University of Glasgow

32nd IEEE International Conference on
Software Engineering Education & Training
November 9-12, Munich, Germany

Keeping software engineering students in touch with not only *what* they are to learn, but with *why*

S Waqar Nabi

Joseph Maguire

Steve Draper

Quintin Cutts

60 YEARS OF COMPUTING AT GLASGOW

Joseph Maguire

Steve Draper

Quintin Cutts

Origins...

Origins

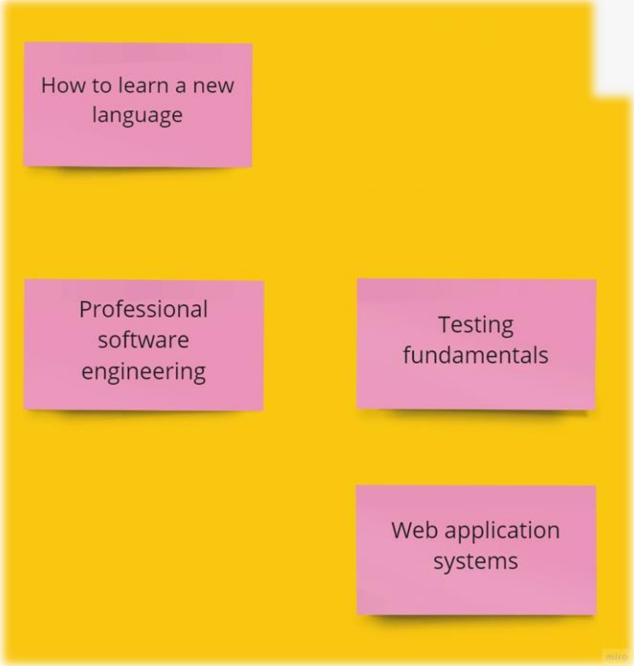
GRADUATE APPRENTICESHIPS

BSc (Hons) Software Engineering

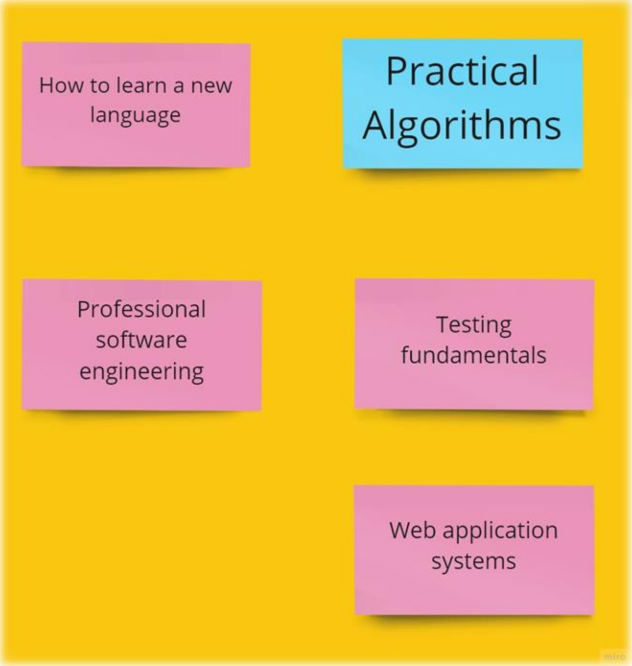
Work in partnership with a world leading university to upskill your existing team or recruit new talent through our fully funded Graduate Apprenticeship (GA) Degree in Software Engineering.

Key features:

- Employers recruit apprentices directly
- Apprentices achieve a BSc in Software Engineering in 4 years
- Programme structure: 20% study; 80% work-based learning



Origins



GRADUATE APPRENTICESHIPS

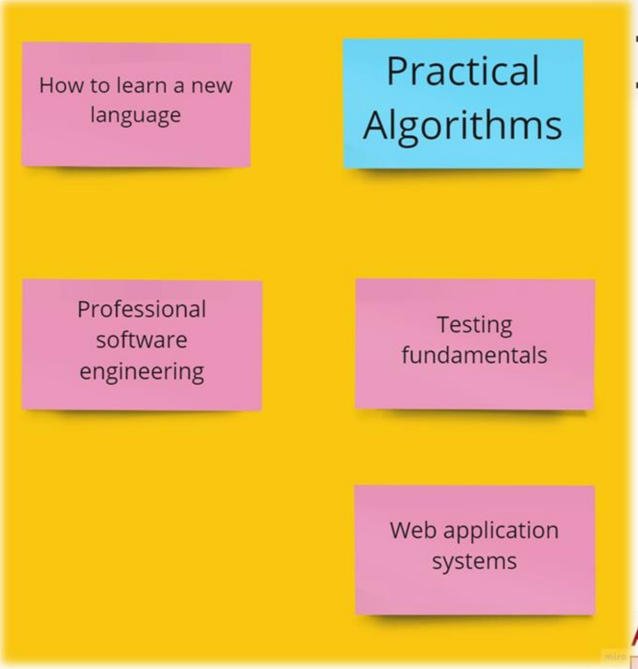
BSc (Hons) Software Engineering

Work in partnership with a world leading university to upskill your existing team or recruit new talent through our fully funded Graduate Apprenticeship (GA) Degree in Software Engineering.

Key features:

- Employers recruit apprentices directly
- Apprentices achieve a BSc in Software Engineering in 4 years
- Programme structure: 20% study; 80% work-based learning

Origins



GRADUATE APPRENTICESHIPS

BSc (Hons) Software Engineering

Work in partnership with a world leading university to upskill your existing team or recruit new talent through our fully funded Graduate Apprenticeship (GA) Degree in Software Engineering.

Key features:

- Employers recruit apprentices directly
- Apprentices achieve a BSc in Software Engineering in 4 years
- Programme structure: 20% study; 80% work-based learning

Algorithms and Data Structures

A1	Introduction to Data Structures and Algorithms
A2	Algorithmic Analysis techniques
A3	Recursion
A4	Sorting Algorithms
A5	Linked Lists
A6	Abstract Data Types (ADTs)
A7	Trees
A8	Hash Tables
A9	Advanced Topics in Algorithmic Design

Discrete Mathematics

B1	Introduction to Discrete Maths / Algorithmic Foundations
B2	Propositional Logic
B3	Predicated and Quantifiers
B4	Sets, Functions, Countability
B5	Sequences, Summations, Integers
B6	Methods of Proof / Rules of Inference
B7	Induction and Recursive Definitions
B8	Counting
B9	Probabilty
B10	Graphs
B11	Relations

Origins

- How to learn a new language
- Practical Algorithms
- Professional software engineering
- Testing fundamentals
- Web application systems

GRADUATE APPRENTICESHIPS

BSc (Hons) Software Engineering

Work in partnership with a world leading university to upskill your existing team or recruit new talent through our fully funded Graduate Apprenticeship (GA) Degree in Software Engineering.

