

Faculty of Information Technology, Monash University

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

This material has been reproduced and communicated to you by or on behalf of Monash University 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 reproduction or communication of this material by you may be the subject of copyright protection under the Act. Do not remove this notice

FIT2004, S2/2016

Week 12: P vs NP, Primes, and Design Principles

Lecturer: Muhammad Aamir Cheema

ACKNOWLEDGMENTS

The slides are based on the material developed by [Arun Konagurthu](#) and [Lloyd Allison](#).

Announcements

- Pre-Exam consultation times published on Moodle
 - I will be overseas from (30th Oct - 5th Nov), but during this period
 - ✦ Four consultations will be provided by tutors
 - ✦ I will monitor emails and Moodle forums
 - I will provide consultations on 7th Nov and 8th Nov (see Moodle)

Overview

- P vs NP
- Primes
- Design Principles
- Programming Competition Winners
- Final Exam etc.


Recommended reading

- “Introduction to Algorithms”, Thomas Cormen et. al (Chapter 34)
- Primes
- Fermat’s Primality Test

Who Wants to be a Millionaire

Clay Institute of Mathematics listed 7 Millennium Problems in 2000.

The correct solution to a problem = 1 Million USD

1. Yang–Mills and Mass Gap
2. Riemann Hypothesis
3. P vs NP Problem
4. Navier–Stokes Equation
5. Hodge Conjecture
6. Poincaré Conjecture 
7. Birch and Swinnerton-Dyer Conjecture

Grigori Perelman Solved Poincaré Conjecture

Grigori Perelman, a Russian Mathematician, solved Poincaré Conjecture

- Declined to accept \$1 Million prize
 - Because he thought it was unfair and his contribution was no greater than another Mathematician Richard Hamilton
- 2006: Fields Medal (Math's Nobel Prize)
 - declined to accept
 - “I'm not interested in money or fame; I don't want to be on display like an animal in a zoo.”



What is P vs NP problem?

Growth Rates

N	log(N)	N	Nlog(N)	N ²	2 ^N	N!
10	0.003 μ s	0.01 μ s	0.033 μ s	0.1 μ s	1 μ s	3.63 ms
20	0.004 μ s	0.02 μ s	0.086 μ s	0.4 μ s	1 ms	77.1 years
30	0.005 μ s	0.03 μ s	0.147 μ s	0.9 μ s	1 sec	8.4x10 ¹⁵ years
40	0.005 μ s	0.04 μ s	0.213 μ s	1.6 μ s	18.3 min	
50	0.006 μ s	0.05 μ s	0.282 μ s	2.5 μ s	13 days	
100	0.007 μ s	0.1 μ s	0.644 μ s	10 μ s	4x10 ¹³ years	
1,000	0.010 μ s	1 μ s	9.966 μ s	1 ms		
10,000	0.013 μ s	10 μ s	130 μ s	100 ms		
100,000	0.017 μ s	100 μ s	1.67 ms	10 sec		
1,000,000	0.020 μ s	1 ms	19.93 ms	16.7 min		
10,000,000	0.023 μ s	10 ms	0.23 sec	1.16 days		
100,000,000	0.027 μ s	0.1 sec	2.66 sec	115.7 days		
1,000,000,000	0.030 μ s	1 sec	29.90 sec	31.7 years		

Assuming one instruction takes 1 nanosecond (10⁻⁹ seconds)

P vs NP (P is for Polynomial!)

A complexity is called polynomial if it is upper bounded by a polynomial $O(N^k)$ where k is a constant. e.g.,

- Constant $O(1)$
- Linear $O(N)$
- Quadratic $O(N^2)$
- Cubic $O(N^3)$
- Logarithmic $O(\log N)$
- Superlinear $O(N \log N)$

Is $O(N \log N)$ polynomial?

Non-Polynomial is a complexity that cannot be bounded by $O(N^k)$, e.g.,

- $O(2^N)$
- $O(N!)$
- NP **DOES NOT** stand for Non-Polynomial.
- It stands for Non-deterministic Polynomial (to be explained soon)

Decision Problem

A problem is called a decision problem if it has only a **YES** or a **NO** solution, e.g.,

- Does there exist a path from s to t , (i.e., are s and t connected)?
- Does this graph have a cycle?
- Is there a clique of size k in this Graph?
- Is there a vertex cover of size k in this Graph?

The following are not decision problems.

- Find the shortest path from s to t .
- Find a cycle in this graph.
- Find a vertex cover of size k in this Graph.
- Find a clique of size k in this Graph.

