or are these problems intrinsically difficult ?
- A remarkable discovery: Complexities of many problems are linked
  - ➢ a polynomial time algorithm for one of these will help solve an entire class of problems

Curtin University of Technology

Example of class NP

- Hamiltonian path in a directed graph
  - ➢ a path that goes through each node exactly once
  - ➢ Whether a directed graph contains a Hamiltonian path connecting two specific nodes
- *HAMPATH* = $\{<G, s,t> | G$ is a directed graph with a Hamiltonian path from s to $t\}$
  - ➢ Modify the algorithm for *PATH* by checking if a potential path is Hamiltonian
    - exponential time complexity
    - not known if a polynomial time algorithm exists

Curtin University of Technology

Polynomial verifiability

- Verifying whether a given path is Hamiltonian can be done in polynomial time
- Some problems may not be polynomially verifiable
  - ➢ Example: $\overline{HAMPATH}$
  - ➢ the complement of *HAMPATH,*
  - ➢ even if we could determine that a graph did not have a Hamiltonion path, we cannot verify the nonexistence in polynomial time

Curtin University of Technology

Polynomial verifiability

- A verifier for a language A is an algorithm V, where
  - $A = \{w \mid V \text{ accepts } \langle w,c \rangle \text{ for some string } c\}$
  - A polynomial time verifier runs in polynomial time in the length of w.
  - c is a certificate or proof of membership in A
  - For HAMPATH, a certificate for a string $\langle G,s,t \rangle \in$ HAMPATH is the Hamiltonian path from s to t.

Curtin University of Technology

Sample Algorithm for *HAMPATH*

- N_1 = "On input <G, s, t> where G is a directed graph with nodes s and t:
  1. Write a list of m numbers $p_1,...,p_m$, where m is the number of nodes in G. Each number in the list is non-deterministically selected to be between 1 and m.
  2. Check for repetitions in the list. If any are found, reject.
  3. Check whether $s=p_1$ and $t=p_m$. If either fail, reject.
  4. For each i between 1 and m-1, check whether (p_i,p_i+1) is an edge of G. If any are not reject. Otherwise accept."

Curtin University of Technology

Analysis

- N_1 = "On input <G, s, t> where G is a directed graph with nodes s and t:
  1. Write a list of m numbers $p_1,...,p_m$, where m is the number of nodes in G. Each number in the list is non-deterministically selected to be between 1 and m.
  2. Check for repetitions in the list. If any are found, reject.
  3. Check whether $s=p_1$ and $t=p_m$. If either fail, reject.
  4. For each i between 1 and m-1, check whether (p_i,p_i+1) is an edge of G. If any are not reject. Otherwise accept."

- To verify that the algorithm runs in nondeterministic polynomial time
  - examine each of the stages
  - each stage runs in polynomial time
  - the whole algorithm runs in nondeterministic polynomial time
    - because stage 1 contains a nondeterministic selection

Curtin University of Technology

NP theorem

- A language is in NP iff it is decided by some nondeterministic polynomial time Turing machine
- Proof idea:
  - How to convert a polynomial time verifier to an equivalent polynonial time NTM and vice versa.
  - NTM simulates a verifier by guessing the certificate.
  - The verifier simulates an NTM by using the accepting branch as the certificate.

Curtin University of Technology

Proof of NP theorem

- Assume that V is a polynomial time verifier for A∈NP
  - ➢ V is a TM that runs in time n^k.
  - ➢ Construct an NTM as follows:
  - ➢ N= "On input w of length n,
  1. Nondeterministically select string c of length n^k.
  2. Run V on input <w,c>.
  3. If V accepts, accept; otherwise reject.

- Assume that A is decided by a polynomial time NTM N
  - ➢ Construct a polynomial time verifier V as follows:
  - ➢ V= "On input <w,c>, where w and c are strings:
  1. Simulate N on input w, treating each symbol of c as a description of the nondeterministic choice to make at each step
  2. If this branch of N's computation accepts, accept; otherwise, reject.

