From Structural Induction to Graph Theory

Motivation, Algroithms, Applications, and Correctness

by Eric Yang Xingyu

This is a handout for the lecture on graph theory for my fellows. It covers the basic concepts of graph theory, including the definition of graphs, trees, and the basic algorithms for graph traversal. The handout also includes the correctness proof of the algorithms. Before delve into the graph theory, we will first introduce the concept of structural induction, which is the foundation of the correctness proof of the graph algorithms, and other structural algorithms.

Contents

§1. From Induction to General Induction

Remark 1.1 .
This lecture note is compiled using typst, and has two versions, one for the lecturer and one for the student. The latter version will be used during the lecture as handouts for the students. After the lecture, the lecturer version will be shared with students for further study or reference, which contains hints for lecture and solution/proofs for examples and exercises.

§1.1. Recap on Induction

Definition 1.1.1 (Induction).
Induction is a method of proof in which we prove that a statement is true for all natural numbers by proving that it is true for the smallest natural number and then proving that if it is true for some natural number, then it is true for the next natural number.

The most common use case of induction is to prove a statement defined on the natural numbers.

Example 1.1.1 .
Prove that $1 + 2 + 3... + n = \frac{n(n+1)}{2}$ for all $n \geq 1$.

Proof. Not discussed as it is trivial. □

§1.1.1. Motivation & Assumption

The principle of induction is based on the completeness of the natural numbers, which is the foundation of the induction. The completeness of the natural numbers is the property that every non-empty subset of the natural numbers has a least element. This property is the foundation of the induction, which allows us to prove a statement for all natural numbers by proving it for the smallest natural number and then proving that if it is true for some natural number, then it is true for the next natural number.

Definition 1.1.1.1 (Naive Definition of Natural Numbers).
The natural numbers are the set of positive integers, possibly including zero.

Is $0 \in \mathbb{N}$ is a question that has been debated for centuries until now. In different branches of mathematics, the definition of natural numbers varies. In this lecture, we will use $\mathbb{N}_0$ for the set of natural numbers including zero, and $\mathbb{N}$ for the set of natural numbers excluding zero.

Axiom 1.1.1.1 (Peano Axioms).
The Peano axioms are a set of axioms for the natural numbers presented by the 19th-

century Italian mathematician Giuseppe Peano. These axioms define the natural numbers as a set of objects, a number system, and a set of operations.

1. $0 \in \mathbb{N}$
2. $\forall n \in \mathbb{N}, n + 1 \in \mathbb{N}$
3. $\forall n \in \mathbb{N}, n + 1 \neq 0$
4. $\forall n, m \in \mathbb{N}, n + 1 = m + 1 \Rightarrow n = m$
5. (Induction) If a set S of natural numbers contains 0 and also the successor of every number in S, then $S = \mathbb{N}$.

Remark 1.1.1.1 .
It is quite normal in both computer science and mathematics that a extremely complex system can be built on a few simple rules, or we say axioms here. Some examples are systems of differential equations, recurrence relations, etc.

§1.1.2. Weak Induction

Definition 1.1.2.1 (Weak Induction) **.**
Weak induction is a method of proof in which we prove that a statement is true for all natural numbers by proving that it is true for the smallest natural number and then proving that if it is true for some natural number, then it is true for the next natural number. Formally, let $P(n)$ be a statement for each $n \in \mathbb{N}_0$. If $P(0)$ is true, and $\forall k \in \mathbb{N}_0, P(k) \Rightarrow P(k+1)$, then $\forall n \in \mathbb{N}_0, P(n)$ is true.

Example 1.1.2.1 .
Prove that $\forall n \in \mathbb{N}_0, 3 \mid n^3 + 2n$.

Remark 1.1.2.1 .
The statement $3 \mid n^3 + 2n$ means that $n^3 + 2n$ is divisible by 3, which means that

$$\exists m \in \mathbb{Z} : n^3 + 2n = 3m.$$

Accordingly, if $3 \nmid n^3 + 2n$, then $n^3 + 2n$ is not divisible by 3, which means that

$$\exists m \in \mathbb{Z} : n^3 + 2n = 3m + r, r \in \mathbb{Z}_{>0}^{<3}$$

- Chain of logic
  - Base case: $P(0)$ holds.
  - Inductive step: Assume $P(k)$ is true, prove $P(k+1)$ is true. Finally we have

$$\forall k \geq 0, P(k) \Rightarrow P(k+1)$$

  - Conclusion: by the principle of induction, we have $\forall k \geq 0, P(k)$ is true.

