CSCI203

Algorithms and Data Structures

Problem Complexity

Lecturer: Dr. Fenghui Ren

Room 3.203

Email: fren@uow.edu.au

Classes of Algorithms

- Logarithmic
  - (all bases have the same growth rate)
  - $\Theta(\log n)$,
  - $\Theta(\log(\log n))$
- Poly-logarithmic
  - $\Theta(n \log n)$
  - $\Theta(n^2 \log n)$
  - …
- Polynomial
  - (exponential is constant)
  - $\Theta(1)$ - sub-linear
  - $\Theta(n^{0.001})$- sub-linear
  - $\Theta(n^{0.5})$ - sub-linear
  - $\Theta(n)$ - linear
  - $\Theta(n^2)$
  - $\Theta(n^3)$
  - …$\Theta(n^{100})$

Classes of Algorithms

- Expotential
  - (base makes a bug difference)
  - $\Theta(2^n)$
  - $\Theta(3^n)$
  - $\Theta(n^n)$
- Exponential-Exponential
  - $\Theta(n^{n^n})$
- Factorial
  - $kC^n < \mathrm{n!} < \mathrm{n^n}$
  - k is a constant and c is a constant less than n

The Great Divide

- All polynomial algorithms, include
  - Linear
  - Poly-log
  - Log
  - Constant
  - Sub-linear
- All exponential algorithms, include
  - Factorial
  - poly-poly

Problems vs. Algorithm

- A problem can have many algorithms with different efficiencies, e.g. sorting integers
  - Mergesort $\Theta(n \log n)$
  - Permutation sort $\Theta(n!)$
- A specific algorithm's efficiency can vary depending on the input, e.g. Quicksort
  - $\Theta(n \log n)$ - on most unsorted list
  - $\Theta(n^2)$ – on pre-sorted or partially sorted list

Problems vs. Algorithms

- So what can we say about sorting problem?
- The sorting problem can be
  - Solved in $\Theta(n \log n)$
  - Solved in $\Theta(n!)$

- We say that the sorting problem is polynomial
  - Because there exists at least one polynomial algorithm for solving the problem

Problems vs. Algorithms

- Some problems have solution algorithms that are intrinsically polynomial
- But, some problems can NOT be solved in polynomial time
  - At least most people think so…
  - But it's hard to prove that something can't be done

Longest Common Subsequence (LCS) Problem

- Finding the longest subsequence common to all sequences in a set of sequences (often just two sequences).
- It differs from the longest common substring problem:
  - unlike substrings, subsequences are not required to occupy consecutive positions within the original sequences.

- E.g. $X = ABCBDAB; Y = BDCABA$
  - The length of LCS is 4
  - LCS are $BDAB, BCAB, BCBA$

LCS: 1960's

- The LCS problem
  - Fastest known algorithm can only solve the problem in $\Theta(2^n)$
- Does there exist an optimal LCS algorithm that has a polynomial time?
  - In 1960, most computer scientists were leaning towards an answer of "no", i.e. there does NOT exist a polynomial algorithm