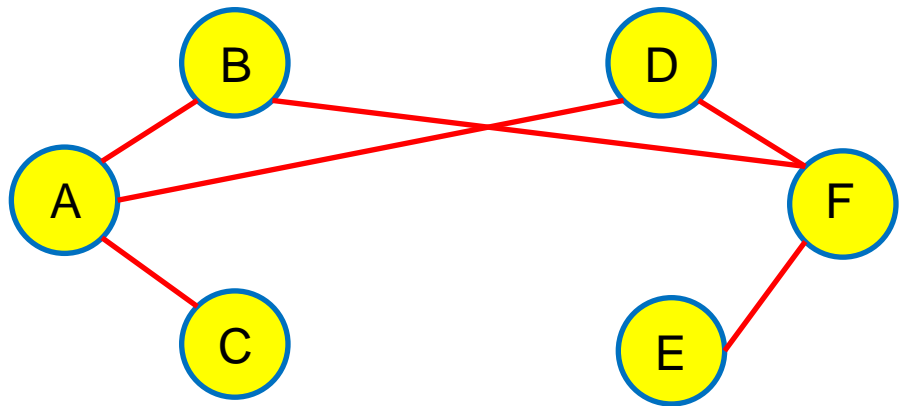
Vertex Cover Decision Problem

- A **vertex cover** in a graph G is a subset of vertices C that covers all edges in the graph.
- Formally, a **vertex cover** C is a set of vertices such that for each edge $\langle u, v \rangle$ in the graph, at least one of the vertices u or v is in C .

Vertex Cover Decision Problem:

- Is there a vertex cover of size k in this graph?
- Is there a vertex cover of size 2 in this graph?
- Which of the following is not a vertex cover of the Graph?

1. A, F
2. A, B, F
3. A, D, F
4. A, B, D
5. B, C, F

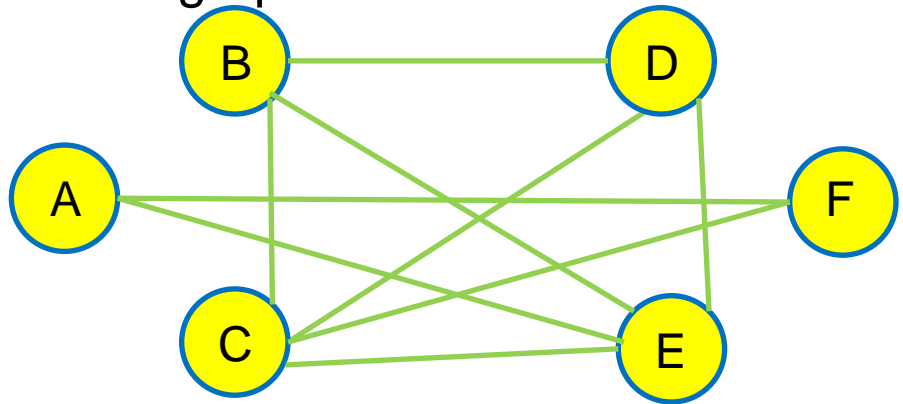


Clique Decision Problem

- A **clique** is a subset of vertices that are all connected to each other.
- Formally, a **clique** C is a subset of vertices that each pair of vertices u and v in C is connected to each other by an edge in the graph.

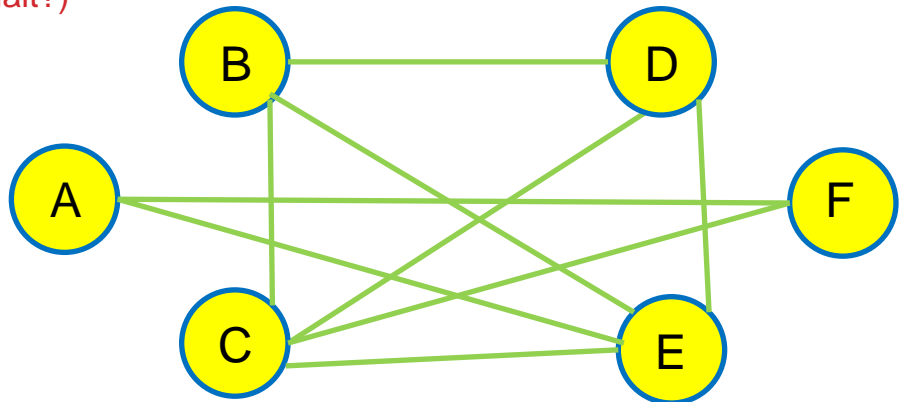
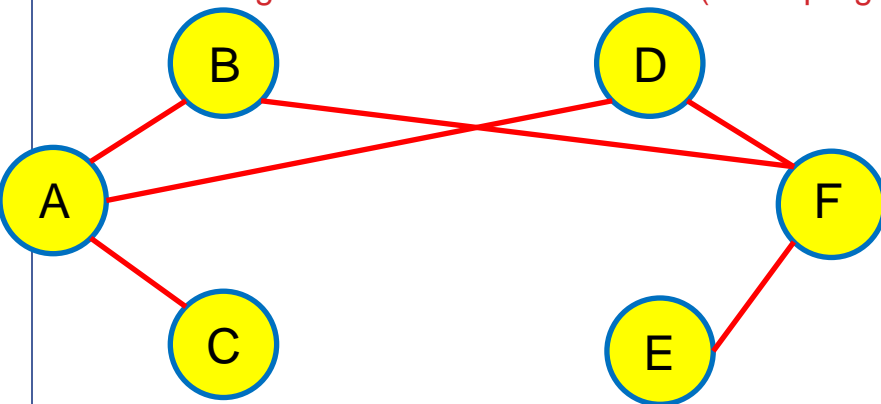
Clique Decision Problem:

