Not the definition of a graph; page 6

A **graph** is a diagram consisting of points called **vertices**, joined by lines, called **edges**. Each edge **joins** exactly two vertices.

The definition of a graph; page 26

A **graph** consists of a set of elements called **vertices** and a set of elements called **edges**. Each edge **joins** exactly two vertices.

Not the definition of a digraph; page 16

A **digraph** is a diagram consisting of points called **vertices**, joined by directed lines, called **arcs**. Each arc **joins** exactly two vertices.

The definition of a digraph; page 85

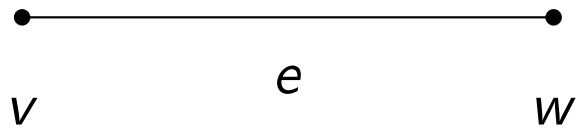
A **digraph** consists of a set of elements called **vertices** and a set of elements called **arcs**. Each arc **joins** two vertices in a specified direction.

Definition; page 26

In a graph, two or more edges joining the same pair of veritces are **multiple edges**. An edge joining a vertex to itself is a **loop**.

A graph with no multiple edges or loops is a **simple graph**.

Definition; page 27



The vertices v and w of a graph are **adjacent** vertices if they are joined by an edge e. The vertices v and w are **incident** with the edge e, and the edge e is **incident** with the vertices v and w.

Definition; page 29

Two graphs G and H are **isomorphic** if H can be obtained by relabelling the vertices of G - that is, if there is a one-one correspondence between the vertices of G and those of H, such that the number of edges joining each pair of vertices in G is equal to the number of edges joining the corresponding pair of vertices in H. Such a one-one correspondence is an **isomorphism**.

Definition; page 33

A **subgraph** of a graph G is a graph all of whose vertices are vertices of G and all of whose edges are edges of G.

Definition; page 35

In a graph, the **degree** of a vertex v is the number of edges incident with v, with each loop counted twice, and is denoted by **deg v**.

Definition; page 36

The **degree sequence** of a graph G is the sequence obtained by listing the vertex degrees of G in increasing order, with repeats as necessary.

Theorem 2.1: Handshaking Lemma; page 37

In any graph, the sum of all of the vertex degrees is equal to twice the number of edges.

Definition; page 39

A **walk of length k** in a graph is a succession of k edges of the form uv, vw, wx, $\ldots$, yz.

This walk is denoted by $uvwx \ldots yz$ and is referred to as **a walk between u and z**.

Definition; page 40

A **trail** is a walk in which all of the edges, but not necessarily all of the vertices, are different.

A **path** is a walk in which all of the edges and all of the vertices are different.

Definition; page 41

A graph is **connected** if there is a path between each pair of vertices, and is **disconnected** otherwise.

An edge in a connected graph is a **bridge** if its removal leaves a disconnected graph.

Every disconnected graph can be split up into a nuber of connected subgraphs, called **components**.

Definition; page 42

A **closed walk** in a graph is a succession of k edges of the form uv, vw, wx, $\ldots$, yz, zu; that starts and ends at the same vertex.

A **closed trail** is a closed walk in which all of the edges are different.

A **cycle** is a closed walk in which all of the edges and all of the intermediate vertices are different.

A walk or a trail is **open** if it starts and finishes at different vertices.

Definition; page 43

A graph is **regular** if its vertices all have the same degree.

A regular graph is r-**regular**, or **regular of degree** r, if the degree of each vertex is r.

Theorem 2.2; page 44

Theorem

Let G be an r-regular graph with n vertices. Then G has $\frac{nr}{2}$ edges.

Definitions and Notation; page 45

A **complete graph** is a graph in which each vertex is joined to each of the others by exactly one edge.
The complete graph with n vertices is denoted K_n.

A **null graph** is a graph with no edges on n vertices.
The null graph with n vertices is denoted N_n.

A **cycle graph** is a graph consisting of a single cycle of vertices and edges.
The cycle graph with n vertices is denoted C_n.

Definition; page 47, 48

A **bipartite graph** is a graph whose set of vertices can be split into two subsets A and B in such a way that each edge of the graph joins a vertex in A and a vertex in B.