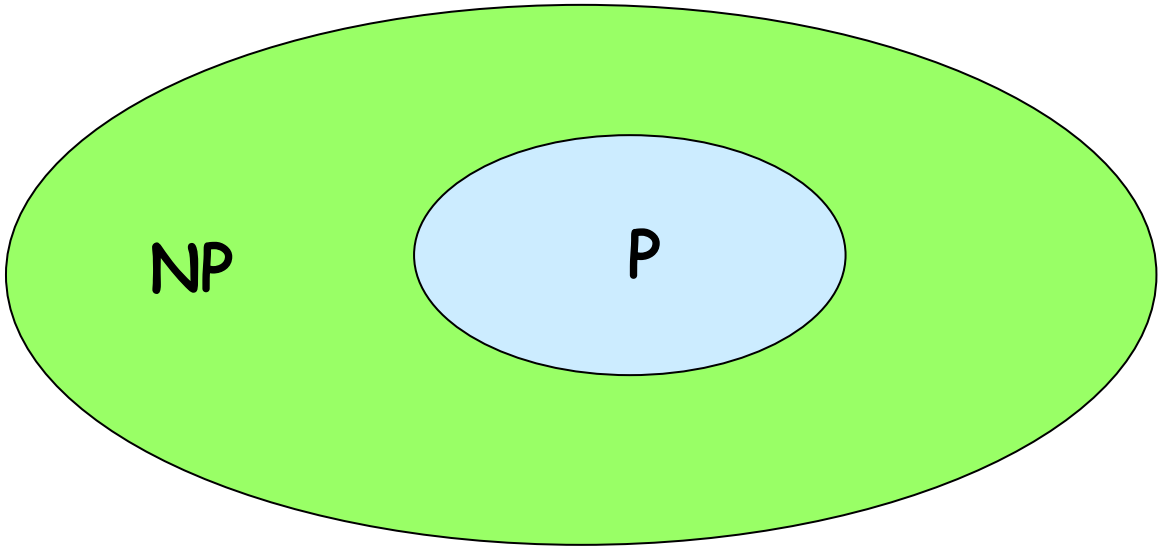
LCS: 1980's

- In 1981, Smith and Waterman used dynamic programming to solve LCS problem in $\Theta(n^2)$ time using $\Theta(n^2)$ memory
- This discover and many other discoveries led the following questions:

- Can all problems be solved in Polynomial Time?
  - Maybe we just haven't discovered all the good algorithms yet

P and NP

- NP – set of problems where the solution can not be computed deterministically in polynomial time
- P – set of problems that can be solved deterministically in polynomial time



Nondeterministic algorithm

- a nondeterministic algorithm is an algorithm that, even for the same input, can exhibit different behaviours on different runs
  - The nondeterministic algorithms are often used to find an approximation to a solution, when the exact solution would be too costly to obtain using a deterministic one

- a deterministic algorithm is an algorithm that, given a particular input, will always produce the same output, with the underlying machine always passing through the same sequence of states.

Polynomial Problems

- Searching
- Sorting
- Minimum Spanning Tree
- LCS Problem
- Shortest Path Problem

Non-deterministically Polynomial Problems

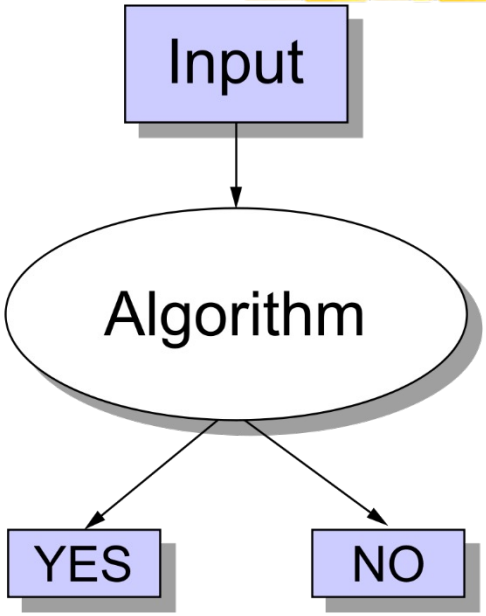
- The **graph isomorphism** – problem of determine whether two graph can be drawn identically
- The **traveling salesman problem** – where we want to know if there is a route of same length that goes through all nodes in a certain network (graph)
- **Knapsack problem**

NP-completeness

- $NP - complete$
  - Is short for "nondeterministic polynomial-time complete"
- A decision problem D_1 is said to be $polynomially\ reducible$ to a decision problem D_2, if there exists a function t that transforms instances of D_1 to instances of D_2 such that:
  - t maps all yes instances of D_1 to yes instances of D_2 and all no instances of $D_1$to no instances of D_2
  - t is computable by a polynomial time algorithm
- A decision problem D is said to be $NP - complete$ if:
  - it belongs to class NP
  - every problem in NP is polynomially reducible to D

What is a decision problem?

- a decision problem is a problem that can be posed as a yes-no question of the input values
  - whether a given natural number is prime
  - given two numbers x and y, does x evenly divide y?



What about optimization problems?

- $NP - completeness$ has been studied in the framework of decision problems.
- Most problems are not decision problems, but optimization problems (where some value needs to be minimized or maximized).
- In order to apply the theory of NP-completeness to optimization problems, we must recast them as decision problems.

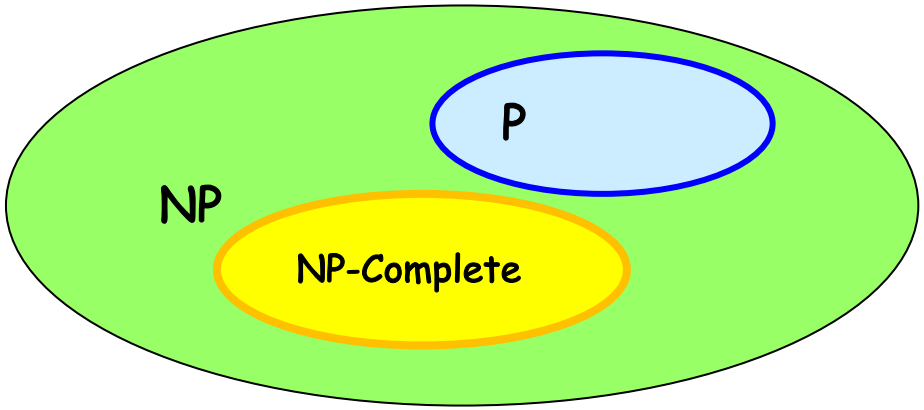
- We provide an example of how an optimization problem can be transformed into a decision problem.