Key features:

- Employers recruit apprentices directly
- Apprentices achieve a BSc in Software Engineering in 4 years
- Programme structure: 20% study; 80% work-based learning

Algorithms and Data Structures

A1	Introduction to Data Structures and Algorithms
A2	Algorithmic Analysis techniques
A3	Recursion
A4	Sorting Algorithms
A5	Linked Lists
A6	Abstract Data Types (ADTs)
A7	Trees
A8	Hash Tables
A9	Advanced Topics in Algorithmic Design

Discrete Mathematics

B1	Introduction to Discrete Maths / Algorithmic Foundations
B2	Propositional Logic
B3	Predicated and Quantifiers
B4	Sets, Functions, Countability
B5	Sequences, Summations, Integers
B6	Methods of Proof / Rules of Inference
B7	Induction and Recursive Definitions
B8	Counting
B9	Probabilty
B10	Graphs
B11	Relations

Origins

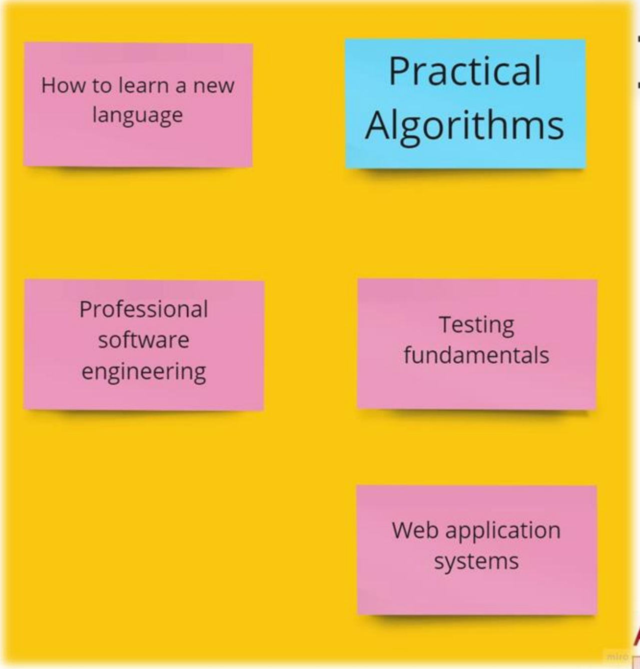
GRADUATE APPRENTICESHIPS

BSc (Hons) Software Engineering

Work in partnership with a world leading university to upskill your existing team or recruit new talent through our fully funded Graduate Apprenticeship (GA) Degree in Software Engineering.

Key features:

- Employers recruit apprentices directly
- Apprentices achieve a BSc in Software Engineering in 4 years
- Programme structure: 20% study; 80% work-based learning



Algorithms and Data Structures

A1	Introduction to Data Structures and Algorithms
A2	Algorithmic Analysis techniques
A3	Recursion
A4	Sorting Algorithms
A5	Linked Lists
A6	Abstract Data Types (ADTs)
A7	Trees
A8	Hash Tables
A9	Advanced Topics in Algorithmic Design

Discrete Mathematics

B1	Introduction to Discrete Maths / Algorithmic Foundations
B2	Propositional Logic
B3	Predicated and Quantifiers
B4	Sets, Functions, Countability
B5	Sequences, Summations, Integers
B6	Methods of Proof / Rules of Inference
B7	Induction and Recursive Definitions
B8	Counting
B9	Probabilty
B10	Graphs
B11	Relations

- What is the narrative here?
- How do these topics connect to each other?
- Why is this useful for my students?
- If I have these questions, most likely my students will too.

Outline:
Keeping software engineering students in touch with not only ***what*** they are to learn, but with *why*

Why is *why* important (*meta-whyness*)

Identifying the challenges

Towards addressing the challenges: using *concept maps*

The importance of *why*

“Meta-whyness”

The goal:
student engagement

Why do students disengage?

Disengagement dynamics: **Student Motivation**

- Encouraging self-motivation: Focus on ***concepts*** rather than ***facts:***
  - Leads to: "Self-directed construction of knowledge structures for deep and sustainable learning"[1].

- Some students may perceive theory as not relevant to their profession (although employers value it)[2].

- Temporal aspect
  - The answer to "why am I learning this" may come many years later

- Last but not the least: Students may not all share the same motivations.
  - Student may not share the "why"

Disengagement dynamics: Structure

- Having little sense of where to begin
  - or how to proceed through the steps

Outline:
Keeping software engineering students in touch with not only ***what*** they are to learn, but with *why*

Why is *why* important (*meta-whyness*)

Identifying the challenges

Towards addressing the challenges: using *concept maps*

Identifying the Challenges

How do we provide a meaningful and <u>motivated</u> learning experience for students in a Work-Based Learning (WBL) degree programme.