- Is there a clique of size k in this graph?
- Is there a clique of size 4 in this graph?
- Which of the following is not a clique in this graph?
 1. A, F
 2. B, D, E
 3. B, C, D, E
 4. A, C, E, F
 5. B, C, D



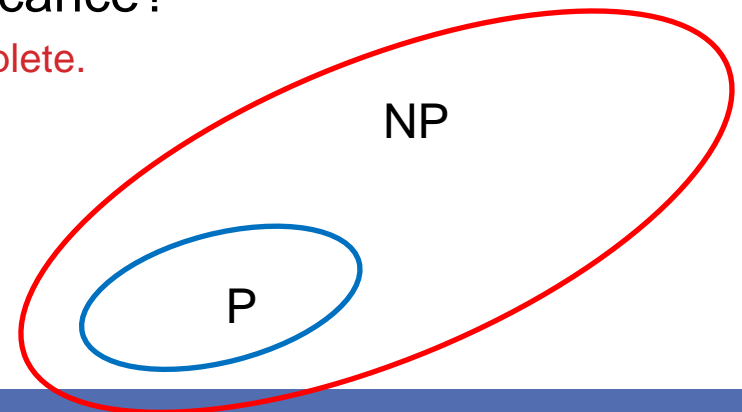
NP Class

- A **Certificate** (or witness) of a decision problem is an instance for which the answer to the problem is **YES**.
 - Is there a vertex cover of size 2 in the graph on left? The certificate is the set {A,F}.
 - Is there a clique of size 4 in the graph on right? The certificate is the set {B, C, D, E}
- A decision problem belongs to NP class if its certificate can be **verified** in polynomial time.
- Examples:
 - Vertex Cover is in NP class
 - Clique is in NP class
 - "Is there a path from s to t" is in NP class
 - Chess winner is not in NP class (Given a state of chess game, will the white player win?)
 - Halting Problem is not in NP class (Will a program halt?)



P Class

- A decision problem is in P class if it can be **solved** in polynomial time.
 - “Is there a path between s and t?” in P Class
 - ✦ A simple breadth-first search can check connectivity
 - Minimum Spanning Tree is in P Class
 - ✦ Is there a minimum spanning tree in the graph with weight W?
 - Compute MST using say Prim’s algorithm and check its weight.
 - Vertex Cover and Clique problems?
 - ✦ So far, we do not have any polynomial time solutions for these problems
 - ✦ And **we do not have a proof** that these problems **cannot** be solved in polynomial time
 - ✦ So these problems are in NP but we do not know whether these are in P or not
- A problem in P class is also in NP class
 - Because its certificate can be verified in polynomial time (by just solving the problem)
- But what is P vs NP and what’s its significance?
 - First, you will need to understand what is NP-Complete.



Polynomial time reduction

- We do not know the polynomial time solutions for Vertex Cover and Clique problems.
- Suppose we have a magic box that transforms (reduces) Vertex Cover problem to the Clique Problem and this reduction takes polynomial time.
- Assume that, in future, someone solves the clique problem in polynomial time.
- This means Vertex Cover can also be solved in polynomial time!
 - Step 1: Transform Vertex Cover to the Clique problem (in polynomial time)
 - Step 2: Solve Clique problem (in polynomial time)