An example

- Optimization problem
  - SHORTEST-PATH that finds a shortest path between two given vertices in an unweighted, undirected graph $G = (V, E)$.
- Decision problem - A decision problem PATH related to the SHORTEST-PATH problem above is :
  - Given a graph $G = (V, E)$, two vertices $u, v \in V$, and a non-negative integer k, does a path exist in G between u and v whose length is at most k ?

P, NP and NP-Complete

- P – Problems with polynomial deterministic algorithms
- NP – problems with polynomial non-deterministic algorithms
- $NP\ Complete$ – A Problem X is NP-complete if there is an NP problem Y, such that Y is reducible to X in polynomial time.



Related References

- Introduction to the Design and Analysis of Algorithms, A. Levitin, 3rd Ed., Pearson 2011.
  - Chapters 11.3

CSCI203

Algorithms and Data Structures

Subject Review & Final Examination

Lecturer: Fenghui Ren

Room 3.203

Topics covered in the subject

- Week01
  - Comparing algorithms and complexity classes
  - 1D and 2D Peak finding, phone books and Quick sort
  - Basic data structures – array, list, stack, queue and record
- Week02
  - Merge sort, heap, heapsort, compact string storage using arras
  - Algorithm efficiency
- Week03
  - Discrete event-driven simulation
  - Min-max heap, priority queue

Topics covered in the subject…

- Week04
  - Binary search trees (BST), AVL trees
  - Binary express trees, K-ary trees & State-machines
- Week05
  - 2-4 trees
  - B-trees of order m
  - Quadtrees and fast search (hashing)
- Week06
  - Hashing - chainning, open addressing, linear probing and double hashing
  - Karp-Rabin string search

Topics covered in the subject…

- Week07 – Graphs
  - Adjacency matrix, Breadth-first-search (BFS), DFS, edge classification, cycle detection, topological sort
  - Articulation points
- Week08 – Weighted Graphs
  - Shortest path (Dijkstra's, Bellman-Ford's)
- Week09 – Dynamic Programming
  - A* algorithms
  - Big numbers

Topics covered in the subject…

- Week10 ~ 12 – Dynamic Programming
  - Optimal structure, recursive formula
  - Implementation (recursive, recursive + memorization, bottom-up with tabular)
  - Fibonacci, coin-row, change-making, shortest paths, coin-collection, rod-cutting
  - Matrix-chain multiplication
- Week13
  - P, NP and NP-complete Problems

Subject Materials for Review

- Lecture slides:
  - Available on the subject Moodle.
- Your notes
- Recommended references:
  - Introduction to the Design and Analysis of Algorithms, A. Levitin, 3rd Ed., Pearson 2011.
  - Introduction to Algorithms, T. H. Cormen, 3rd Ed, MIT Press 2009.
- Assignments
- Lab exercises

Assessments

- Assignments (45% in total)
  - 3x Coding assignments = 45%
- Final Exam (55%)
  - Minimum requirement 40% = 22 marks

Final Examination

- Materials and Aids Allowed
  - Open book
- Exam Structure
  - Short answer questions
  - 8 questions, 6-8 marks each as specified
  - Each question has multiple sub-questions
- This exam will run via Moodle

Final Examination…

- Exam Date & Starting time
  - **9:00 (Sydney time) Friday 11 November 2022**
  - Please check SOLS
- Exam Duration
  - 3 hours
- Grace Period
  - 15 minutes for preparing and submitting answer sheets in a single pdf file

Final Examination - Instructions

- Have a set of A4 blank paper ready
- On the first page, write
  - Your full name, Student Number & UOW login name
- Answer each question on a separate page clearly
  - either handwriting or using suitable editing software at your own choice
- Scan or take photos of your answer sheets and convert them into one single pdf file (<200MB)
- Name the pdf file as
  - <your login name>.pdf
- Submit the pdf file via Moodle
  - See the next slide on how to scan/convert your hand-write answer sheets into a single pdf file using your mobile

How to create one pdf file

- Important: Be prepared with knowing how to create one pdf file from your working solutions.
- There is freely available software that can be used to scan your answer sheets and convert them into a single pdf file. These links may be of assistance.