§1.1.3. Strong Induction

Definition 1.1.3.1 (Strong Induction).
Strong induction is a method of proof in which we prove that a statement is true for all natural numbers by proving that it is true for the smallest natural number and then proving that if it is true for all natural numbers less than or equal to some natural number, then it is true for the next natural number. Formally, let $P(n)$ be a statement for each $n \in \mathbb{N}_0$). If $P(0)$ is true, and $\forall k \in \mathbb{N}_0, (\forall m \leq k, P(m)) \Rightarrow P(k+1)$, then $\forall n \in \mathbb{N}, P(n)$ is true.

Example 1.1.3.1 .
Prove that every integer greater than 1 can be written as a product of prime numbers.

- Chain of logic:
  - Base case: $P(0)$ holds.
  - Inductive step: Assume $P(0), P(1), ..., P(k)$ are true, prove $P(k+1)$ is true. Finally we have

$$\forall k \geq 0, (\forall m \leq k, P(m)) \Rightarrow P(k+1) \text{ or}$$
$$\forall k \geq 0, (P(0) \land P(1) \land ... \land P(k)) \Rightarrow P(k+1) \text{ or}$$
$$\forall k \geq 0, \bigwedge_{m \leq k} P(m) \Rightarrow P(k+1)$$

§1.1.4. Generalised Induction

Definition 1.1.4.1 (Generalised Induction).
Generalised induction is a method of proof in which we prove that a statement is true for all objects in a set by proving that it is true for the smallest object and then proving that if it is true for all objects smaller than or equal to some object, then it is true for the next object. Formally, we implicitly define a bijection between the set of objects and the natural numbers, and then prove the statement for all natural numbers, as what we do in normal induction.

- pattern:
  - The conclusion holds when the structure is minimized.
  - Assume the conclusion holds for all smaller sub-structures, prove the conclusion holds for the current structure.
  - The conclusion holds for all structures.
- Why Generalize Induction?
  - Natural numbers are insufficient for complex structures (e.g., proving properties of graphs or programs).
  - Unified framework: Well-foundedness captures the essence of induction, allowing it to apply broadly.

§1.1.4.1. Structural Induction

Definition 1.1.4.1.1 (Structural Induction).
Structural induction is a generalisation of mathematical induction to data structures. It is a method of proof in which we prove that a property holds for all elements of a data

structure by proving that it holds for the smallest elements and then proving that if it holds for all sub-structures of some element, then it holds for the element itself.

In the context of computer science, we often use structural induction, which is a kind of generalised induction, to prove properties of data structures including strings, trees, and graphs.

Example 1.1.4.1.1 (Well-formed Parentheses).
A Well-formed string of parentheses is a string that consists of a series of opening and closing parentheses, such that each closing parenthesis matches the most recent unmatched opening parenthesis. For example, the strings "(()())" and "((()))" are well-formed, while the strings "())(" and "(()" are not well-formed. Prove that every well-formed string of parentheses has an equal number of opening and closing parentheses.

Remark 1.1.4.1.1.
Let $\Sigma = \{(,)\}$. Let Σ^* be the set of all finite strings over Σ. Let $\text{open} : \Sigma^* \to \mathbb{N}_0$ and $\text{close} : \Sigma^* \to \mathbb{N}_0$ be functions counting the number of opening and closing parentheses in a string S, respectively. ($\mathbb{N}_0$ denotes the set of non-negative integers). The set of **Well-Formed Parentheses** strings, denoted WFP, is a subset of Σ^* defined recursively as the smallest set satisfying:

[**Base Case:**] The empty string λ is in WFP.

[**Recursive Step 1:**] If S' is in WFP, then the string (S') is in WFP, where (S') is the concatenation of S' with an opening and closing parenthesis.