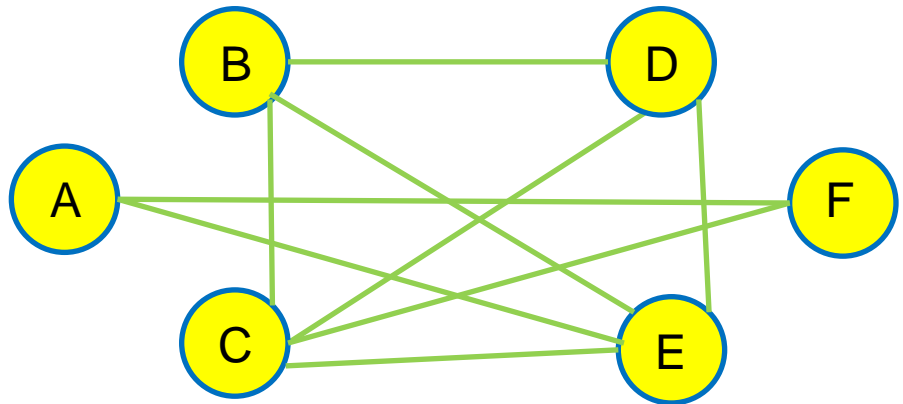
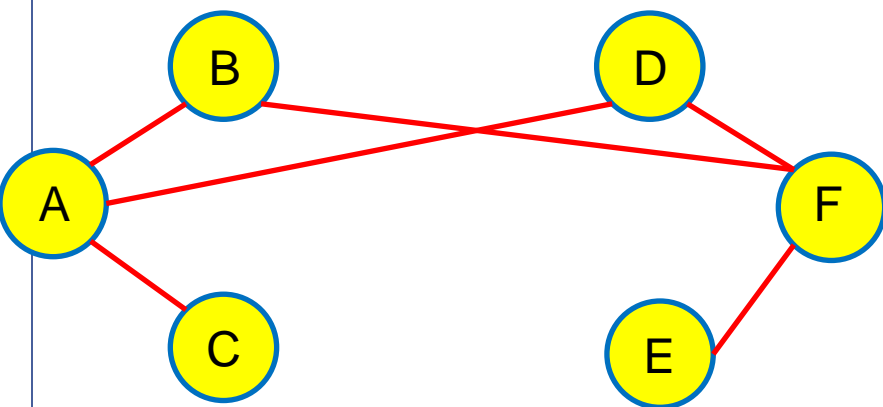
NP-Complete Class

- A problem X belongs to NP-complete class if
 - it is in NP class AND
 - **every** NP problem Y can be transformed (reduced) to X in polynomial time.
- Suppose we denote $Y \rightarrow X$ to show that Y can be reduced to X in polynomial time.
- If $Z \rightarrow Y$ and $Y \rightarrow X$ then $Z \rightarrow X$ (i.e., Z can be reduced to X in polynomial time).
- To show that a problem is NP-complete, all we need to do is to show that
 - Its certificate can be verified in polynomial time (i.e., it is in NP class)
 - **at least one** NP-complete problem can be reduced to it.
- Let Z be a problem known to be in NP-Complete and we want to prove that X is also in NP-complete. What do we need to prove? $Z \rightarrow X$, or $X \rightarrow Z$?
 - $Z \rightarrow X$



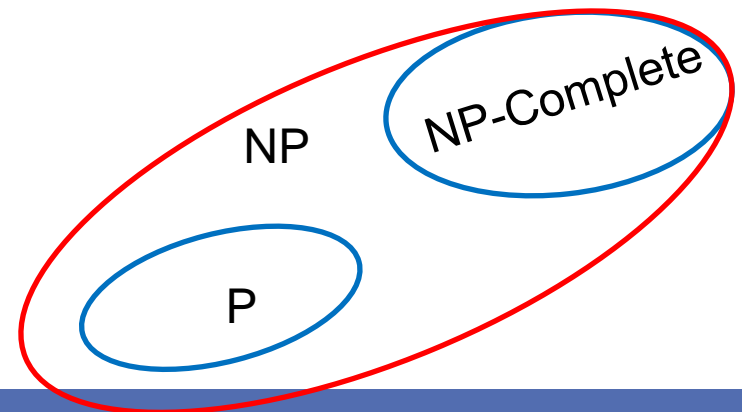
Example of polynomial time reduction

- Suppose we have proved that the problem of computing Clique is NP-Complete
- To prove that Vertex Cover is also NP-Complete, we need to show that
 - Its certificate can be verified in polynomial time
 - It can be reduced in polynomial time to at least one NP-Complete problem (e.g., Clique)
- Clique Problem: Is there a clique of size k in graph G ?
- Vertex Cover: Is there a vertex cover of size k in graph G ?
- Complement of a graph $G=(V,E)$ is a graph $G' = (V,E')$ that has the same vertices V as G and has an edge between two vertices u and v if and only if there is not edge between u and v in the graph G .
- Given a graph $G = (V,E)$, it has a vertex cover of size k if and only if its complement graph G' has a clique of size $N - k$ where N is the number of vertices.