- **Android**

  https://www.youtube.com/watch?v=BCccqxhPyJw (Scan documents)
  https://www.youtube.com/watch?v=d_olWftSgIM (Convert image to pdf)

- **iPhone**

  https://www.idownloadblog.com/2017/05/12/how-to-save-photos-pdf-iphone-ipad/

  https://www.igeeksblog.com/how-to-convert-photos-to-pdf-on-iphone-ipad/

Example Questions

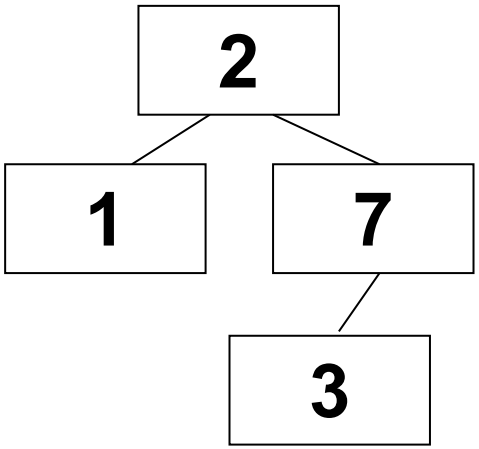
- Disclaimer
  - *These examples are only for illustration purpose. They DO NOT indicate in any way the scope and questions of the final exam.*

Types of questions

- About the data structures and algorithms
- Workout the final and or interim results of algorithms given the initial data or interim results of the algorithms

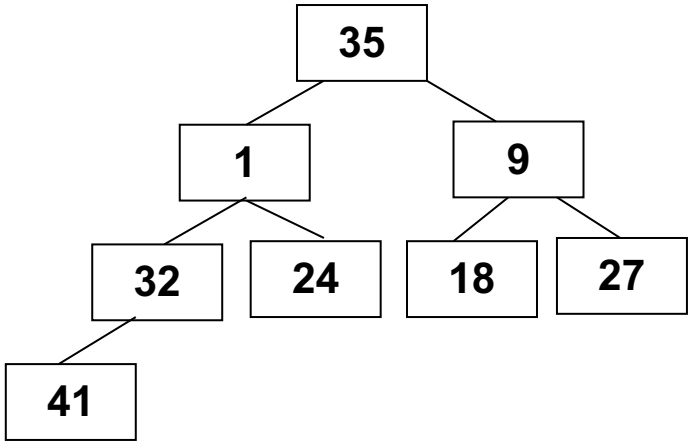
Examples…

- Draw the AVL tree resulting from the insertion of the node 4, into the tree:



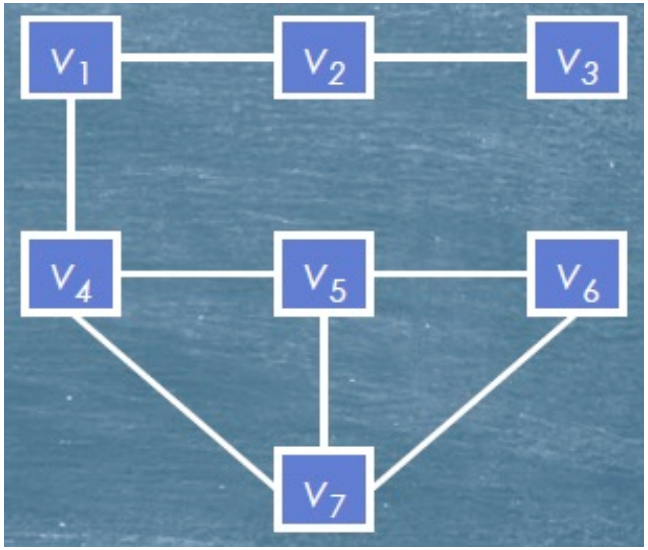
Examples…

- Show the tree resulting from the operation of the $siftdown$ function, which is called on the root node to restore the min heap property for the following tree:



Examples…

- List the nodes in the order in which they are encountered if the graph is traversed using Breadth-First and Depth-First search, starting from v_1



Examples…

- Consider inserting a set of keys $899, 950, 369, 980, \cdots$ into a hash table for a given length m and a scheme of probing, e.g linear probing or double hashing
  - Note: probing hashing function(s) will be given

Examples…

- The solution to a changing-making problem can be expressed as

$$F(n) = \min_{j:n \geq d_j} \{F(n - d_j)\} + 1 \quad for\ n > 0$$
$$F(0) = 0$$

- Where $F(n)$ be the minimum number of coins whose values add up to n; $F(0) = 0$ using the minimum number of coins of denominations $d_1 < d_2 < \ldots < d_m$, where $d_1 = 1$.
- What is the solution to the Change-making to amount n=6 and with denominations 1, 3 and 4, show $F[0 \ldots 6]$