[**Recursive Step 2:**] If S' and T' are in WFP, then their concatenation $S'T'$ is in WFP.

Example 1.1.4.1.2.
Let Σ be a finite alphabet. A string over Σ is a finite sequence of characters from Σ recursively defined by the alphabet. We use Σ^* to denote the set of all strings over Σ. For some string $w \in \Sigma^*$, let w^R denote the reversed string of w. Prove that for any two strings $w_1, w_2 \in \Sigma^*$,

$$w_1^R w_2^R = (w_2 w_1)^R$$

Note: you may use λ to denote the empty string. Additionally, w^R is recursively defined as follows:
- If $w = \lambda$, then $w^R = \lambda$.
- If $w = xw'$, where $x \in \Sigma$ and $w' \in \Sigma^*$, then $w^R = w'^R x$.

- Other generalised induction(not covered in this lecture):
  - Transfinite Induction
  - Noetherian Induction

§2. Introduction to Graph Theory

Definition 2.1 (Graph).
A graph G is an ordered pair $G = (V, E)$, where V is a set of vertices and E is a set of edges. Each edge is a pair of vertices (u, v), where $u, v \in V$. We use $V(G)$ and $E(G)$ to denote the set of vertices and edges of G, respectively.

- In a nutshell, a graph is an abstraction of some tangible or intangible objects and their relationships.
- The vertices represent the objects, and the edges represent the relationships between the objects.

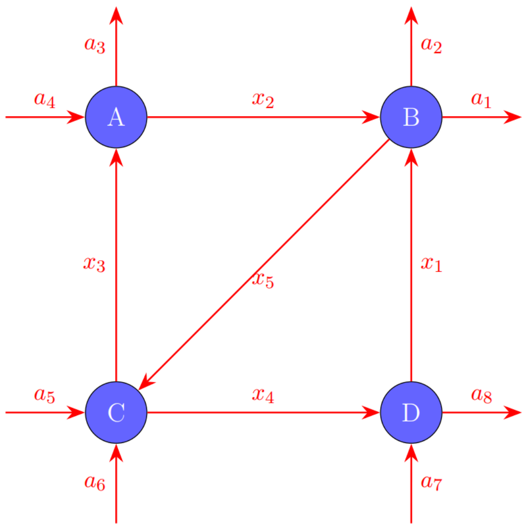


Figure 1: An example of a graph for road traffic

- The above is a Directed Graph, where the edges have directions.
- The graph is also weighted, meaning that the edges have weights, or attached with a value.
- Using the notation $G = (V, E)$, we can represent the graph as $G = (\{A, B, C, D\}, \{(A, B), (B, C), (C, A), (C, D), (D, B)\})$.

§2.1. Some Basic Notions

Definition 2.1.1 (Weight of Edges).
The weight of an edge in a graph is a value associated with the edge. The weight can represent the cost, distance, probability or any other value associated with the edge.

Definition 2.1.2 (Direction of Edges).
The edges of a graph can be directed or undirected. In a **directed graph**, the edges have directions, meaning that the edge (u, v) is different from the edge (v, u). In an **undirected graph**, the edges do not have directions, meaning that the edge (u, v) is the same as the edge (v, u).

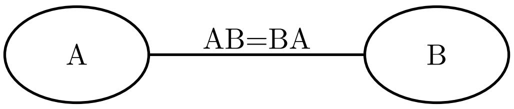


Figure 2: Undirected Graph

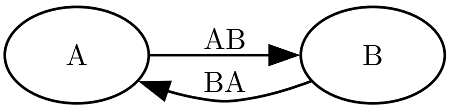


Figure 3: Directed Graph

Definition 2.1.3 (Loop).
A loop is an edge that connects a vertex to itself. In a graph, a loop is an edge of the form (v, v), where $v \in V$.

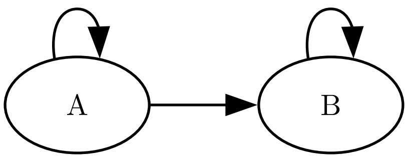


Figure 4: Graph with a loop

Definition 2.1.4 (Simple Graph).
A simple graph is a graph in which there is at most one edge between any two vertices and no edge from a vertex to itself.