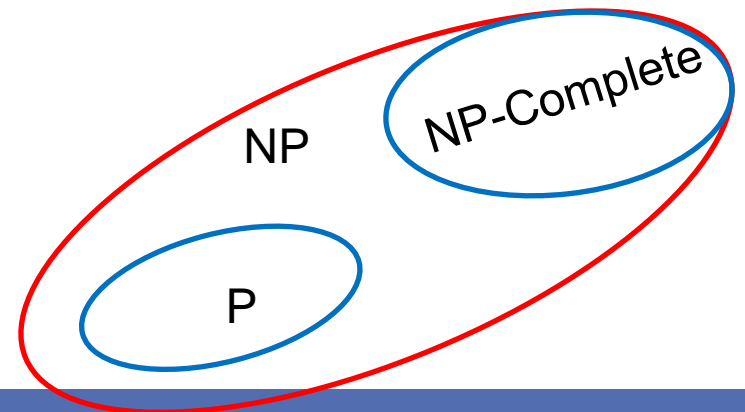
P vs NP problem

- If we can solve one NP-complete problem X in polynomial time, this means every NP problem can be solved in polynomial time.
 - Because every NP problem Y can be reduced to X in polynomial time
 - Then, X can be solved in polynomial time
- This would mean that $P = NP$
- Unfortunately, we are unable to solve any NP-Complete in polynomial time **AND** we are also unable to prove that a polynomial time solution is impossible
- Thus, we are not sure if $P = NP$ or $P \neq NP$
- This is called P vs NP problem



Consequences of solving P vs NP

- If $P = NP$ is proved
 - We will also be able to solve a lot of challenging (and very useful) problems really efficiently
 - On the other hand, cryptography is based on the challenging nature of factorization and will fail if $P = NP$ is proved
- If $P \neq NP$ is proved
 - A great theoretical leap
 - We will move towards finding alternate solutions to NP problems – we already are doing it since it is largely believed that $P \neq NP$



Primality Test

Prime:

- A prime number (or a prime) is a natural number greater than 1 that has NO positive divisors/factors other than 1 and itself.
- Primes in the first 100 natural numbers: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97
- Prime numbers have preoccupied human interest from time immemorial.

Primality Test:

- Given a number (arbitrarily long) as input, is this number a prime?
- Reliance on primes of modern crypto-systems (eg. RSA) makes primality testing an important algorithmic problem.
- From an algorithmic point-of-view, how fast/rapidly can we check if a given number n is prime?

Primality Test: Naïve Algorithm

#test if N is prime or not

for i = 2 to N-1:

if N % i == 0:

return "Composite!";

return "Prime!"

- The cost is $O(N)$ division operations
- Assuming that the input size is M denoting the number of digits in N . What will be the complexity in terms of number of division operations?
 - $M = \log N \rightarrow N = 2^M$. Hence, the total cost in terms of M is $O(2^M)$ division operations
- Can we change “for i = 2 to N-1” to “for i=2 to sqrt(N)”?
 - Yes.
 - The complexity would be $O(\sqrt{N})$ divisions or $O(2^{0.5M})$ division operations.

Fermat's Little Theorem

- The naïve algorithm on previous slide is exponential in M .
- Fermat's Little Theorem provides a polynomial time “approximate” algorithm

Fermat's Little Theorem:

- If p is a **prime number**, then for any integer $1 < a < p$, $a^{p-1} - 1$ is an integer multiple of p , i.e., $(a^{p-1} - 1) \% p = 0$.
 - E.g., $p = 11$ and $a = 2$, then $2^{10} - 1 = 1023$ and $1023 \% 11 = 0$.
- Thus we can guarantee that N is composite if $(a^{N-1} - 1) \% N \neq 0$.
- However, N , $(a^{N-1} - 1) \% N = 0$ does not guarantee that N is prime, i.e., this may also hold if N is composite.
 - E.g., $n = 221$ and $a = 38$, $(38^{220} - 1) \% 221 = 0$ but 221 is not prime ($221 = 17 \times 13$)
 - Such value of a is called Fermat's liar
- Thus, Fermat's primality test may return incorrect results

Fermat's Primality Test

#test if N is prime or not

Repeat k times:

a = any random number between 2 and N-1

if $(a^{N-1} - 1) \% N \neq 0$:

return "Composite!";

return "Probably Prime!"

- The above algorithm does not guarantee correct result
- The first exact polynomial time algorithm for primality test was introduced in 2002 (called AKS). Prior to this, primality test problem was not known to be in P class.

