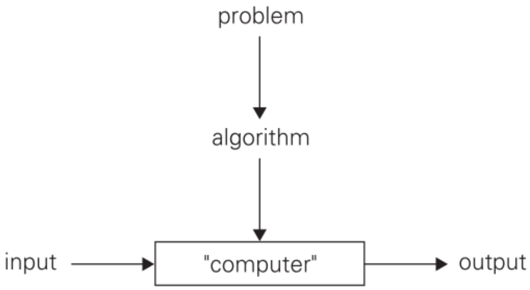
CS142 Wrapped

(*Design and Analysis of Algorithms*)

Department of Mathematics and Computer Science
College of Science
University of the Philippines Baguio

Algorithms

- The notion of an algorithm consitutes the **cornerstone of computer science**.
- Algorithms lie at the heart of **practical computing**.



Classifying Algorithms

Although algorithms can be classified in numerous ways, two of them are particularly important:

- **By Underlying Design Technique**
- **By Efficiency**

These two principal dimensions reflect the **needs of computing practice**.

General Design Techniques

1. Brute Force/ Exhaustive Search
2. Decrease-and-Conquer
3. Divide-and-Conquer
4. Transform-and-Conquer
5. Space-Time Trade-offs
6. Dynamic programming
7. Greedy technique
8. Backtracking
9. Branch-and-bound

There are other areas of algorithmics:

- **randomized algorithms** – makes random choices during its execution
- **parallel algorithms** – takes advantage of the capability of some newer computers to execute operations concurrently

Basic Efficiency Classes

The analysis framework classifies algorithms by the **order of growth** of their running time as a function of input size.

Class	Notation	Important examples
constant time	$\Theta(1)$	hashing (on average)
logarithmic	$\Theta(\log n)$	binary search (worst and average cases)
linear	$\Theta(n)$	sequential search (worst and average cases)
linearithmic	$\Theta(n \log n)$	advancced sorting algorithms
quadratic	$\Theta(n^2)$	elementary sorting algorithms
cubic	$\Theta(n^3)$	Gaussian elimination
exponential	$\Omega(a^n)$	combinatorial problems

Lower Bound Arguments and Decision Trees

Given a class of algorithms for solving a particular problem, a **lower bound** indicates the *best possible efficiency* any algorithm from this class can have.

- **Information−theoretic lower bound** - usually obtained through a mechanism of **decision trees**. This technique is particularly useful for comparison-based algorithms for sorting and searching. Specifically,
  - Any general comparison-based sorting algorithm must perform at least $\lceil \log_2 n! \rceil \approx n \log_2 n$ key comparisons in the worst case.
  - Any general comparison-based algorithm for searching a sorted array must perform at least $\lceil \log_2(n+1) \rceil$ key comparisons in the worst case.

Lower Bound Arguments and Decision Trees

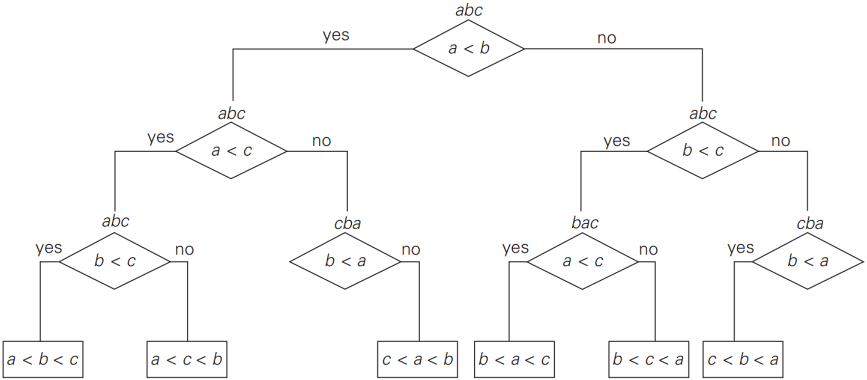


Figure: Decision tree for the three-element selection sort.