A **complete bipartite graph** is a bipartite graph in which each vertex in A is joined to each vertex in B by just one edge.
The complete bipartite graph with r vertices in A and s vertices in B is denoted $K_{r,s}$.

Definition; page 49

A **tree** is a connected graph with no cycles.

In a tree, there is just one path between each pair of vertices.

Definition; page 50

A **path graph** is a tree consisting of a single path through all of its vertices.
The path graph with n vertices is denoted P_n.

A k-**cube** or k-**dimensional cube** is the graph obtained from labelling the vertices with binary words of length k, and joining two vertices with an edge if the words differ in one place.
The k-cube is denoted Q_k.

Definition and Notation; page 58

The **complement** $\overline{G}$ of a simple graph G is obtained by taking the vertices of G and joining two of them whenever they are not joined in G.

Homework, Chapter 2

Problems: 2.1 – 2.25

Exercises: 2.2,
2.4, 2.5 and 2.6,
2.8, 2.9,
2.10, 2.11, 2.12 (Important definition), (2.13), 2.14, (2.15),
2.16, 2.17, a solution to one of Game 1, 1A, 2, 3,
2.18, 2.19, 2.20 (Note the typo, it should read - ... even number of *negative* edges.)

★ – (...) – recommended.

Definitons; page 63

A connected graph is Eulerian if it contains a closed trail that includes every edge; such a trail is an **Eulerian trail**.

A connected graph is Hamiltonian if it contains a cycle that includes every vertex; such a trail is an **Hamiltonian cycle**.

Theorem 3.1, 3.2, 3.3; page 64, 65

Theorem

Let G be a graph in which each vertex has even degree. Then G can be split into cycles, no two of which have an edge in common.

Theorem

A connected graph is Eulerian if and only if each vertex has even degree.

Theorem

An Eulerian graph can be split into cycles, no two of which have an edge in common.

Definition, Theorem 3.4; page 67

A connected graph is **semi-Eulerian** if there is an open trail that includes every edge; such a trail is a **semi-Eulerian trail**.

Theorem

A connected graph is semi-Eulerian if and only if it has exactly two vertices of odd degree.

Theorem 3.5; page 73

Theorem (Ore's Theorem)

Let G be a simple connected graph with n vertices, where $n \geq 3$ and

$$deg v + deg w \geq n$$

for each pair of non-adjacent vertices v and w. Then G is Hamiltonian.

Definiton; page 74

A connected graph is **semi-Hamiltonian** if there is a path, but not a cycle, that includes every vertex; such a path is a **semi-Hamiltonian path**

Homework Chapter 3

Problems: 3.1 – 3.15

Exercises: 3.1, 3.2, 3.2,
3.4, 3.5, 3.6, 3.7
(3.8, 3.9,), 3.10, 3.11

★ – (...) – recommended.

The definition of a digraph; page 85

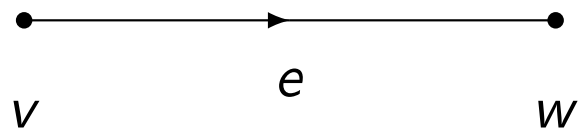
A **digraph** consists of a set of elements called **vertices** and a set of elements called **arcs**. Each arc **joins** two vertices in a specified direction.

Definition; page 85

In a digraph, two or more arcs joining the same pair of vertices in the same direction are **multiple arcs**. An arc joining a vertex to itself is a **loop**.

A digraph with no multiple arcs or loops is a **simple digraph**.

Definition; page 86



The vertices v and w of a digraph are **adjacent** vertices if they are joined (in either direction) by an arc e. An arc e that joins v to w is **incident from** v and **incident to** w; v is **incident to** e and w is **incident from** e

Definition; page 87

Two digraphs C and D are **isomorphic** if D can be obtained by relabelling the vertices of C - that is, if there is a one-one correspondence between the vertices of C and those of D, such that the arcs joining each pair of vertices in C agree in both number and direction with the arcs joining to corresponding vertices in D.

Definition; page 90

A **subdigraph** of a digraph D is a digraph all of whose vertices are vertices of D and all of whose arcs are arcs of D.

