MTH210 — Discrete Mathematics II

James Li — 501022159 Professor: M. Delcourt
Email: mdelcourt@torontomu.ca

Table of Contents

1 Sequences and Series

A **sequence** is an ordered set of numbers.

$2, 4, 6, 8, 10 \ldots$ is an example of a sequence of positive numbers.
$a_1, a_2, a_3, \ldots, a_n$ denotes an infinite sequence.

- A sequence is defined **analytically** if each term a_i is defined by some function $f(i) = a_i$
- A sequence is defined **recursively** if the first k terms are given **explicitly** and the rest are given through a recursive function $a_n = f(a_{n-1}, a_{n-2}, \ldots, a_{n-k})$ for $n > k$.
- Even if two sequences are equal for small indexes, does not indicate that they don't diverge at some further point.

A **series** is the sum of all the terms in a sequence.

If m and n are integers and $m \leq n$, the symbol $\sum_{k=m}^{n} a_k$ is the summation from k, defined as:

$$\sum_{k=m}^{n} a_k = a_m + a_{m+1} + a_{m+2} + \cdots + a_n$$

- We call k the **index** of the summation.
- m is the **lower limit** of the summation.
- n is the **upper limit** of the summation.

1.1 Sums and Products

If m and n are integers and $m \leq n$, the symbol $\prod_{k=m}^{n} a_k$ is read as product from k equals m to n of a sub k, it can be written as:

$$\prod_{k=m}^{n} a_k = a_m \cdot a_{m+1} \cdot a_{m+2} \times \cdots \times a_n$$

Theorem 1.1. *The following properties hold for any integer $n \geq m$, given $a_m, \ldots$ and $b_m, \ldots$ sequences of real numbers.*

- $\sum_{k=m}^{n} a_k + \sum_{k=m}^{n} b_k = \sum_{k=m}^{n} (a_k + b_k)$
- $c \cdot \sum_{k=m}^{n} a_k = \sum_{k=m}^{n} (c \cdot a_k)$, *given some constant c*
- $(\prod_{k=m}^{n} a_k) \cdot (\prod_{k=m}^{n} b_k) = \prod_{k=m}^{n} (a_k \cdot b_k)$

Theorem 1.2. *The binomial theorem, also called n choose r is computed by using the following formula for $0 \leq r \leq n$:*

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}$$

2 Mathmatical Induction

Mathematical Induction is a two-step process, Take a statement in the form "For every integer $n \geq a$, a property P(n) holds true". We then apply the following:

1. The first step is called the **Basis Step**, this is where you show that the condition P of your starting point a, is true $\rightarrow$ Show that $P(a)$ is true.
2. The second step is called the **Inductive Step**, you show that for every integer $k \geq a$, if $P(k)$ is true, then $P(k+1)$ must also be true.
   (a) To perform this step, we must suppose that $P(k)$ holds, where $k \geq a$ (**Inductive Hypothesis**), therefore $P(k+1)$ must be true.

To write an inductive proof formally, we must clearly state all steps and assumptions, take the following example:

Prove. For every integer $n \geq 1, 1+2+\cdots+n = \dfrac{n(n+1)}{2}$ by **Induction**:

Proof. Basis Step (*Base Case*):

$$\text{Let } n = 1, P(1) \text{ holds because } \frac{1(1+1)}{2} = \frac{1(2)}{2} = 1$$

Inductive Step:

Let $n = k$ with $k \geq 1$. Suppose that $P(k)$ is true (*inductive hypothesis*). Thus we can show:

$$\begin{aligned} 1+2+\cdots+k+(k+1) &= (1+2+\cdots+k)+(k+1) \\ &= \frac{k(k+1)}{2} + (k+1) \\ &= \frac{k(k+1)}{2} + \frac{2(k+1)}{2} \\ &= \frac{k^2+k}{2} + \frac{2k+2}{2} \\ &= \frac{k^2+3k+2}{2} \\ &= \frac{(k+1)(k+2)}{2} \end{aligned}$$

Thus $1+2+\cdots+(k+1) = \dfrac{(k+1)((k+1)+1)}{2}$ (notice how this is in the form $1+2+\cdots+n = \dfrac{n(n+1)}{2}$). Therefore we can conclude that $P(k+1)$ is true. □

2.1 Recursion

Given a recursively defined sequence a_i, find an analytical formula using the following steps:

1. Find an explicit formula for a_i by making educated guesses.
2. Prove that the formula holds through induction.

Take the following example:

Prove. Given $b_0 = 1, b_n = \dfrac{b_n - 1}{1 - b_n - 1}$ for $n > 1$

Proof. Make an assumption for the formula of b_n, take the following guess:

$$b_n = \frac{1}{n+1} \text{ for } n \geq 0 \ (\textit{Conjecture})$$

Basis Case:

$$\text{Let } n = 0. P(0) \text{ is known to be true as } b_0 = 1 = \frac{1}{0+1}$$

Inductive Step:

Let $n = k$ with $k \geq 0$. Assume that $P(k)$ is true (*inductive hypothesis*). Thus we can show:

$$\begin{aligned} b_{k+1} &= \frac{b_k}{1 + b_k} \\ &= \frac{\frac{1}{k+1}}{1 + \frac{1}{k+1}} \\ &= \frac{\frac{1}{k+1}}{\frac{k+2}{k+1}} \\ &= \frac{1}{k+2} \\ &= \frac{1}{(k+1)+1} \end{aligned}$$

Hence $P(k+1)$ must be true. □

2.2 Strong Induction

Strong induction follows the same steps as normal/weak induction, with the difference that the basis step may contain proofs for several values and $P(n)$ is assumed not just for a single n but for all values n through k, only then is the truth of $P(k+1)$ proved. The steps for strong induction are as follows:

Let $P(n)$ be a property that is defined for integers n, let a, b be fixed integers such that $a \leq b$. Suppose the following statements:

1. $P(a), P(a+1), \ldots, P(b)$ are all true. (**Basis Step**)
2. For every integer $k \geq b$, if $P(i)$ is true for each integer i from a through k, then $P(k+1)$ is true. (**Inductive Step**)

then the statement "For every integer $n \geq a, P(n)$" is true. The supposition that $P(i)$ is true is the inductive hypothesis in this case.

Prove. Given a sequence a_i,

$$a_0 = 12, a_1 = 29 \quad a_n = 5a_{n-1} - 6a_{n-2}$$

Show that for all $n \geq 0$,

$$a_n = 5 \cdot 3^n + 7 \cdot 2^n$$

Proof. Proceed by induction on n.

Basis Step: We take that $P(0)$ and $P(1)$ are true.

Inductive Step: Let $n = k + 1$. Assume $P(i)$ is true for $k \geq i \geq 0$ (*inductive Hypothesis*). Thus:

$$\begin{aligned} a_k &= 5 \cdot 3^k + 7 \cdot 2^k \\ a_{k-1} &= 5 \cdot 3^{k-1} + 7 \cdot 2^{k-1} \\ a_{k+1} &= 5(5 \cdot 3^k + 7 \cdot 2^k) - 6(5 \cdot 3^{k-1} + 7 \cdot 2^{k-1}) \\ &= 25 \cdot 3^k + 35 \cdot 2^k - 30 \cdot 3^{k-1} - 42 \cdot 2^{k-1} \\ &= 25 \cdot 3^k + 35 \cdot 2^k - 10 \cdot 3^k - 21 \cdot 2^k \\ &= 15 \cdot 3^k + 14 \cdot 2^k \\ &= 5 \cdot 3^{k+1} + 7 \cdot 2^{k+1} \end{aligned}$$

Hence $P(k+1)$ is true. □

3 Trails, Paths and Circuits

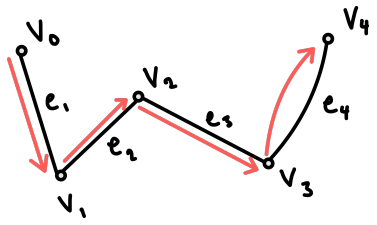
Recall the following:

- Given some graph G, the **vertex set** of G is denoted as $V(G)$ and the **edge set** of G is denoted as $E(G)$.
- A **simple** graph has no parallel edges or loops and is assumed to always be finite.
- The **degree** of a vertex is the number of edges **incident** on itself, edges that loop are counted twice.
- Complete graphs are fully connected simple graphs.

Given some graph G with vertices $v, w \in G$, a **walk** from v to w is defined as a fininte alternating sequence of adjacent vertices and edges, in the form:

$$v_0e_1, v_1e_2, v_2e_3, \ldots, v_{n-1}e_n \to w$$

Example of a walk between $v = v_0$ and $w = v_4$ on some graph G:



This would be denoted as: $ve_1, v_1e_2, v_2e_3, v_3e_4$

Two vertices v and w are said to be **connected** if there exists a walk between them. The graph G is said to be connected if there exists a walk between every pair of vertices within it, otherwise it is called **disconnected**.

- **Trails** are walks without repeated **edges**.
- **Paths** are trails without repeated **vertices**.
- **Closed Walks** start and end on the **same vertex**.
- **Circuits** are closed walks that contain no **repeated edges** and contain at least one edge.
- **Simple Circuits** are circuits that contain no **repeated vertices** except the first and last.

3.1 Euler Circuits

Given some graph G, an **Euler Circuit** for G is defined as a circuit in G which contains every vertex and edge in G. Meaning it starts and ends at the same vertex and contains all vertices with no repeated edges.

- If a graph has an Euler Circuit, then the graph must have a positive **even** degree.
- If a graph has an **odd** degree, then it does not have a Euler Circuit.
- A graph G, has a Euler Circuit if and only if, G is connected and every vertex of G has a positive even degree.

3.2 Euler Trails

Given some graph G, an **Euler Trail** for G from vertices $v, w \in G$, is defined as a sequence of adjacent edges and vertices from v to w that passes through every edge **exactly one** and every vertex **at least once**.