Student vs Teacher Perspective

Teacher Perspective:
Why teach something

- Work-based learning vs Higher education
  - Students' and employers' expectation:
    - complement workplace
    - address workplace requirements
  - Higher education perspective
    - *broad-based* education, *foundational concepts*
    - academic *rigour* and (considerable) *theory*

Teacher Perspective:
Why teach something

- Work-based learning vs Higher education
  - Students' and employers' expectation:
    - complement workplace
    - address workplace requirements
  - Higher education perspective
    - *broad-based* education, *foundational concepts*
    - academic *rigour* and (considerable) *theory*

- Applied skills vs Universal truths
  - Applied skills: Extrinsically motivated, short to medium term focus, focus on *skills*.
  - Universal truths: Intrinsically motivated focus on axioms, theories, laws, concepts etc (long shelf life)

Student Perspective:
Why learn something

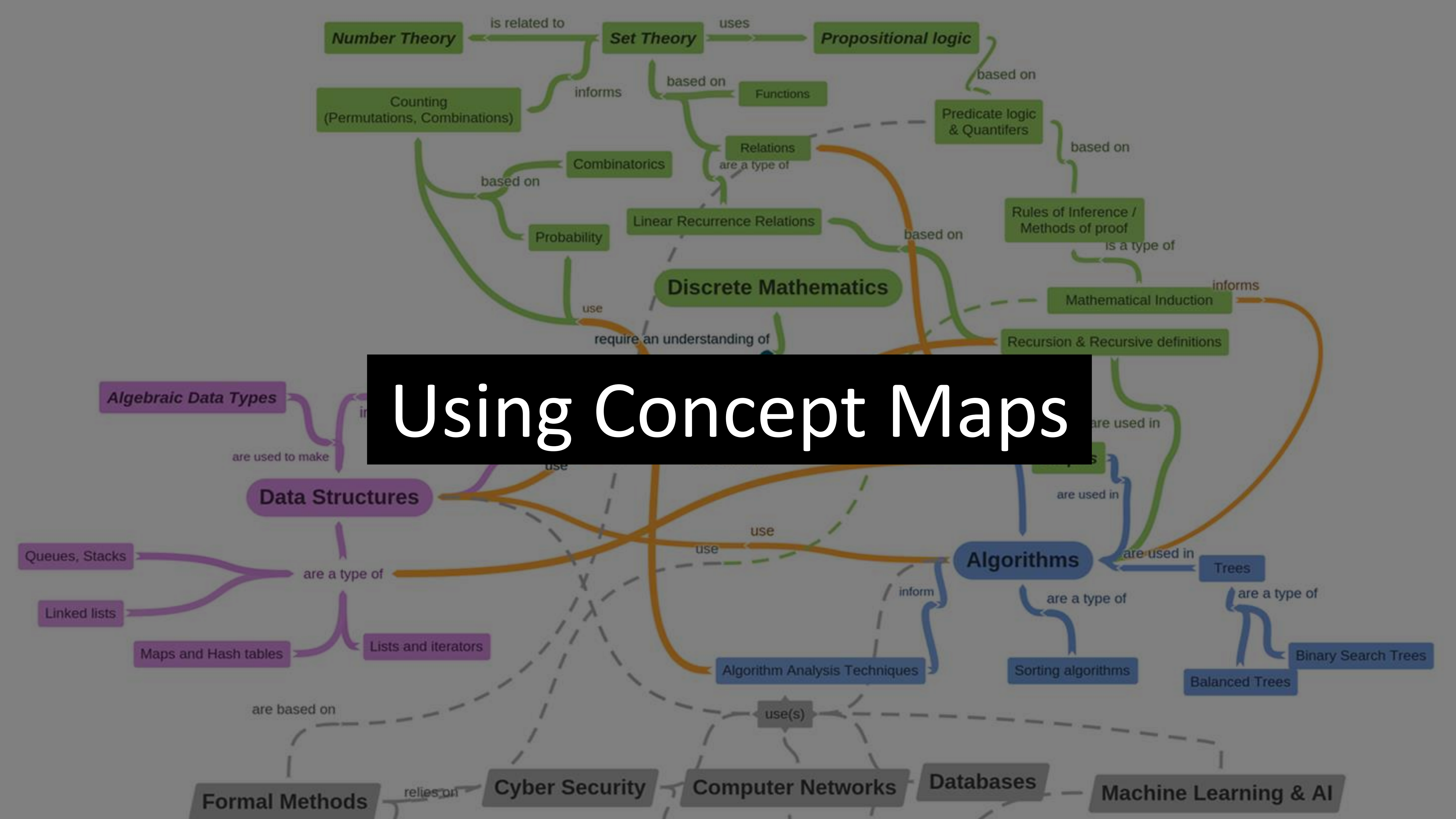
- The *temporal* aspect
  - WBL students may (to some extent, should) prioritize immediate relevance of learning
  - However, some knowledge may become relevant a lot later.

- Personal view points
  - There is a “normal” variation of prior knowledge and motivations in students in the general case
  - For WBL students working as apprentices, the workplace is adds another dimension to the variation

Outline:
Keeping software engineering students in touch with not only ***what*** they are to learn, but with *why*

- Why is *why* important (*meta-whyness*)
- Identifying the challenges
- Towards addressing the challenges: using *concept maps*

Using Concept Maps



The course: “Practical Algorithms”

