

Let's be safe

Please sit so that there is at least one empty set between you and any student on your left or right.

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Preface to the course

An acknowledgement of country

We acknowledge and celebrate the Ngunnawal people, the First Australians on whose traditional lands we meet. We pay our respect to the elders, past and present.

Course material

The lecture notes for the course are intended to be self-contained. Notes will be posted to our Wattle page immediately after the lecture, if not before. For most students, these notes, the workshop materials and the web will be sufficient.

If you would like an alternative presentation of the material, our optional text is [1].

- Copies of the text are on 2-hour loan in the library.
- From the publisher: “Students who wish to purchase their own eBook copies they can do so through <https://au.cengage.com/> Use the coupon code TENOFF to receive a discount at checkout.”

Credits

This course has been developed by a number of mathematicians over more than a decade. Contributing authors include:

- Judy-anne Osborne
- Pierre Portal
- Malcolm Brooks
- Adam Piggott

The mathematics in this course has been developed over millennia. Credit for discovery will be given only occasionally.

Other notes

- You must check the Wattle page, particularly the announcements, often.
- MATH1005 Assessment: workshop participation and weekly assignments starting in week 3, a mid-semester exam in week 7, a final exam.
- MATH6005 Assessment: weekly assignments starting in week 3, three graduate assignments, a mid-semester exam in week 7, a final exam.
- All students should sign up for a workshop through the course Wattle page.
- In weeks 3 and 8, the Monday lecture will not be held and Monday workshops will be rescheduled.

Why study discrete
mathematical models?

What are discrete mathematical models?

Discrete mathematical models are abstract representations of processes and objects, the steps or units of which can be indexed by the non-negative integers. In particular, we avoid continua (like the open interval $(0, 1)$).

Discrete mathematics is the study of discrete mathematical models.

Q: Is completing a Sudoku puzzle an exercise in discrete mathematics?

Computing and discrete mathematical models

Discrete mathematics and computer science go hand-in-hand because ...

Q: Describe an abstract model of the memory of a computer. That is, how do you think about memory?

Q: Do you think about computers working in continuous time or discrete time (“clock cycles”)?

Q: Can the interval $(0, 1)$ be modeled inside a computer?

Discrete mathematics and computer science

In 1974, Knuth wrote:

“Discrete mathematics, especially combinatorial theory, has been given an added boost by the rise of computer science, in addition to all the other fields in which discrete mathematics is currently being extensively applied.” [2]

This paper by Knuth is now available in the Week 1 section of our Wattle page. Please read the first 4 sections (pages 323-329) of the paper before lecture 2.

Discrete mathematics and computer scientists

Let's consider the educational background of some
[A. M. Turing Award](#) laureates.

- Donald Knuth (1974)
- Andrew Chi-Chih Yao (2000)
- Barbara Liskov (2008)
- Shafi Goldwasser(2012)
- David Patterson (2017)

References

- [1] Susanna S. Epp. *Discrete Mathematics with Applications: Metric Version*. Cengage Learning, 5th edition, 2019. ISBN: 9780357114087.
- [2] Donald E. Knuth. Computer science and its relation to mathematics. *Amer. Math. Monthly*, 81:323–343, 1974.

Section A: The language of mathematics and computer science

Part 1: Logic

Statements: The basic unit of logic

A **statement** is a sentence that is true or false but not both.

Statements: The basic unit of logic

A **statement** is a sentence that is true or false but not both.

Some sentences that are statements:

- Australia is in the Southern Hemisphere.
- Tropical storms spin clockwise in the Southern Hemisphere.
- Canberra is in New Zealand.
- $3 > 2$

Some sentences that are not statements:

- How are you going? (question)
- Canberra is a better city than Sydney. (ambiguous)
- Wake up! (instruction/advice)
- This statement is false. (neither true nor false; self-referencing)

Compound statements

Logical connectives such as ‘and’, ‘or’, ‘not’, ‘if-then’, ‘if and only if’ may be used to glue the statements together to make new statements.