- A graph G only has an Euler Trail from v to w, if v, w have odd degree whilst every other vertex has even degree.

3.3 Subgraphs

Given graphs G, H. H is said to be a **subgraph** of G if:

$$V(H) \subseteq V(G), E(H) \subseteq E(G) \text{ and every edge in } H \text{ has the same endpoints in } G.$$

All graphs are considered subgraphs of themselves.

3.4 Hamiltonian Circuits

Given a graph G, a **Hamiltonian Circuit** for G is defined as a simple circuit which includes every vertex of G, meaning it is a sequence of adjacent vertices and distinct edges where every vertex appears once, except for the first and last vertex, which are the same.

4 Isomorphisms of Graphs

Recall the following:

- Injective/One-to-one Functions: $f : X \to Y$ maps all elements in the domain of X on Y **distinctly**.
- Surjective/Onto Functions: $f : X \to Y, \forall y \in Y, \exists x \in X$ such that $f(x) = y$.
- Bijective Functions: $f : X \to Y$ such that f is both one-to-one **and** onto.

Given graphs G, G' with vertex sets $V(G), V(G')$ and edge sets $E(G), E(G')$, we say G is **Isomorphic** to G', if and only if there exists **one-to-one** correspondences, $g : V(G) \to V(G')$ and $h : E(G) \to E(G')$, that preserves adjacencies for each $v \in V(G), e \in E(G)$.

$$v \text{ is an endpoint of } e \Leftrightarrow g(v) \text{ is and endpoint of } h(e)$$

To determine if two graphs are isomorphic, start by mapping a **bijection** between vertices and edges relative between the two graphs and see if the function holds.

4.1 Isomorphism of Simple Graphs

Given simple graphs G, G', they are isomorphic if $[u, v]$ is an edge in $G \Leftrightarrow [g(u), g(v)]$ is an edge in G' and preserves adjacencies. The difference between this definition and the standard graph isomorphism definition is that for simple graphs we do not have to consider correspondence between the edge sets of the two graphs.

4.2 Invariants for Graph Isomorphism

Given graphs G, G', a property P is called an **invariant** for graph isomorphism, if and only if, G has the property P and G is isomorphic to $G' \to G'$ has the property P. The following properties are invariants for graph isomorphism, where n, m, k are non-negative positive integers.

1. has n vertices.
2. has m edges.
3. has a vertex of degree k.
4. has m vertices of degree k.
5. has a circuit length k.
6. has a simple circuit of length k.
7. has m simple circuits of length k.
8. is connected.
9. has an Euler Circuit.
10. has a Hamiltonian Circuit.

5 Trees

5.1 Characterizing Trees

If n is a positrive integer then any tree with n vertices has $n-1$ edges, the converse to this fact is that any connected graph with n vertices and $n-1$ edges is a tree.

- If even one new edge (not vertex) is added to a tree then the graph must contain a circuit.
- Removing an edge from a circuit does not disconnect a graph.
- It can be shown that every connected graph has a subgraph that is a tree.
- If n is a positive integer any graph with n vertices and fewer than $n-1$ edges is not connected.

Let T be a tree, if T has at least 2 vertices , then a vertex of degree 1 in T is called a **leaf** or a **terminal vertex**, and a vertex of degree greater than 1 in T is called an **internal vertex** or **branch vertex**. The unique vertex in a trivial tree is also called a **leaf**.

5.2 Spanning Trees

A spanning tree for a graph G is a subgraph of G that contains every vertex of G and is a tree. Every connected graph G has at least 1 spanning tree and any two spanning trees for a graph have the same number of edges.

5.3 Weighted Graph

A **weighted graph** is a graph for which each edge has an associated positive real number called **weight**. The sum of all the edges is called the **total weight** of the graph. A **minimum spanning tree** (MST) for a connected weighted graph, is a spanning tree that has the least possible total weight compared to all other spanning trees for the graph. If G is a weighted graph and e is an edge of G then $w(e)$ denotes the weight of e and $w(G)$ denotes the total weight of G.

5.4 Algorithms for finding Minimum Spanning Trees

Types of Algorithms:

- Brute Force: Simply finding every spanning tree and comparing the total weights of each.
- Greedy Algorithms: Making a sequence of locally optimal moves can lead to a globally optimal solution.

5.4.1 Kruskal's Algorithm

In this algorithm, the edges of a connected, weighted graph are examined one by one in order of increasing weight, then:

1. Initialize T to have all the vertices of G and no edges.
2. Let E be the set of all the edges of G, and let $m := 0$
3. **while** $(m < n-1)$

   Find an edge e in E of least weight.

   Delete e from E.

   if addition of e to the edge set of T does not produce a circuit

   then add e to the edge set of T and set $m := m+1$
4. The resulting T will be the MST of the input graph.

5.4.2 Prim's Algorithm

6 Placeholder

7 Placeholder

8 Placeholder