Design Principles (Summing up FIT2004)

Here are some broad strategies to (try to) solve algorithmic problems:

- Look out for good invariants to exploit
- Attempt to balance your work as much as possible
- Do not repeat work (so, store and re-use!)
- Use appropriate data representations
- Try well-known problem solving strategies
- Sometimes greed is good!
- These are general guidelines. As always, there are many exceptions

Look out for good variants to exploit

- Here are **some** algorithms we considered in the unit that do precisely this!
- Binary Search (Refer Week 2 lecture)
- Sorting (Refer Lectures from Weeks 2 and 3)
- Shortest Paths and Connectivity
 - Dijkstra's algorithm (Refer Week 8 Lectures)
 - Warshall's and (related) Floyd's algorithm (Refer Week 9 lectures)
- Minimum Spanning Tree Algorithms (Refer Week 10 lectures)

Balance your work as much as possible

- For problems that allow division of labour (eg. Divide and Conquer)
- Try to divide work **equally** as much as possible
- Merge sort achieves this (Refer week 3)
 - $(N \log N)$ -time always!
- Quick sort does not necessarily achieve this – depends on the choice of the median (Refer week 3)
 - Good estimates of the median give $O(N \log N)$ -time
 - Bad estimates of the median give $O(N^2)$ -time
- Hirschberg's algorithm (Refer Week 4)

Choose Data Structures with care

- Certain data representations are more efficient than others for a given problem
- Priority Queue in Dijkstra's algorithm (Refer Week 8)
- Union-Find data structure in Kruskal's algorithm (refer Week 9)
- Efficient Search and retrieval data structures of various kinds (Refer Weeks 5,6,7 lectures)

Don't repeat work

- Do not compute anything more than once (if there is room to store it for reuse)
- Underpins Dynamic Programming strategy
 - Edit Distance (Refer Week 4 Lecture)
 - Largest Common Subsequence problem (Refer question 1 in Week 7 tute sheet)
 - Dijkstra's Algorithm (Refer Week 8 Lectures)

Try well known problem solving strategies

- Divide and Conquer (Refer Weeks 3, 4 lectures)
- Dynamic Programming (Refer Weeks 4, 8, 9 lectures)

Sometimes greed is good

- A **greedy strategy** is to make a “local” choice based on current information
- Sometimes gives optimal solution, e.g.
 - Dijkstra’s single source shortest paths algorithm (Refer Week 8 Lectures)
 - Minimum Spanning Tree Algorithms – Prim’s and Kruskal’s (Refer Week 10 lectures) minimum spanning tree algorithm.
- **Greedy is sometimes a good heuristic!**
 - Sometimes gives a “good” solution to a (combinatorial) problem even if not guaranteed optimal

Programming Competition Winners

- **Round 1**
 - 1st: Khao Phan Anh Tran
 - 1st: Wenyu Zhao
 - 1st: Alex Phu Hung Ong
- **Round 2**
 - 1st: Weiwei Lin
 - 2nd: Khao Phan Anh Tran
 - 3rd: Benjamin King-Hunn Siow
- **Round 3**
 - 1st: Khao Phan Anh Tran
 - 2nd: Yue Chen
 - 3rd: Weiwei Lin
- **Overall**
 - 1st: Khao Phan Anh Tran
 - 2nd: Weiwei Lin
 - 3rd: Benjamin King-Hunn Siow

Final Exam

- Time allowed: 3 hours + 10 minutes reading time
- Total Marks: 70
- Exam is **NOT** open book
- Question style similar to past sample exam uploaded – but do not try to guess what will be on the exam
- Do not miss final exam even if you fail in-semester hurdle.
 - It affects your WAM
- I have two news for you: one is good, one is bad! Which one you want to hear first?
- **Bad News:** Exam is set in a way that if you prepare only selective topics, you will struggle! Study thoroughly.
- **Good News:** P vs NP is not examinable + exam questions do not ask you to write algorithms that you have not seen in lectures, tutorials, or pracs!

Algorithm for exam preparation

1. **for** $i=1$ to 12 :
2. go through lecture slides week i :
3. **if** something **is** unclear:
4. listen MULO recording # slides are optimized for lectures. Just reading the slides might not be enough
5. **if** something **is still** unclear:
6. seek help
7. **for** i in range($3,12,2$):
8. attempt tutorial questions week i
9. **for** each question attempted:
10. compare your solution **with** the sample solution
11. seek help **if** something **is** unclear
12. **for** $i=1$ to 12 :
13. write pseudocode **for** questions in prac week i
14. **if in** doubt:
15. post your pseudocode on Moodle forum
16. **for** each question on the sample exam:
17. attempt the question
18. post your solution on Moodle forum
19. If $\text{current_date} \leq \text{exam_date}$
20. goto line 1