Practical Algorithms: Course Outline

Algorithms and Data Structures

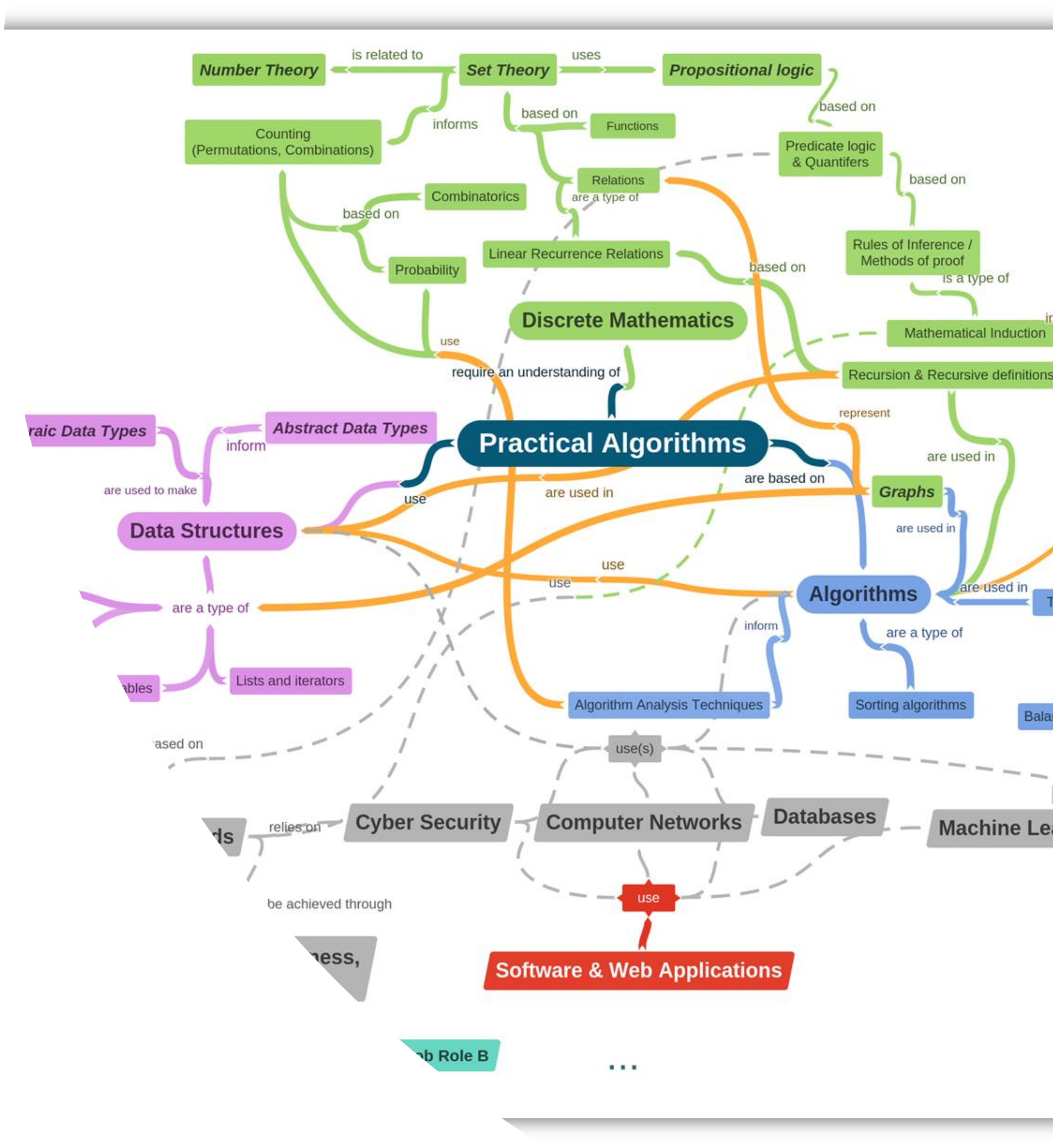
A1	Introduction to Data Structures and Algorithms
A2	Algorithmic Analysis techniques
A3	Recursion
A4	Sorting Algorithms
A5	Linked Lists
A6	Abstract Data Types (ADTs)
A7	Trees
A8	Hash Tables
A9	Advanced Topics in Algorithmic Design

Discrete Mathematics

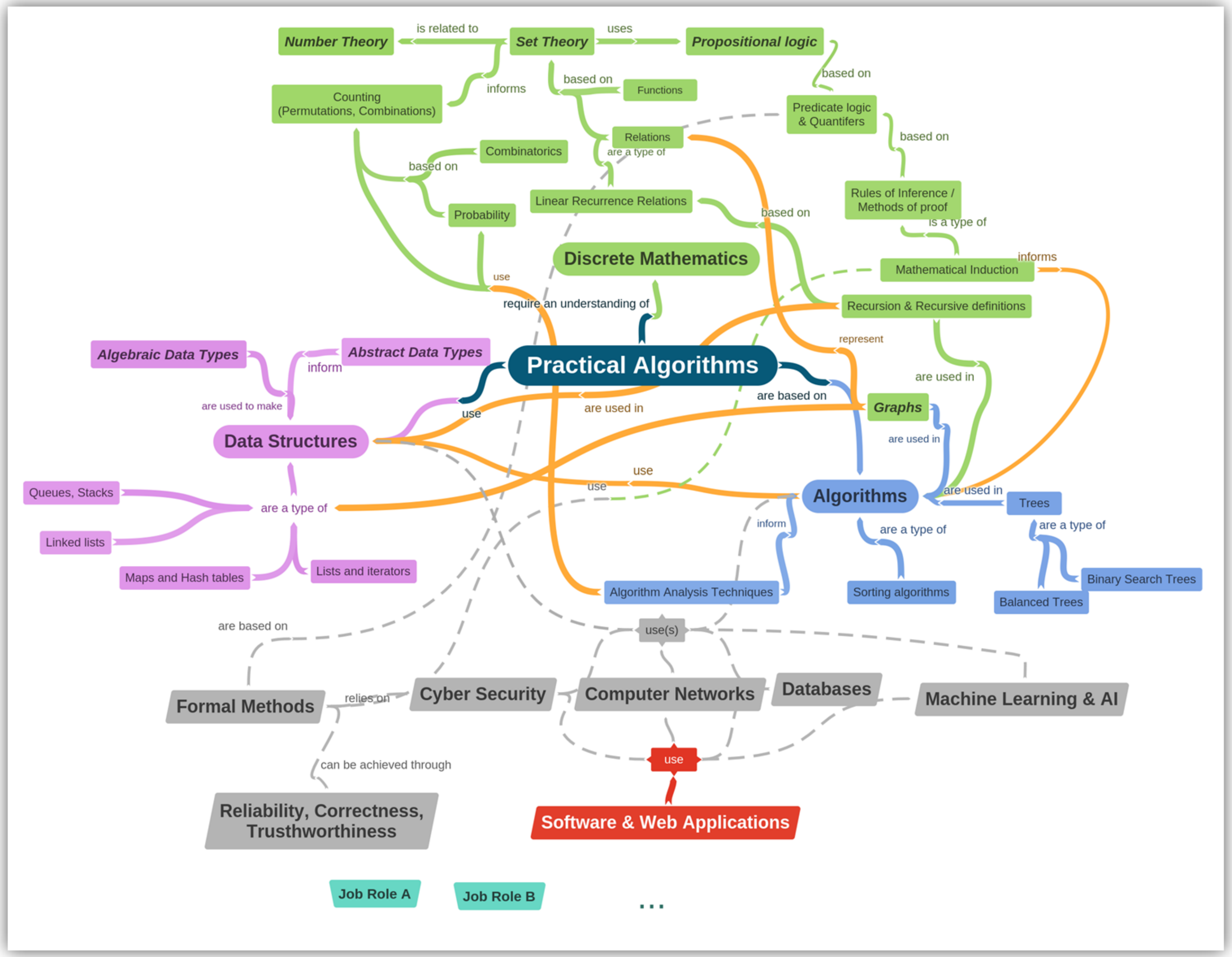
B1	Introduction to Discrete Maths / Algorithmic Foundations
B2	Propositional Logic
B3	Predicated and Quantifiers
B4	Sets, Functions, Countability
B5	Sequences, Summations, Integers
B6	Methods of Proof / Rules of Inference
B7	Induction and Recursive Definitions
B8	Counting
B9	Probabilty
B10	Graphs
B11	Relations