Compound statements are statements built from other statements using logical connectives.

Example: Australia is in the Southern Hemisphere and tropical storms spin clockwise in the Southern Hemisphere.

Example: If 2 divides 4238 and 3 divides 4238, then 6 divides 4238.

Compound statements

Sometimes the way a compound statement is built from other statements is less clear, because our everyday use of language does not make the logical structure obvious. For example, the word ‘not’ sometimes causes confusion because of its placement.

Examples of compound statements

Q: In each of the following examples, identify clearly the way the statement is built from other statements using logical connectives:

- Australia is in the Southern hemisphere and tropical storms spin clockwise in the Southern Hemisphere.
- Australia is in the Southern Hemisphere and Canberra is not in New Zealand.
- I don't like either the one or the other.

Compound statements

Takeway: What makes for nice spoken or written language, may not necessary make the logical structure of a compound statement most clear. Worse, the everyday use of language can be ambiguous.

Mathematicians and computer scientists must agree upon the meaning of logical connectives, and must be proficient at recognising the way that compound statements are built from simpler statements, so that we can communicate effectively (human-to-human, and human-to-machine).

How to introduce a symbol for a statement

When you first learned algebra, you learned that it was useful to use variables to make abstract statements about relationships between numbers. In logic, we sometime save time and space, and clarify the logical structure of a compound statement, by introducing symbols as names for statements.

For example, we write

p : Australia is in the Southern Hemisphere

to mean that the symbol p now represents the statement “Australia is in the Southern Hemisphere.”

Statement variables and statement forms

We can also use variables to represent arbitrary statements. When a variable represents an arbitrary statement, it is called a **statement variable**.

We have symbols for various logical connectives (such as \neg for ‘not’, \wedge for ‘and’, \vee for ‘or’). We will define these properly in a moment.

An expression built using statement variables, parentheses and logical connectives is called a **statement form** if the expression becomes a statement when actual statements are substituted for the component statement variables.

Examples of statement forms

Example: Let p and q be statement variables.

The expression $q \vee \neg(p \wedge q)$ is a statement form.

The expression $p \neg \wedge q$ is not a statement form (just like $x + \times y$ is not an algebraic expression).

About statement forms

The value of the algebraic expression $x \times y + y$ depends on the values taken by x and y . You need to understand the multiplication and addition operations, and the order of precedence among operations, to figure out the value of the expression.

Analogously, the **truth value** of the statement form $q \vee \neg(p \wedge q)$ depends on the truth values taken by p and q . You need to understand the logical connectives \neg , \wedge , \vee and the order of precedence among them to figure out the truth value of the statement form.

Truth tables

We are now ready to define logical connectives. We shall do so using truth tables. A **truth table** records the truth value taken by a statement form for each possible combination of truth values taken by the statement variable appearing in the statement form.

A logical connective: NOT

If p is a statement variable, the **negation** of p is read “not p ”, denoted $\neg p$, and defined by the following truth table:

p	$\neg p$
T	F
F	T

A logical connective: AND

If p and q are statement variables, the **conjunction** of p and q is read “ p and q ”, denoted $p \wedge q$, and defined by the following truth table:

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

A logical connective: OR

If p and q are statement variables, the **disjunction** of p and q is read “ p or q ”, denoted $p \vee q$, and defined by the following truth table:

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

A logical connective: XOR

If p and q are statement variables, the **exclusive disjunction** of p and q is read “ p or q but not both”, denoted $p \oplus q$, and defined by the following truth table:

p	q	$p \oplus q$
T	T	F
T	F	T
F	T	T
F	F	F

Note: Grammatically, the words ‘and’ & ‘but’ are both conjunctions, so are both interpreted as AND in logic. You could read $p \oplus q$ as “ p or q and not both.”

A logical connective: IMPLIES

If p and q are statement variables, the **conditional** of q by p is read “if p then q ” or “ p implies q ”, denoted $p \rightarrow q$, and defined by the following truth table:

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

There is extra vocabulary associated with conditional. In the statement form $p \rightarrow q$, we call p the **hypothesis** (or antecedent) and q the **conclusion** (or consequent).