Definiton; page 92

The **underlying graph** of a digraph D is the graph obtained by replacing each arc of D by the corresponding undirected edge.

Definition; page 92

In a digraph, the **out-degree** of a vertex v is the number of arcs incident from v, and is denoted by **outdeg v**; the **in-degree** of a vertex v is the number of arcs incident to v, and is denoted by **indeg v**;

Definition; page 93

The **out-degree sequence** of a digraph D is the sequence obtained by listing the out-degrees of D in increasing order, with repeats as necessary. The **in-degree sequence** is defined analgously.

Theorem 4.1; page 94

Theorem (Handshaking Dilemma)

In any digraph, the sum of all the out-degrees and the sum of all the in-degrees are both equal to the number of arcs.

Definition; page 95

A **walk of length k** in a digraph is a succession of k arcs of the form uv, vw, wx, $\ldots$, yz.
This walk is denoted by $uvwx \ldots yz$ and is referred to as **a walk between** u **and** z.

A **trail** is a walk in which all of the arcs, but not necessarily all of the vertices, are different.

A **path** is a walk in which all of the arcs and all of the vertices are different.

Definition; page 95

A **closed walk** in a digraph is a succession of k arcs of the form uv, vw, wx, $\ldots$, yz, zu; that starts and ends at the same vertex.

A **closed trail** is a closed walk in which all of the arcs are different.

A **cycle** is a closed trail in which all of the intermediate vertices are different.

Definition

A digraph is **connected** if its underlying graph is a connected graph, and it is **disconnected** otherwise.

A digraph is **strongly connected** if there is a path between each pair of vertices.

Definitons; page 97

A connected digraph is **Eulerian** if it contains a closed trail that includes every arc; such a trail is an **Eulerian trail**.

A connected digraph is **Hamiltonian** if it contains a cycle that includes every vertex; such a cycle is an **Hamiltonian cycle**.

Theorem 4.2, 4.3; page 99

Theorem

A connected digraph is Eulerian if and only if, for each vertex, the out-degree equals the in-degree.

Theorem

An Eulerian digraph can be split into cycles, no two of which have an arc in common.

Homework Chapter 4

Problems: 4.1 – 4.15, 4.17 – 4.19

Exercises: 4.1, 4.2, 4.2,
4.4, 4.5, 4.6,
4.7, 4.8, 4.9, (4.10)
4.11, 4.12
4.14, 4.15, 4.16. 4.17

★ – (...) – recommended.

Definition; page 113

Let G be a graph with n vertices labelled $1, 2, 3, \ldots, n$.
The **adjacency matrix A(G)** of G is the $n \times n$ matrix in which the entry in row i and column j is the number of edges joining the vertices i and j.

Definition; page 115

Let D be a digraph with n vertices labelled $1, 2, 3, \ldots, n$.
The **adjacency matrix A(D)** of D is the $n \times n$ matrix in which the entry in row i and column j is the number of arcs from the vertex i to the vertex j.

Theorem 5.1; page 118

Let D be a digraph with n vertices labelled $1, 2, 3, \ldots, n$, let $\mathbf{A}$ be its adjacency matrix with respect to this listing of the vertices, and let k be any positive integer.
Then the number of walks of length k from vertex i to vertex j is equal to the entry in row i and column j of the matrix $\mathbf{A}^k$ (the kth power of the matrix $\mathbf{A}$).

Theorem 5.2; page 119

Let D be a digraph with n vertices labelled 1, 2, 3, ..., n, let $\mathbf{A}$ be its adjacency matrix with respect to this listing of the vertices, and let $\mathbf{B}$ be the matrix

$$\mathbf{B} = \mathbf{A} + \mathbf{A}^2 + \ldots + \mathbf{A}^{n-1}$$

Then D is strongly connected if and only if each non-diagonal entry in $\mathbf{B}$ is positive - that is $b_{ij} > 0$ whenever $i \neq j$.

Definition; page 123

Let G be a graph without loops, with n vertices labelled ①, ②, . . . , ⓝ and m edges labelled $1, 2, \ldots, m$
The **incidence matrix I(G)** of G is the $n \times m$ matrix in which the entry in row i and column j is

- 1 if the vertex i is incident with the edge j
- 0 otherwise.