Consultations for Final Exam

- Starting from next week, (almost) daily consultation sessions will be provided by me or some of the tutors. There might be some minor changes to the schedule (always check latest schedule on Moodle)

Staff	Date	Time	Location
Aamir Cheema	Monday, 24-Oct-2016	4-5 pm	Room 113, 25 Exhibition Walk
Han Phan	Tuesday, 25-Oct-2016	1-2 pm	TBA
Aamir Cheema	Wednesday, 26-Oct-2016	3-4 pm	Room 113, 25 Exhibition Walk
Aamir Cheema	Thursday, 27-Oct-2016	4-5 pm	Room 113, 25 Exhibition Walk
Aamir Cheema	Friday, 28-Oct-2016	2-3 pm	Room 113, 25 Exhibition Walk
Chaluka Salgado	Tuesday 01-Nov-2016	11-12 pm	Room G20, 14 Rainforest Walk
Ammar Sohail	Wednesday, 02-Nov-2016	11-12 pm	Room G20, 14 Rainforest Walk
Ammar Sohail	Thursday, 03-Nov-2016	11-1 pm	Room G20, 14 Rainforest Walk
Chaluka Salgado	Friday, 04-Nov-2016	11-1 pm	Room G20, 14 Rainforest Walk
Aamir Cheema	Monday, 07-Nov-2016	3-5 pm	Room G20, 14 Rainforest Walk
Aamir Cheema	Tuesday, 08-Nov-2016	11- 12 pm	Room G20, 14 Rainforest Walk

Consultations for Final Exam

- Don't come to consultations in a hope to squeeze some useful information about final exam
 - **Important:** Last night I got an accident and have lost my memory – so do not waste your energy
 - **Very Important:** In many movies, a second accident brings the memory back. Please do not pray that I get another accident.

Reward for attending lectures

- To be given in class
- Please do not tell others (those who did not attend the lecture) what kind of reward you got – let them wonder :P
- If someone insists, just tell them something random so that they think they know what you got
 - E.g., you can say we were given chocolates

Some Thoughts

- Don't study for marks, study to learn
 - ✦ You may get a job if you have exceptionally good marks
 - ✦ You will get a job and will never lose it if you have exceptionally good skills
 - ✦ Self satisfaction is more important than marks
- Moral: **Knowledge is far more important than marks ... seek knowledge!**
- Try to learn something about everything and everything about something - **Thomas Henry Huxley**
- Do what you enjoy! Enjoy what you do!
- Do not hesitate to contact me if I can be of help (even after this semester)
 - ✦ E.g., I will be happy to write reference letter if you did well in the unit or the programming competition