Type	Edges	Multiple Edges?	Loops Allowed?
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

Table 1: Graph Typology

§2.2. Applications of Graph

Graphs are widely used in computer science, mathematics, and other fields to model relationships between objects. Some common applications of graphs include:

- Social networks: representing relationships between people.
- Road networks: representing roads and intersections.
- Network traffic: representing data flow between devices.
- Computer networks: representing connections between computers.
- Scheduling: representing tasks and dependencies between tasks.
- Circuit design: representing components and connections between components.

Example 2.2.1 .
An example of a graph for network traffic is the webpage ranking. Webpage ranking is an algorithm used by search engines to rank web pages in search results. The algorithm uses a graph to represent the web pages and the links between them. We cannot discuss the details of the algorithm here, but in this algorithm, we use a directed weighted graph to represent possibility of a user visiting a webpage from another webpage.

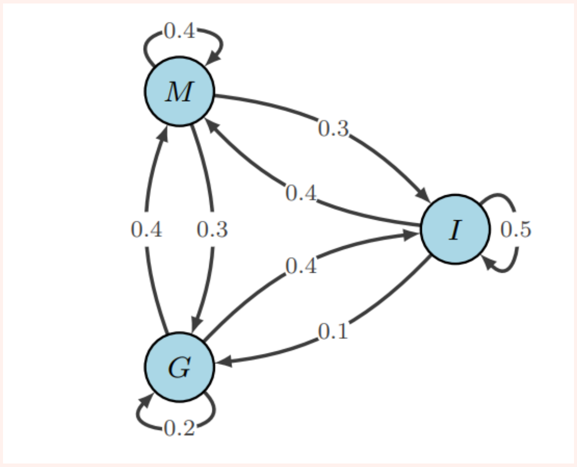


Figure 5: An example of a graph for network traffic

We will look into the details of this example in matrix representation of a graph.

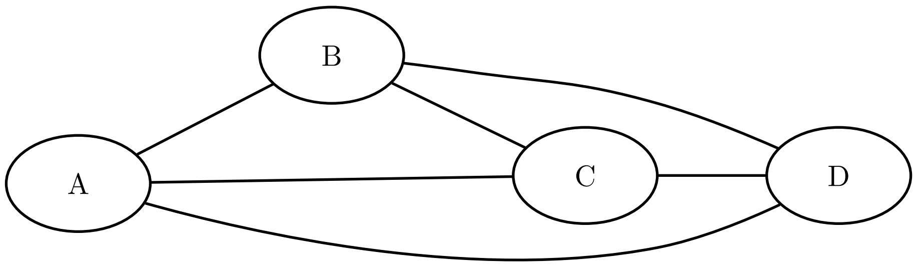
§2.3. More Terminologies

Definition 2.3.1 (Degree of a Vertex).
The degree of a vertex v in a graph is the number of edges incident to v. In an undirected graph, the degree of a vertex is the number of edges connected to the vertex. In a directed graph, the degree of a vertex is the sum of the in-degree and out-degree of the vertex, where the in-degree is the number of edges pointing to the vertex, and the out-degree is the number of edges pointing from the vertex.

§2.3.1. Path and Cycle

Definition 2.3.1.1 (Path).
A path in a graph is a sequence of vertices in which each vertex is connected to the next vertex by an edge. A path is simple if it does not contain any repeated vertices.

For example, in the graph below, the sequence $A \to B \to C \to D$ is a path.



We also introduce some special subsets of paths.

Definition 2.3.1.2 (Cycle).
A cycle in a graph is a path that starts and ends at the same vertex. A cycle is simple if it does not contain any repeated vertices except the starting and ending vertex.

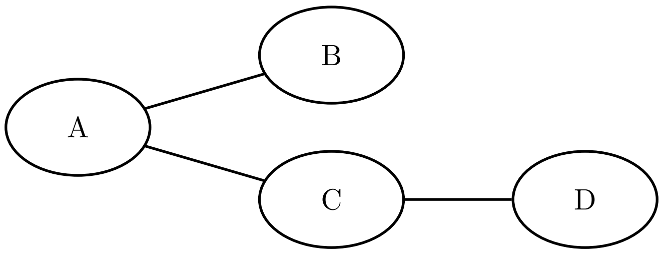
For example, in the previous graph, the sequence $A \to B \to C \to D \to A$ is a cycle.