Definition; page 125

Let D be a digraph without loops, with n vertices labelled ①, ②, . . . , ⓝ and m edges labelled $1, 2, \ldots, m$
The **incidence matrix I(D)** of D is the $n \times m$ matrix in which the entry in row i and column j is

- 1 if arc j is incident from vertex i
- -1 if arc j is incident to vertex i
- 0 otherwise.

Homework Chapter 5

Problems:
5.1 – 5.12, 5.14

Exercises:
5.1, 5.2, 5.3, 5.4
5.5, 5.6, 5.7, 5.8
5.9, 5.10, 5.11
5.14, 5.15

Theorem 6.1; page 143

Theorem (Equivalent Definitions of a Tree)

Let T be a graph with n vertices. Then the following statements are equivalent.

- *T is connected and has no cycles.*
- *T has $n-1$ edges and has no cycles.*
- *T is connected and has $n-1$ edges.*
- *T is connected and teh removal of any edge disconnects T.*
- *Any two vertices of T are connected by exactly one path.*
- *T contains no cycles, but the addition of any new edge creates a cycle.*

Definition; page 144

Let G be a connected graph. Then a **spanning tree** in G is a subgraph that includes every vertex and is also a tree.

Rigidity Criterion; page 154

If a braced rectangular framework, with rows r_1, r_2, r_3, ..., and columns c_1, c_2, c_3, ..., is rigid, then the braces must be located such that, under any attempted deformation of the framework,

- r_i remains parallel to r_j, for all r_i and r_j.
- c_i remains parallel to c_j, for all c_i and c_j.
- r_i remains perpendicular to c_j, for all r_i and c_j.

Bracings; page 157, 158

A braced rectangular framework is rigid if and only if its associated bipartite graph is connected.

A given bracing of a rectangular framework is not a minimum bracing if the associated bipartite graph has either of the following properties:

- the graph has n vertices and more then $n - 1$ edges;
- the graph contains a cycle.

Homework Chapter 6

Problems:
6.1 – 6.7, (6.8 – 6.9), 6.10 – 6.11

Exercises:
6.1,
6.3, 6.4,
(6.7)
6.8, 6.9, 6.10, 6.11

Construction of a Prüfer sequence; page 165

- Find the vertices of degree 1 and choose the one with the smallest label.
- Look at the vertex adjacent to the one just chosen and place its label in the first available position in the Prüfer sequence.
- Remove the vertix chosen in Step 1 and its incident edge, leaving a smaller tree.

Repeat for the remaining tree, continuing until there are only two vertices left. Then STOP; the required Prüfer sequence has been constructed.

Construct a labelled tree from a Prüfer sequence; page 167

- Draw the n vertices, labelling them from 1 to n, and list the integers from 1 to n
- Find the smallest number that is in the list but not in the Prüfer sequence, and also find the first number in the sequence; then add an edge joining the vertices with these labels.
- Remove the first number found in the last step from the list and the second number found from the sequence, leaving a smaller list and smaller sequence.

Repeat the last two steps for the remaining list and sequence, continuing until there are only two terms left in the list. Join the vertices with these labels and STOP; the required labelled tree has been constructed.

Theorem 7.1; page 170

Theorem (Cayley's Theorem)

The number of labelled trees with n vertices is n^{n-2}.

Definition; page 179

What is the 'middle' of a tree?

Repeatedly remove the vertices of degree 1.

If what remains is a single vertex, then that vertex is the **centre** of the tree.
If two adjacent vertices remain, then those two vertices are the **bicentre** of the tree.

A tree with a center is a **central tree**, a tree with a bicenter is a **bicentral tree**.

Definition; not in book

An alternate definition of 'middle' of a tree.

For every vertex v of degree 2 or more, count the number of vertice in each subtree along each of the edges joining the vertex v. Let n_v be the maximum of those numbers.

For a tree with n vertices:
If one vertex v has $n_v \leq \frac{1}{2}(n-1)$ then v is the **centroid**.
If two adjacent vertices $n_v = n_w = \frac{1}{2}n$ then vw is the **bicentroid**.