Lower Bound Arguments and Decision Trees

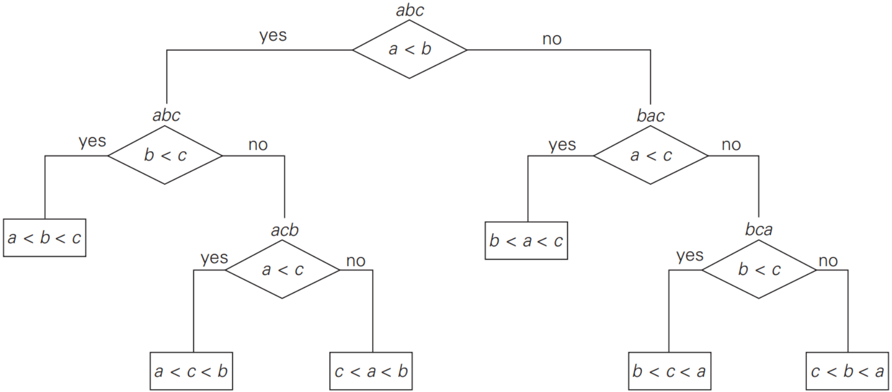


Figure: Decision tree for the three-element insertion sort.

Complexity Theory

Complexity theory seeks to classify problems according to their computational complexity: **tractable** and **intractable problems** – problems that can and cannot be solved in polynomial time, respectively.

Class P - class of all decision problems that can be **solved** in polynomial time

Class NP - class of all decision problems whose randomly guessed solutions can be **verified** in polynomial time.

Is $P = NP$?

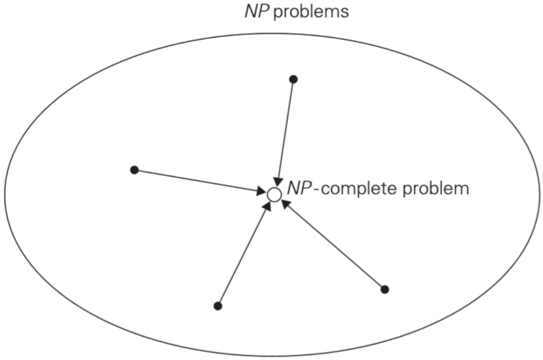
Most important unresolved issue in theoretical computer science

$NP-$ Complete Problems

A decision problem D is said to be $NP-$**complete** if:

(i) it belongs to class NP

(ii) every problem in NP is **polynomially reducible** to D

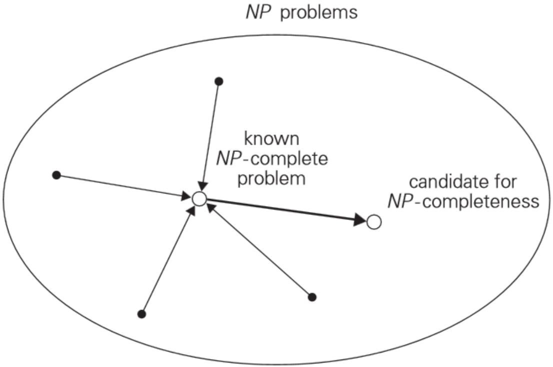


$NP-$ Complete Problems

A decision problem D is said to be $NP-$**complete** if:

(i) it belongs to class NP

(ii) every problem in NP is **polynomially reducible** to D



$NP-$**completeness** implies that if there exists a deterministic polynomial-time algorithm for just one $NP-$complete problem, then $P = NP$.

“ …whichever direction you take in your future journey through the land of algorithms in your studies and your career, the road ahead is as exciting as it has ever been.”
-Anany Levitin

The **field of algorithms** is **constantly evolving**.
Stay curious and keep building your understanding.

Thank you for stopping by at the CS142 Place!