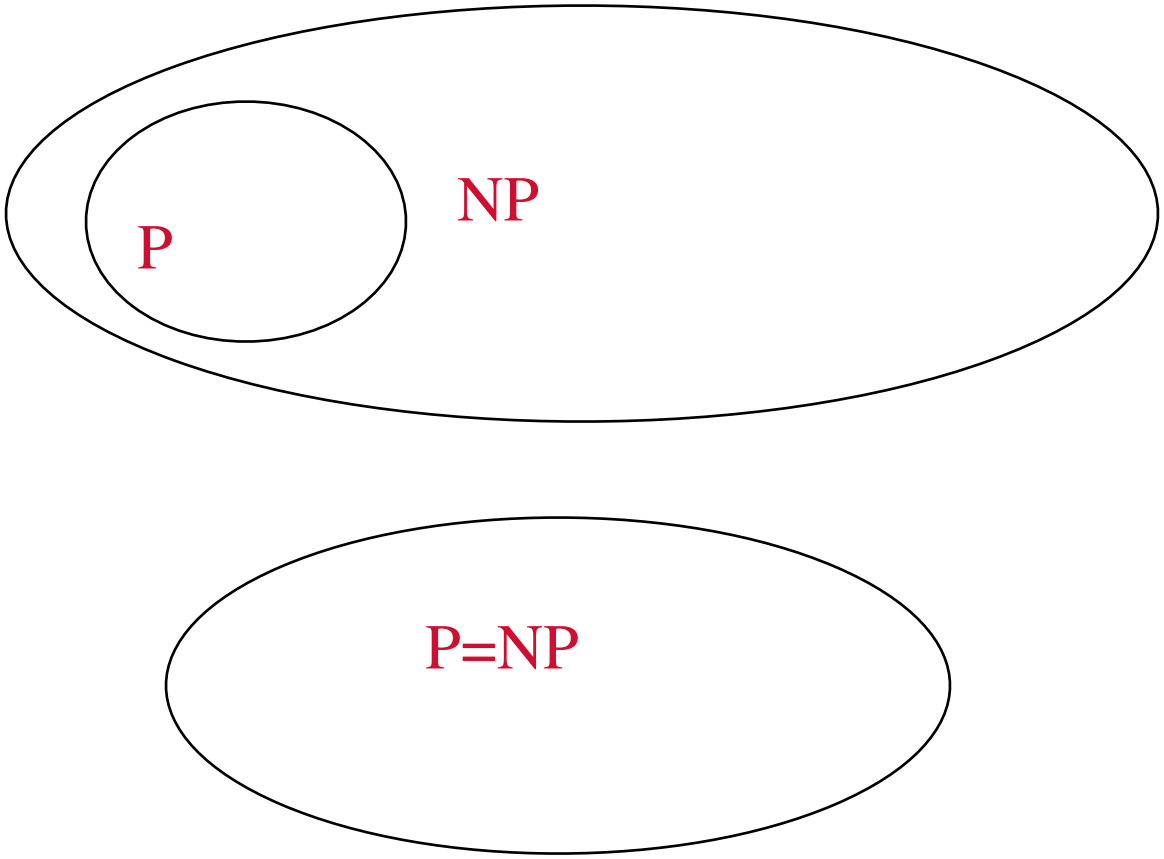
Curtin University of Technology

P versus NP

- P= class of languages where membership can be decided quickly
- NP= class of languages where membership can be verified quickly
- Could P=NP ?
  - ➢ Unable to prove any language in NP to be not in P
  - ➢ If P=NP, any polynomially verifiable problem could be polynomially decidable
  - ➢ Most researchers believe that the two classes are not equal
    - because no polynomial time algorithms have been found for certain problems in NP

Curtin University of Technology

P versus NP



- One of these two possibilities is correct.
- Prove which one and earn US$1,000,000.

Curtin University of Technology

Summary

- What is computational complexity?
- Time complexity.
- How to measure time complexity?
- Time complexity relationships.
- Complexity classes: P, NP.