Homework Chapter 7

Problems:
7.1 – 7.6, (7.7 – 7.8), 7.9 – 7.12

Exercises:
7.1 – 7.5
(7.6)
7.7–7.9

Definition; page 183

Let T be a spanning tree of minimum total weight in a connected graph G. Then T is a **minimum spanning tree** or a **minimum connector** in G.

The MINIMUM CONNECTOR PROBLEM is: Given a weighted graph, find a minimum spanning tree.

Kruskal's Algorithm; page 184

START with a finite set of vertices, where each pair of vertices is joined by a weighted edge.

- List all the weights in ascending order.
- Draw the vertices and weighed edges corresponding to the first weight in the list, provided that, in so doing, no cycle is formed. Delete the weight from the list.

Repeat the second step untal all the verices are connected, and then STOP.
The weighted graph obtained is a miminum connector, and the sum of the weights on its edges is the total weight of the minimum connector.

Prim's Algorithm; page 188

START with a finite set of vertices, where each pair of vertices is joined by a weighted edge.

- Choose and draw any vertex
- Find the edge of least weight joining a drawn vertex to a vertex *not* currently drawn. Draw this weighted edge and the corresponding new vertex.

Repeat the second step until all the vertices are connected, then STOP

Theorem 8.1; page 190

Theorem

Prim's and Kruskal's algorithm always produce a spanning tree of minimum weight.

Travelling Salesman Problem; page 191

The TRAVELLING SALESMAN PROBLEM is: Given a weighted complete graph, find a minimum-weight Hamiltonian cycle.

Travelling Salesman Problem - Upper Bound; page 192

START with a finite set of vertices, where each pair of vertices is joined by a weighted edge.

- Choose any vertex and find a vertex joined to it by an edge of minimum weight. Draw these two vertices and join them with two edges to form a *cycle*; give the cycle a clockwise orientation.
- Find a vertex NOT currently drawn joined by an edge of least weight to a vertex already drawn. Insert this new vertex into the cycle in front of the *nearest* already-connected vertex.

Repeat the second step until all of the vertices are joined by a cycle, then STOP.
The weighted cycle obtained is a Hamiltonian cycle, and its total weight - given by the sum of the weights on its edges - is an upper bound for the solution to the travelling salesman problem.

Travelling Salesman Problem - Lower Bound; page 196

- Choose any vertex v and remove it from the graph.
- Find a minimum spanning tree connecting the remaining vertices, and calculate its total weight w.
- Find the two smallest weights, w_1 and w_2, of the edges incident with v.
- Calculate the lower bound $w + w_1 + w_2$.

Homework Chapter 8

Problems:
8.1 – 8.5

Exercises:
8.1 – 8.6
8.7–8.15

Fleury's Algorithm; page 203

START with an Eulerian graph G

- Choose a starting vertex
- Starting from the current vertex, traverse any available edge, choosing a bridge only if there is no alternative. Then erase that edge and any isolated vertex.

Repeat the second step until there are no more edges, then STOP.

Shortest Path Algorithm; page 211

START Assign a potential of 0 to the start vertex S.

GENERAL STEP Consider the vertex (or vertices) just assigned a potential. For each such vertex v, consider each vertex w that can be reached from v along an arc vw, and assign w the label
(potential of v) + (distance vw)
unless w already has a *smaller* label assigned from an earlier iteration.
When all such vertices w have been labelled, choose the smallest vertex label that is not already a potential, and make it a potential at each vertex where it occurs.

Shortest Path Algorithm Continued; page 211

REPEAT the general step with the new potential.

STOP when the terminal vertex T has been assigned a potential.

To find a shortest path, trace backwards from T and include an arc vw whenever
(potential of w) - (potential of v) = distance vw
until S is reached.

Longest Path Algorithm

START Assign a potential of 0 to the start vertex S; label each vertex v reached only from S with the distance from S to v and make all these labels potentials.

GENERAL STEP Consider all vertices which can be reached *only* from vertices of known potential. For each such vertex w that can be reached from v along an arc vw, and assign w the label
(potential of v) + (distance vw)
unless w already has a *larger* label;
When all such arcs vw have been considered make the label at w a potential.