§2.3.2. Connected & Ascyclic Graph

Definition 2.3.2.1 (Connected Graph).
A graph is connected if there is a path between every pair of vertices in the graph.

Definition 2.3.2.2 (Acyclic Graph).
A graph is acyclic if it does not contain any cycles.

Definition 2.3.2.3 (Complete Asyclic Graph).
A complete acyclic graph is a graph in which every pair of vertices is connected by a **unique** path.



Remark 2.3.2.1.
A more well-known name for a complete acyclic graph is a **tree**.

§2.3.3. Subgraph

Definition 2.3.3.1 (Subgraph).
A subgraph of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq$

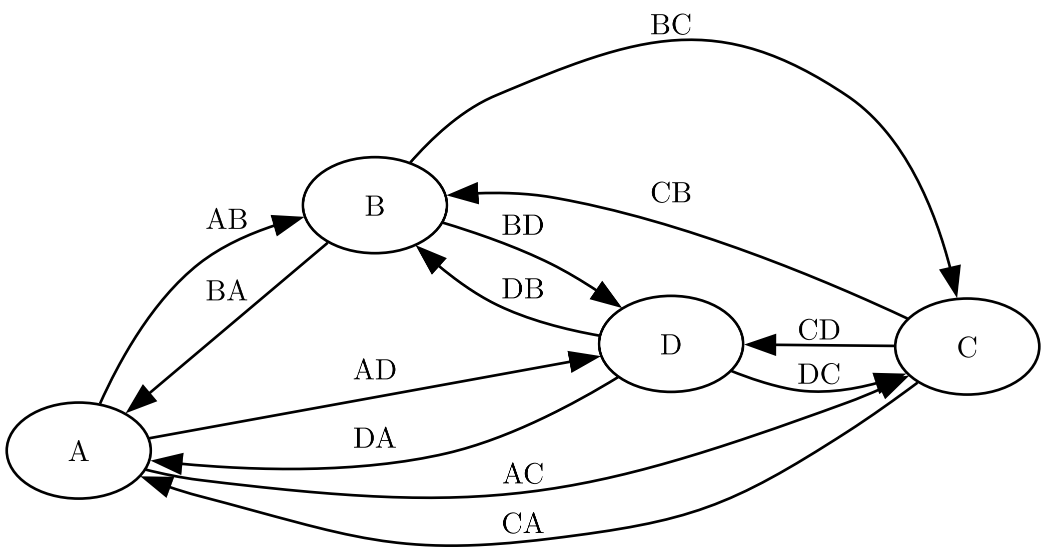
E. That is, a subgraph is a graph that contains a subset of the vertices and edges of the original graph.

In a straightforward way, a subgraph is a graph that can be obtained by removing some vertices and edges from the original graph.

§2.3.4. Complete Graph

Definition 2.3.4.1 (Complete Graph).
A complete graph is a graph in which every pair of vertices is connected by an edge. The number of edges in a complete graph with n vertices is

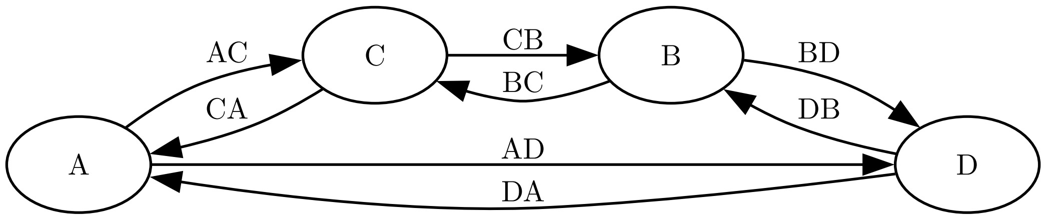
$$\binom{n}{2} = \frac{n(n-1)}{2}.$$



Remark 2.3.4.1 .
A complete graph with n vertices is denoted by K_n.