One proposal: Use concept maps

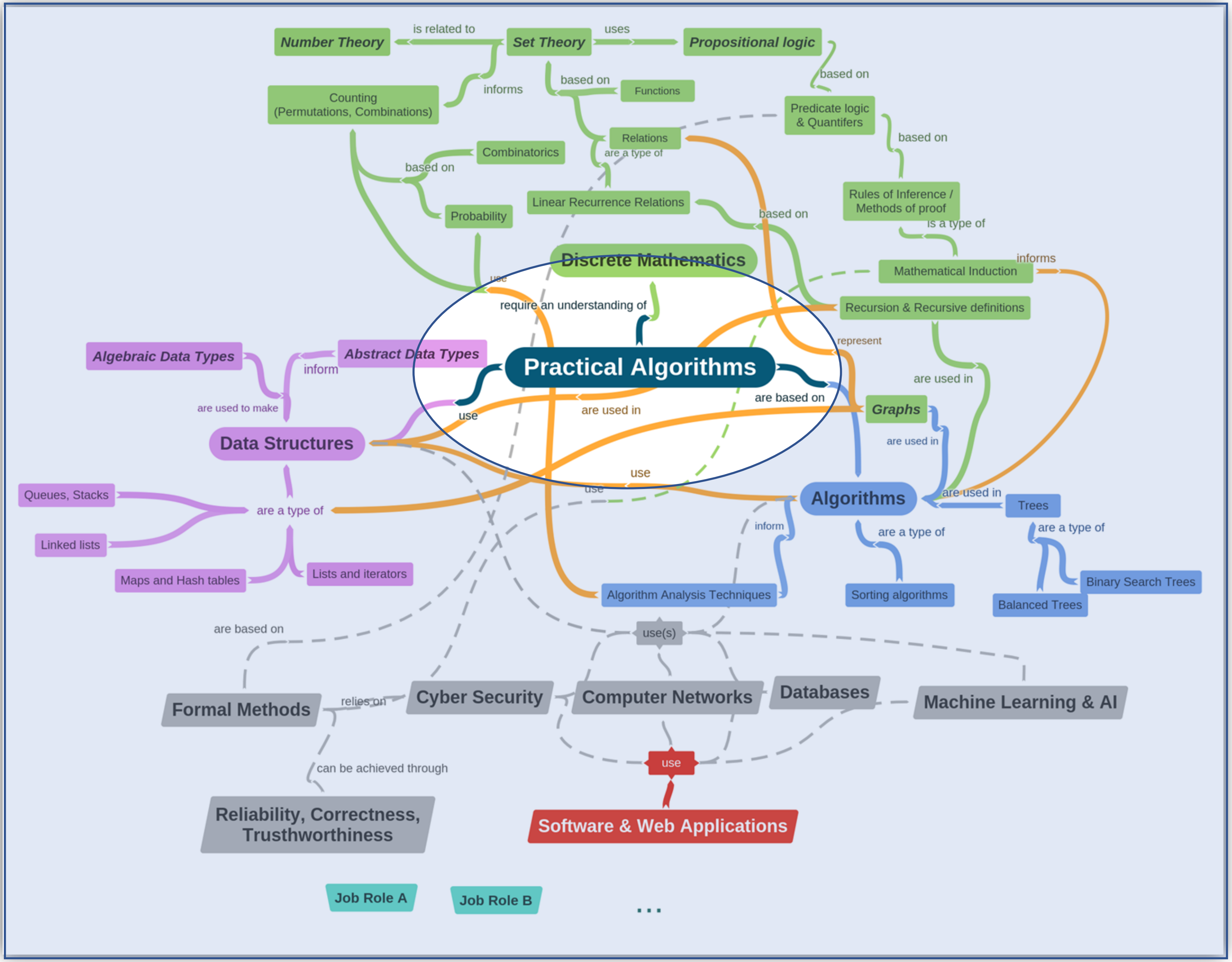
- A graphical tool that is useful in illustrating the relationships between concepts
- Developed by Novak in the 1970s as an aid to understanding and following changes in childrens' understanding of science
- Based on the cognitive development theory of *subsumption*
    - learning takes place by *assimilation* of new concepts into an existing framework and concepts