Longest Path Algorithm Continued

REPEAT the general step with the new potentials.

STOP when the terminal vertex T has been assigned a potential; this is the longest distance from S to T

To find a longest path, trace backwards from T and include an arc vw whenever
(potential of w) - (potential of v) $=$ distance vw
until S is reached.

Scheduling

Consider S the start of a project, and T denotes the termination of the project.

The vertices represent intermediate stages, or events. The arcs represent **activities**, and the weight of the arc represents the time needed to carry out the activity.

The length of the longest path represents the amount of time needed to complete the project. The arcs on the longest path need to be completed on time, and hence it is known as a **critical path**.

Scheduling Continued

The earliest start time for an activity XY is the length of the longest path from S to X.
(This is the potential assigned to X in the longest path algorithm)

The latest start time for an activity XY is
(total time for the project) - (length of the longest path from X to T via XY)

The float time for an activity is the difference between the earliest and latest start times.
(The float time for any activity on a critical path is zero (0)).

Chinese Postman Problem; page 212

The Chinese Postman problem is: Find a closed walk of minimum total weight that includes every edge at least once.

Homework Chapter 9

Problems:
9.1, 9.2 (do longest path & scheduling as well), 9.3

Exercises:
9.1 – 9.2
9.3– 9.5
Find longest path and schedule the graphs for 9.3– 9.5
9.6–9.7

Definitions; page 217

A graph is **connected** if there is a path between each pair of vertices, and is **disconnected** otherwise.

Every disconnected graph can be split up into a number of connected subgraphs, called **components**

A digraph is **connected** if its underlying graph is a connected graph, and it is **disconnected** otherwise.

A digraph is **strongly connected** if there is a path between each pair of vertices.

Definition; page 219

The **edge connectivity** $\lambda(G)$ of a connected graph G is the *smallest* number of edges whose removal disconnects G.

Definition; page 220

A **cutset** of a connected graph G is a set S of edges with the following two properties:

- removal of all the edges of S disconnects G.
- removal of some but not all of the edges in S does not disconnect G.

Definition; page 222, 223

The **connectivity** (or **vertex connectivity**) $\kappa(G)$ of a connected graph G (other than a complete graph) is the *smallest* number of vertices whose removal disconnects G.

The **connectivity** $\kappa(K_n)$ of the complete graph K_n is $n-1$.

Definition; page 223

A **vertex cutset** of a connected graph G is a set S of vertices with the following two properties:

- removal of all the vertices of S disconnects G.
- removal of some but not all of the vertices in S does not disconnect G.

Theorem 10.1; page 224

Theorem

Let G be a connected graph with smallest vertex degree $\delta(G)$. Then

$$\kappa(G) \;\leq\; \lambda(G) \;\leq\; \delta(G)\,.$$

Definition; page 226

Let G be a connected graph, and let s and t be vertices of G.

A path between s and t is an st-**path**.

Two or more st-paths are **edge-disjoint** if they have no edges in common, and **vertex-disjoint** if they have no vertices in common, other than s and t.

Definition; page 227

Let G be a connected graph, and let s and t be vertices of G.

Certain *edges* **separate** s **from** t if the removal of these edges destroys all paths between s and t.

Certain *vertices* **separate** s **from** t if the removal of these vertices destroys all paths between s and t.

Theorem 10.2; page 229

Theorem (Menger's Theorem for Graphs (Edge Form))

Let G be a connected graph, and let s and t be vertices of G. Then the maximum number of edge-disjoint st-paths is equal to the minimum number of edges separating s from t.

Corollary; page 230

Corollary (of Menger's Theorem for Graphs - Edge Form)

A connected graph has edge connectivity ℓ if and only if there are ℓ or more edge-disjoint paths between each pair of vertices in G, and there are exactly ℓ edge-disjoint paths between at least one pair of vertices.

Definition; page 237

The average of the vertex degrees is $\frac{2E}{V}$.

A graph G is said to have **optimal connectivity** if $\kappa(G) = \lambda(G) = \delta(G) = \frac{2E}{V}$.