Summary

Take Home Message

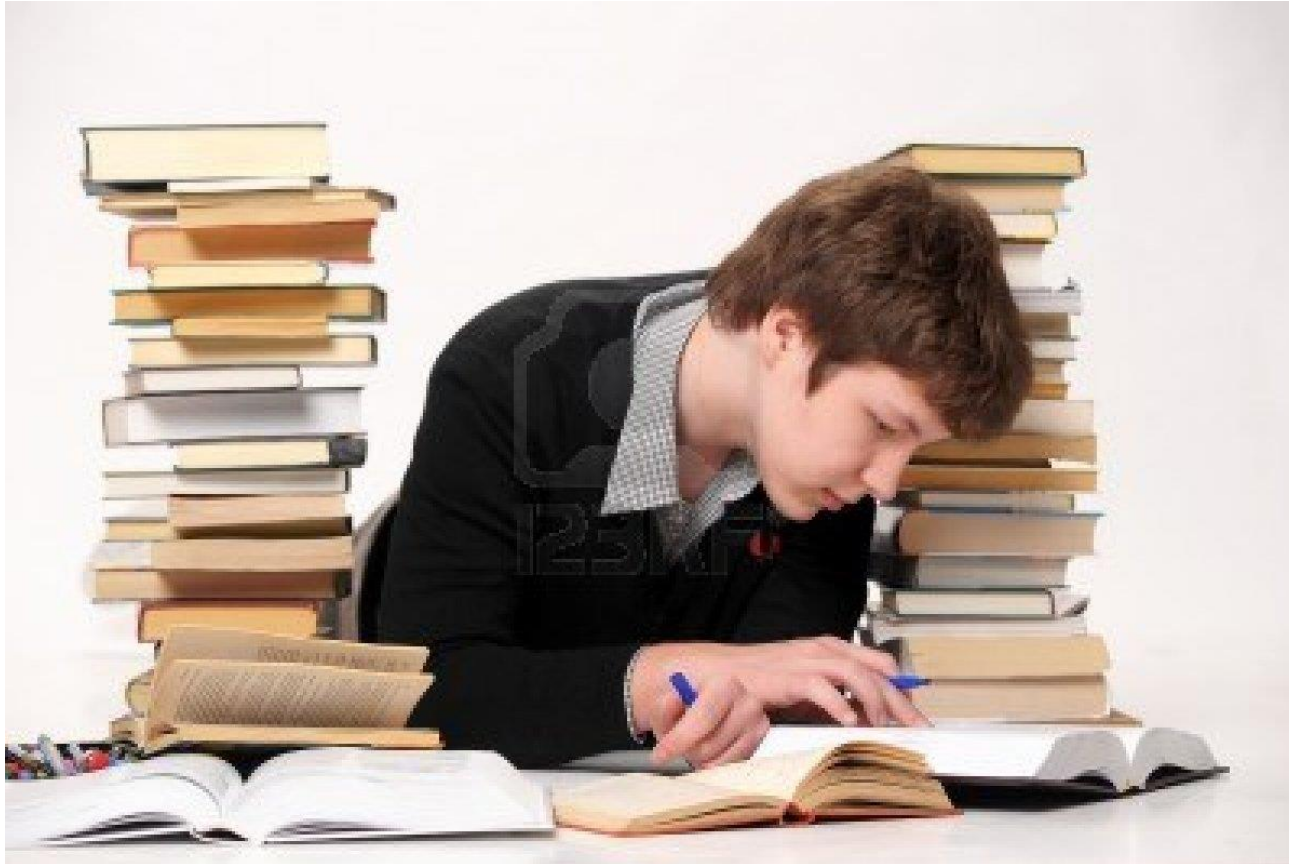
- See Design Principles in this lecture

Things to do (this list is not exhaustive)

- See the algorithm to prepare for final exam

Coming Up Next

Coming Up Next



SWOT VAC

Coming Up Next



Final Exam

Coming Up Next



Results Day

And you live happily ever after

Coming Up Next (Second Scenario)



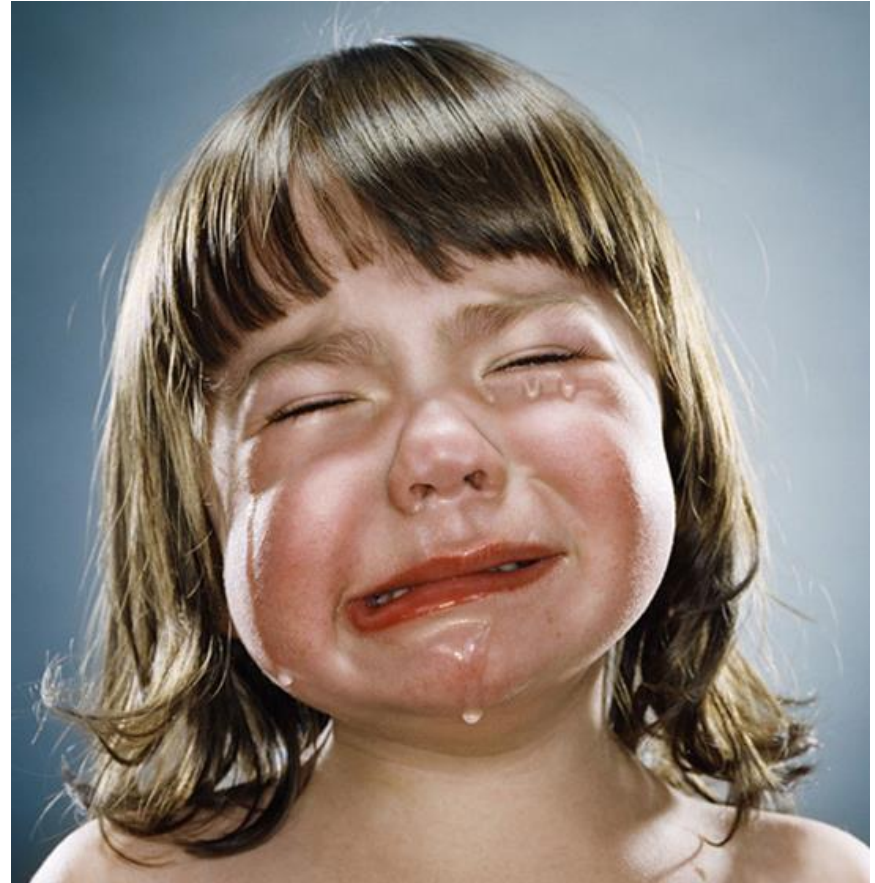
SWOT VAC

Coming Up Next (Second Scenario)



Final Exam

Coming Up Next (Second Scenario)



Results Day

Moral: Work hard or be brave!

That's all folks!



Wish You All the Best For Your Future