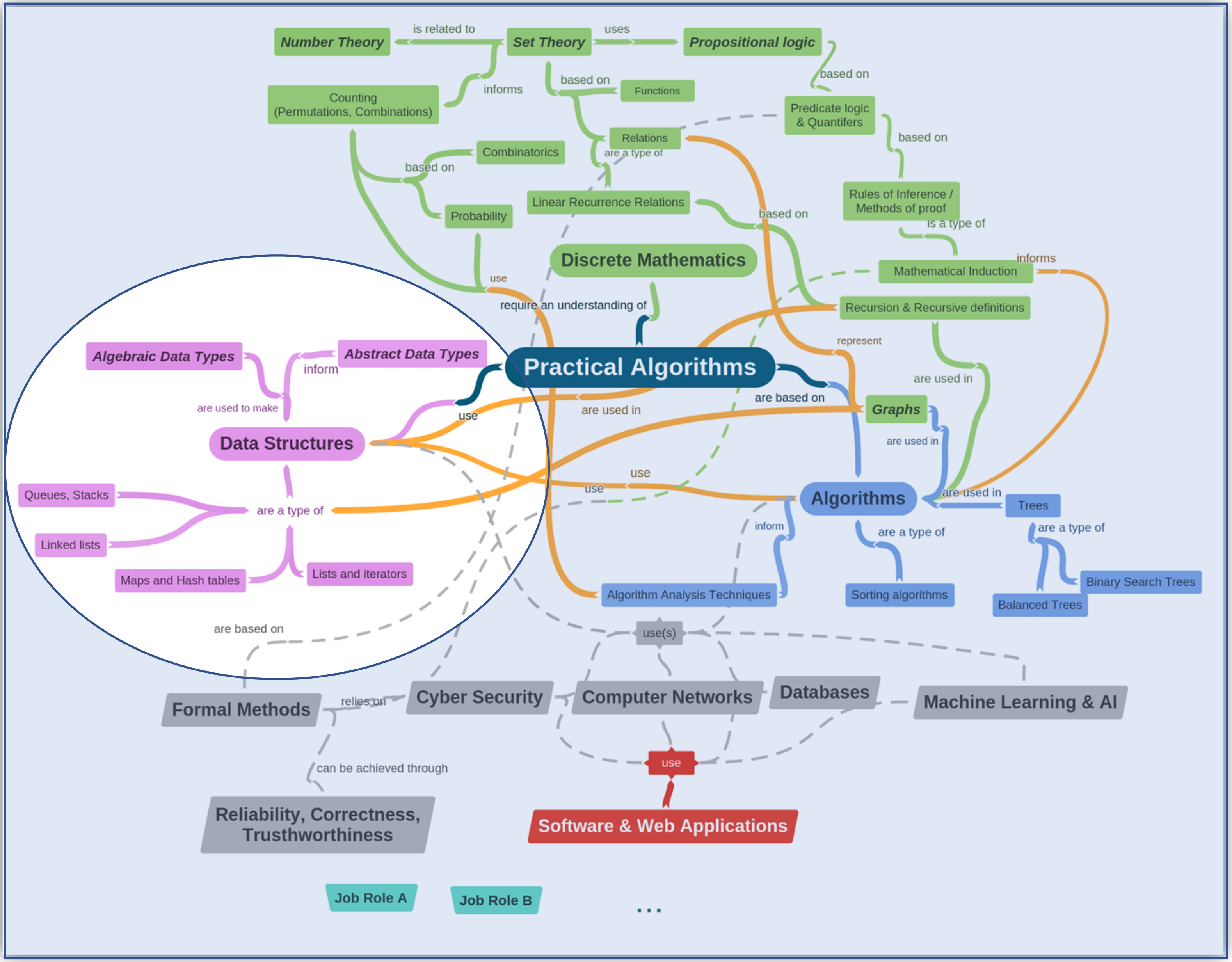
The “Practical Algorithms” Concept Map



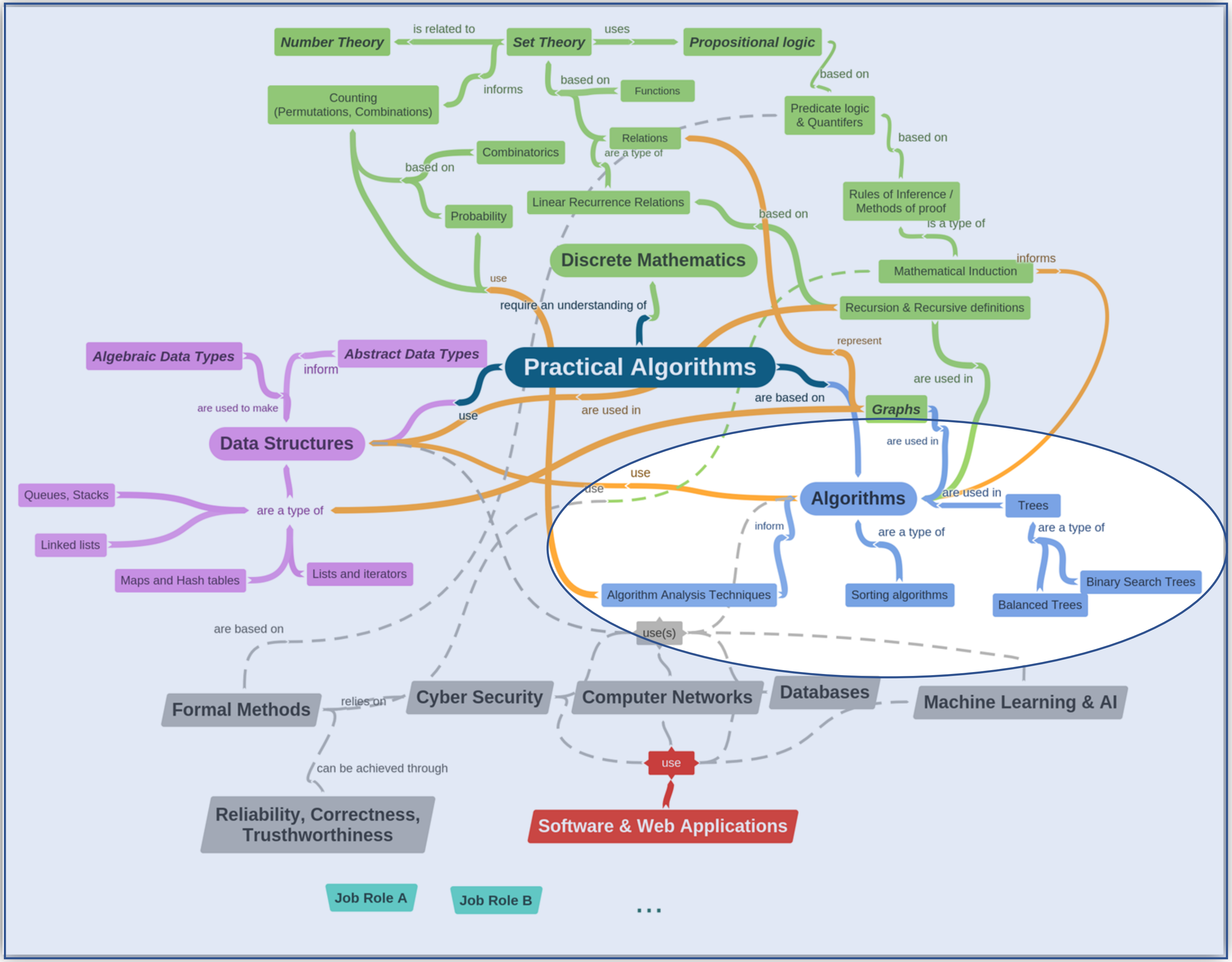
The "Practical Algorithms" Concept Map



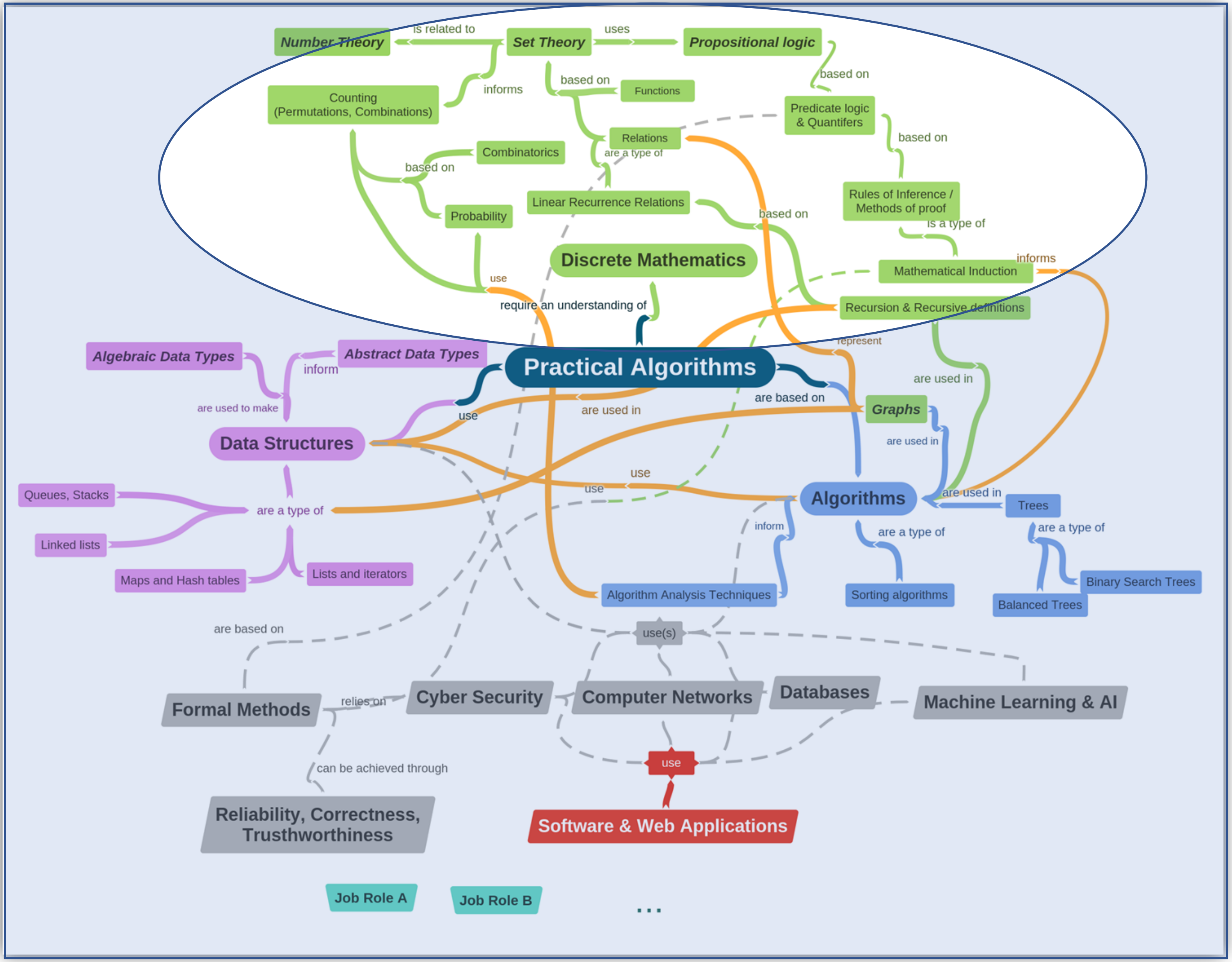
The “Practical Algorithms” Concept Map



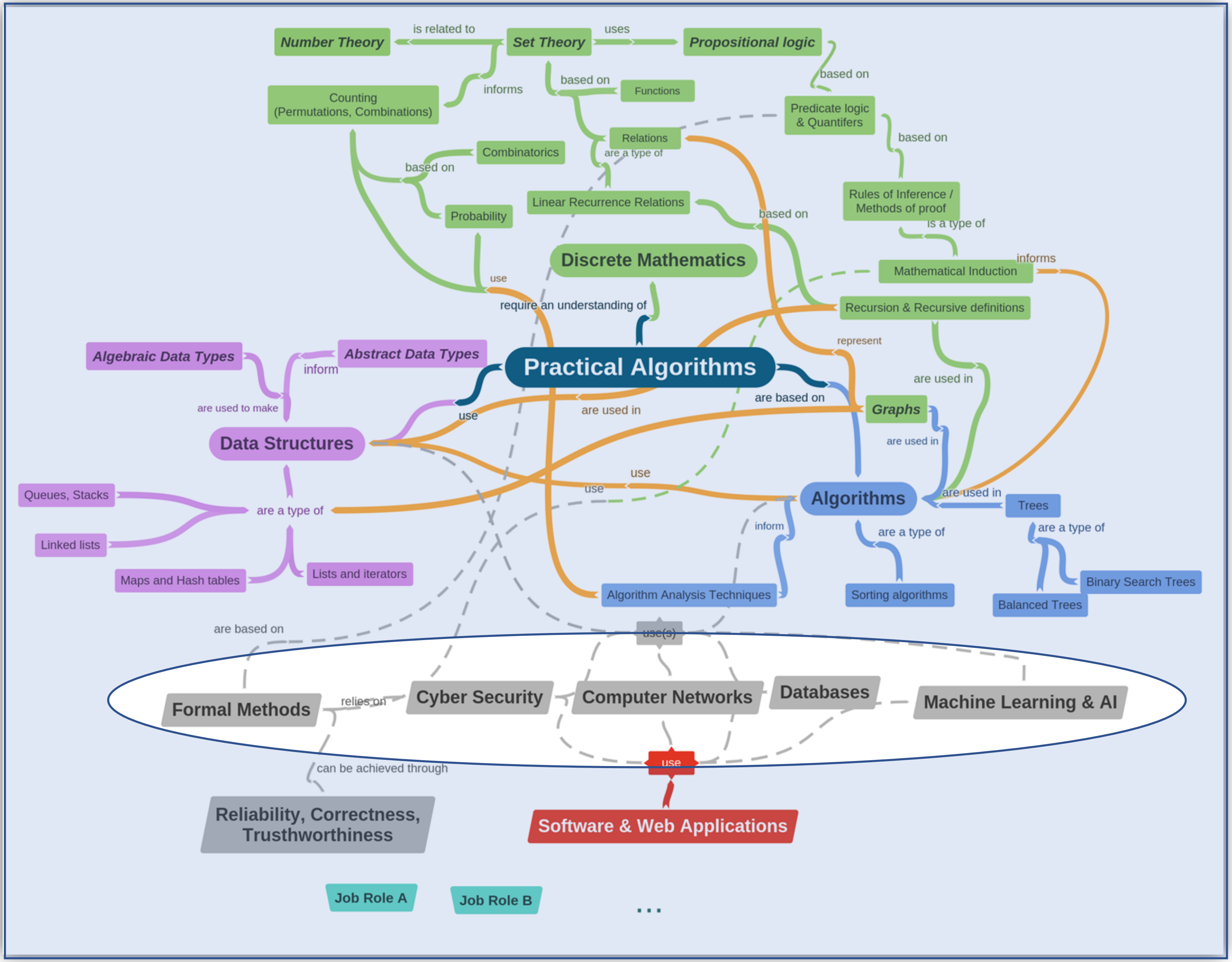
The "Practical Algorithms" Concept Map



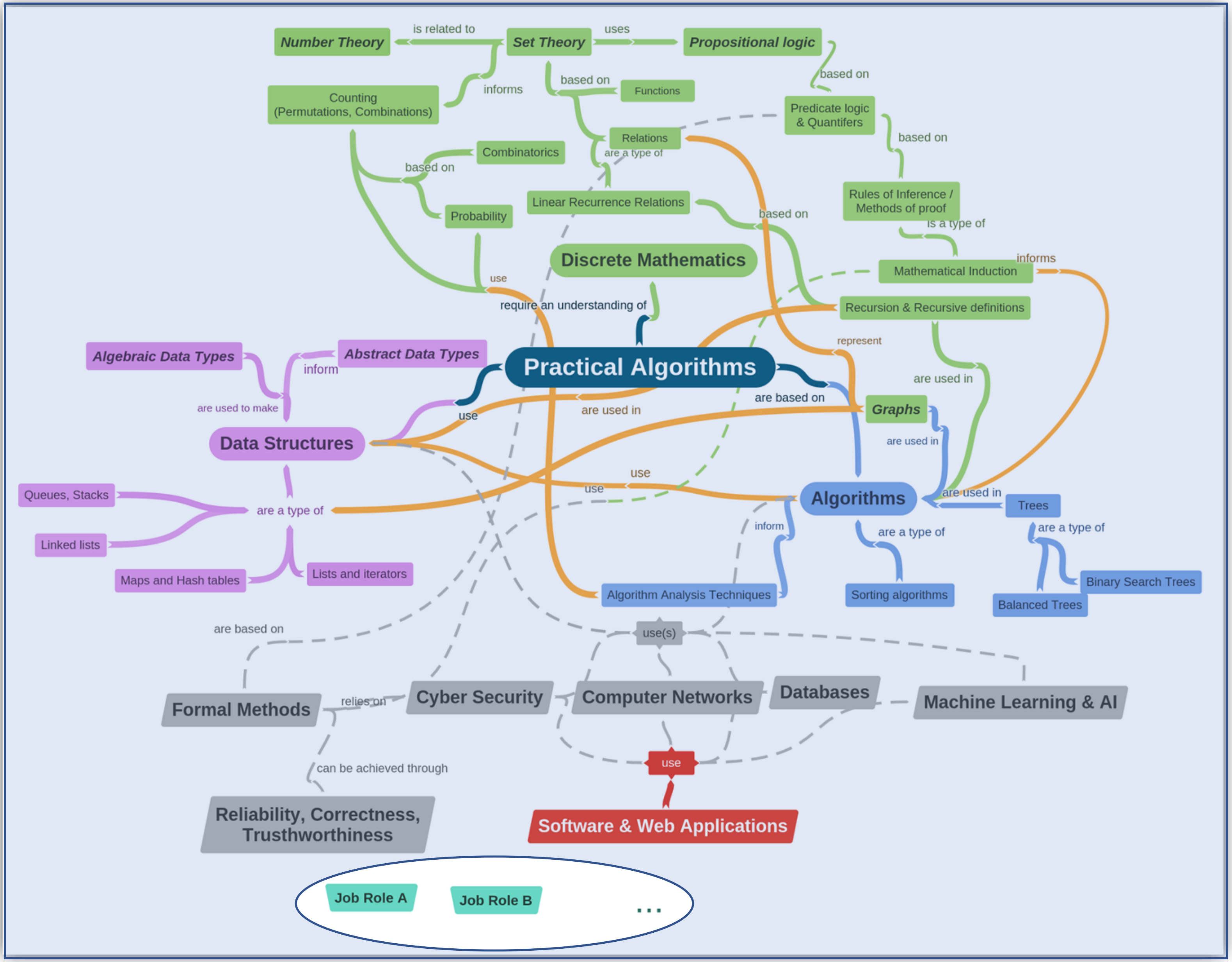
The “Practical Algorithms” Concept Map



The “Practical Algorithms” Concept Map



The “Practical Algorithms” Concept Map



Utility: Teacher

- A personal, internal model of concepts and their interconnectedness
- Develop an over-arching narrative for the course
- Connections between *universal truths* and *applied skills*
- Identify a suitable order of delivery of topics

Utility: Student

- Appreciate the temporal aspect of knowledge acquisition
- Connections between course concepts and workplace roles
- Track and evaluate their own understanding as the course progresses
- Deeper, more meaningful learning *(retain and utilize)*

The Solution?

- Not quite...
- We have framed the problem, and highlighted the challenges.
- We now suggest one approach – the use of concept maps - that we have used against this backdrop.
  - Can be useful in addressing different viewpoints and motivations
- There can (should) be others that can complement this tool.
- We have not yet carried out a quantitative or qualitative analysis of our approach.

A proposal to

address the challenge of motivating students

learning theoretical concepts

in a work-based learning setting.

Why is *why* important (*meta-whyness*)

Identifying the challenges

Towards addressing the challenges: using *concept maps*


The End