Since $\kappa(G) \leq \lambda(G) \leq \delta(G) \leq \frac{2E}{V}$, it is enough to show that $\kappa(G) = \frac{2E}{V}$.

Homework Chapter 10

Problems:
10.1 – 10.7, 10.11 – 10.12

Exercises:
10.1 – 10.3
10.4
10.6–10.8

Note: in 10.6; *Links* are edges, and *exchanges* are vertices.

Definition; page 244

A graph G is **planar** if it can be drawn in the plane in such a way that no two edges meet except at a vertex with which they are both incident. Any such drawing is called a planar drawing of G

A graph G is **non-planar** if no plane drawing of G exists.

Definition; page 248

Let G be a planar graph. Then any plane drawing of G divides the set of points of the plane not lying on G into regions, called **faces**; one face is of infinite extent and is the **infinite face**.

Definition; page 249

Let G be a connected planar graph, and let f be any face of a plane drawing of G. Then the **degree of** f, denoted by **deg** f, is the number of edges encountered in a walk around the boundary of the face f.

If all faces have the same degree g,then G is **face-regular of degree** g.

Theorem 11.1; page 249

Theorem (Handshaking Lemma for Planar Graphs)

In any plane drawing of a planar graph, the sum of all the face degrees is equal to twice the number of edges.

Theorem 11.2; page 251

Theorem (Euler's Formula for Planar Graphs)

Let G be a connected planar graph, and let n, m, and f denote, respectively, the number of vertices, edges and faces in a plane drawing of G. Then

$$n - m + f = 2$$

Also known as

$$V - E + F = 2$$

Corollary 11.1, 11.2, 11.3; page 253, 254, 255

Corollary

Let G be a simple connected planar graph with $V(\geq 3)$ vertices and E edges. Then $E \leq 3V - 6$

Corollary

Let G be a simple connected planar graph with $V(\geq 3)$ vertices and E edges and no triangles. Then $E \leq 2V - 4$

Corollary

Let G be a simple connected planar graph. Then G contains a vertex of degree 5 or less.

Theorem 11.3; page 261

Theorem (Kuratowski's Theorem)

A graph is planar if and only if it contains no subdivision of K_5 or $K_{3,3}$.

Theorem 11.4; page 262

Theorem

A graph is planar if and only if it contains no subgraph that has K_5 or $K_{3,3}$ as a contraction.

Definition; page 264

Let G be a connected planar graph. Then a **dual graph** G^* is constucted from a planar drawing of G as follows.

Draw one new vertex in each face of the plane drawing: these are the vertices of G^*.

For each edge e of the plane drawing, draw a line joining the vertices of G^* in the faces on either side of e; these lines are the edges of G^*.

Theorem 11.5; page 266

Theorem

Let G be a plane drawing of a connected planar graph with V vertices, E edges and F faces.
Then G^* has F vertices, E edges and V faces.

Correspondence between G and G^*; page 267

plane drawing of G	dual graph G^*
an edge of G	an edge of G^*
a vertex of degree k in G	a face of degree k in G^*
a face of degree k in G	a vertex of degree k G^*
a cycle of length k in G	a cutset of G^* with k edges
a cutset of G with k edges	a cycle of length k in G^*

Theorem 11.6, 11.7; page 268

Theorem

Let G^* be a connected planear graph with F faces and E edges, and with no cutsets with 1 or 2 edges.
Then $E \leq 3F - 6$

Theorem

Let G^* be a connected planear graph with no cutsets with 1 or 2 edges. Then G^* has a face of degree 5 or less.

Definition; page 269

A **regular polyhedron** is a convex polyhedron in which all of the polygonal faces are congruent regular polygons, and each vertex has exactly the same arrangement of polygons around it.

Theorem 11.8 ; page 270

Theorem (Euler's Polyhedron Formula)

Let V, E and F denote, respectively, the numbers of vertices, edges and faces of a convex polyhedron. Then

$$V - E + F = 2$$

Theorem 11.9 ; page 271

Theorem (Handshaking Lemma for Polyhedra)

In any polyhedron, the sum of all the faces degrees is equal to twice the number of edges.