§2.3.5. Bipartite Graph

Definition 2.3.5.1 (Bipartite Graph).
A bipartite graph is a graph whose vertices can be divided into two disjoint sets V_1 and V_2 such that every edge connects a vertex in V_1 to a vertex in V_2.



§2.3.6. Finite and Infinite Graph

Definition 2.3.6.1 (Finite Graph).
A graph is finite if it has a finite number of vertices and edges.

Definition 2.3.6.2 (Infinite Graph).
A graph is infinite if it has an infinite number of vertices or edges.

Remark 2.3.6.1.
The concept of infinite graphs is essential in mathematics and computer science, especially in the study of infinite structures and algorithms on infinite structures.

Problem 2.3.6.1.
Anything discussed earlier in the lecture can be abstract by a infinite graph?

§2.4. Useful Results on Graph

§2.4.1. Handshaking Theorem

Theorem 2.4.1.1 (Handshaking Theorem).
The handshaking theorem states that for any simple graph, the sum of the degrees of all vertices is equal to twice the number of edges. A loop at a vertex v is typically counted as contributing 2 to $\deg(v)$.

Formally, let $G = (V, E)$ be an undirected graph with $m = |E|$ edges. Then

$$\sum_{v \in V} \deg(v) = 2m.$$

How can we prove it? Eventhough it is trivial, we can prove it by structural induction on the definition of a undirected graph.

Problem 2.4.1.1.
There are 605 people in a party. Each person shakes hands with some other people. Suppose that each of them shakes hands with at least one person. Prove that there must be someone who shakes hands with at least two persons.

Theorem 2.4.1.2 (Number of Vertices with Odd Degree).
In any simple graph G, the number of vertices with odd degree is always even.

Problem 2.4.1.2 .
Prove that among a group of people($n > 2$), there are at least 2 persons where the number of people they know is the same.

§2.5. Representation of Graph

Graph can be represented in different ways, including adjacency matrix, adjacency list, and incidence matrix. Each representation has its own advantages and disadvantages, and the choice of representation depends on the specific problem being solved. We introduce two of the most common representations: adjacency matrix and adjacency list.

§2.5.1. Adjacency Matrix

Definition 2.5.1.1 (Adjacency Matrix) .
An adjacency matrix is a square matrix used to represent a graph. The rows and columns of the matrix correspond to the vertices of the graph, and the entries of the matrix indicate whether there is an edge between the corresponding vertices. If there is an edge between vertices u and v, the entry a_{uv} is 1; otherwise, it is 0.

If the graph is weighted, the entries of the adjacency matrix can be the weights of the edges instead of 0 or 1.

Example 2.5.1.1 .
Consider the graph $G = (V, E)$ with

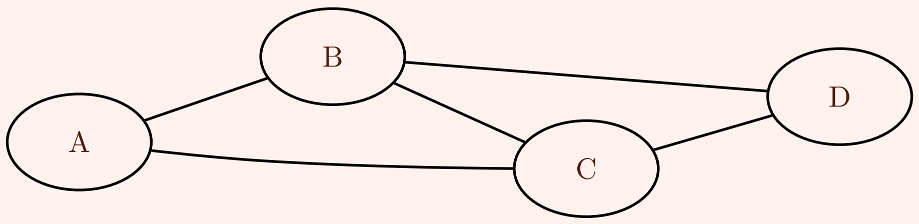
$$V = \{A, B, C, D\}$$

and

$$E = \{(A, B), (B, C), (C, A), (C, D), (D, B)\}.$$

The adjacency matrix of G is

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$



§2.5.1.1. Implementation

§2.5.2. Adjacency List

Definition 2.5.2.1 (Adjacency List).
Adjacency List

§2.5.2.1. Implementation

§2.5.3. Trade-offs between Adjacency Matrix and Adjacency List

§2.5.4. Graph Isomorphism

§2.6. Basic Graph Algorithms

§2.6.1. Breadth-First Search

§2.6.1.1. Definition

§2.6.1.2. Implementation

§2.6.1.3. Complexity Analysis

§2.6.1.4. Correctness

§2.6.2. Depth-First Search

§2.6.2.1. Definition

§2.6.2.2. Implementation

§2.6.2.3. Complexity Analysis

§2.6.2.4. Correctness