Theorem 11.10 ; page 272

Theorem

There are only five regular polyhedra.

The five regular polyhedra are :

the tetrahedron	having 4 triangular faces,
the cube	having 6 square faces,
the octagon	having 8 triangular faces,
the dodecahedron	having 12 pentagonal faces,
the icosahedron	having 20 triangular faces.

Homework Chapter 11

Problems:
11.1 – 11.20

Exercises:
11.1 – 11.2
11.3– 11.9
11.11
11.12– 11.14
(11.16– 11.17)

matchings

A *matching* is a mapping from some elements to some other elements, and a matching is *stable* whenever there is no element A of the first matched set that prefers an element B (that it's not matched to) of the second matched set, and at the same time B also prefers A over the one B is matched with.

Gale-Shapely Algorithm

- The Gale-Shapley algorithm involves a number of 'rounds' where each un-sudo-matched element in set A "proposes" to the most-preferred element in set B to whom he has not yet proposed.
- Each element in set B then considers all their proposals and tells the one he/she most prefers "Maybe" and all the rest of them "No".
- The element in set A that got the maybe is now sudo-matched to the element in set B.

Gale-Shapely Algorithm

- In each subsequent round, each un-sudo-matched element in set A proposes to the most-preferred element in set B to whom he has not yet proposed (the element may or may not already be sudo-matched),
  and the element in set B once again reply with one "maybe" to the most-preferred suitor and reject the rest (including for consideration her current sudo-match).
  This may mean that already sudo-matched elements in set B can "trade up", and already-sudo-matched elements in set A can be "jilted".
- Once every element in set A is sudo-matched, then that is now the matching.

Definition; page 279

Let G be a simple graph. A *k-colouring* of G is an assignment of at most k colours to the vertices of G in such a way that adjacent vertices are assigned different colours. If G has a k-colouring, then G is *k-colourable*.

The *chromatic number* of G, denoted $\chi(G)$, is the smallest number k for which G is k-colourable.

Theorem; page 281

Let G be a simple graph whose maximum vertex degree is d.

Then $\chi(G) \leq d + 1$.

Brooks' Theorem; page 282

Let G be a simple graph whose maximum vertex degree is d.

If G is neither a cycle graph with an odd number of vertices, nor a complete graph, then $\chi(G) \leq d$.

Theorem; page 284

The vertices of any simple connected planar graph G can be coloured with six (or fewer) colours in such a way that adjacent vertices are coloured differently.

Theorem; page 285

The vertices of any simple connected planar graph G can be coloured with five (or fewer) colours in such a way that adjacent vertices are coloured differently.

Four Colour Theorem; page 288

The vertices of any simple connected planar graph G can be coloured with four (or fewer) colours in such a way that adjacent vertices are coloured differently.

Definition; page 304

Let G be a graph without loops. A *k-edge colouring* of G is an assignment of at most k colours to the edges of G in such a way that any two edges meeting at a vertex are assigned different colours. If G has a k-edge colouring, then G is *k-edge colourable*.

The *chromatic index* of G, denoted $\chi'(G)$, is the smallest number k for which G is k-edge colourable.

Vising's Theorem; page 307

Let G be a simple graph whose maximum vertex degree is d.

Then $d \leq \chi'(G) \leq d + 1$.

Vising's Theorem (Extended version); page 307

Let G be a graph whose maximum vertex degree is d, and let h be the maximum number of edges joining a pair of vertices.

Then $d \leq \chi'(G) \leq d + h$.

Greedy algorithm for vertex colouring; page 288

- Start with a Graph G and a list of colours $1, 2, 3, \ldots$.
- Label the vertices $a, b, c, \ldots$ in any manner.
- Identify the uncoloured vertex labelled with the earliest letter in the alphabet; colour it with the first colour in the list not used for any adjacent coloured vertex.

Greedy algorithm for edge colouring; page 313

- Start with a Graph G and a list of colours $1, 2, 3, \ldots$.
- Label the edges $a, b, c, \ldots$ in any manner.
- Identify the uncoloured edge labelled with the earliest letter in the alphabet; colour it with the first colour in the list not used for any coloured edge that meets it at a vertex.