Understanding IMPLIES

Pay careful attention to the truth table for a conditional.
On first sight, some find the last two lines surprising.
Spend some time making sure you know the details of
the definition.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

It sometimes helps to use language to emphasize *why* a conditional is true. When the hypothesis in a conditional is false, we say that the conditional is **vacuously true**.

A logical connective: IFF

If p and q are statement variables, the **biconditional** of p and q is read “ p if and only if q ”, denoted $p \leftrightarrow q$, and defined by the following truth table:

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Order of precedence among logical connectives

When evaluating a statement form:

- Obey parentheses over any precedence rule
- Evaluate \neg first
- Evaluate \wedge , \vee and \oplus second. When two or more are present, parentheses may be needed
- Evaluate \rightarrow and \leftrightarrow third. When both are present, parentheses may be needed.

When writing statement forms, you must use parentheses to avoid ambiguity. For example, you should write $p \wedge (q \vee r)$ rather than $p \wedge q \vee r$.

Example: Truth table for a complex statement form

Q: Write a truth table for $(p \wedge q) \vee (\neg p \wedge \neg r)$.

First we note that:

- Our table needs a row for each combination of truth values among the statement variables. Since there are three statement variables in the statement form, our table will need $2^3 = 8$ rows.
- We need to obey the order of precedence among logical connectives as we evaluate the truth value of the statement form.

We shall present two ways to lay out our work.

Method 1: Use construction columns

p	q	r	$p \wedge q$	$\neg p$	$\neg r$	$\neg p \wedge \neg r$	$(p \wedge q) \vee (\neg p \wedge \neg r)$
T	T	T	T	F	F	F	T
T	T	F	T	F	T	F	T
T	F	T	F	F	F	F	F
T	F	F	F	F	T	F	F
F	T	T	F	T	F	F	F
F	T	F	F	T	T	T	T
F	F	T	F	T	F	F	F
F	F	F	F	T	T	T	T

An advantage of this method is that you leave a record of your working. A disadvantage is that it can take a lot of room on the page.

Method 2: Align truth values under connectives

p	q	r	$(p \wedge q) \vee (\neg p \wedge \neg r)$
T	T	T	$T \quad T \quad F \quad F \quad F$
T	T	F	$T \quad T \quad F \quad F \quad T$
T	F	T	$F \quad F \quad F \quad F \quad F$
T	F	F	$F \quad F \quad F \quad F \quad T$
F	T	T	$F \quad F \quad T \quad F \quad F$
F	T	F	$F \quad T \quad T \quad T \quad T$
F	F	T	$F \quad F \quad T \quad F \quad F$
F	F	F	$F \quad T \quad T \quad T \quad T$

If you use this method:

- Be careful because your work leaves less of a record of how you did it.
- Bold, circle or highlight the column with the final result.

Tautologies

A statement form that is true regardless of the truth values of its variables is called a **tautology**. A statement whose form is a tautology is also called a **tautology**.

We can show that a statement form is a tautology by making a truth table. We can show that a statement is a tautology by making a truth table for the corresponding statement form.

Showing that a statement (form) is a tautology

Example: The statement form $p \vee \neg p$ is a tautology, as shown by:

p	$\neg p$	$p \vee \neg p$
T	F	T
F	T	T

Example: The statement “ $\sqrt{2}$ is negative or non-negative” is a tautology because:

- it is an instance of the statement form $p \vee \neg p$ with $p : \sqrt{2}$ is negative,
- the statement form $p \vee \neg p$ is a tautology.

Contradiction

A statement form that is false regardless of the truth values of its variables is called a **contradiction**. A statement whose form is a contradiction is also called a **contradiction**.

We can show that a statement form is a contradiction by making a truth table. We can show that a statement is a contradiction by making a truth table for the corresponding statement form.

Logical equivalence

When two statement forms f, g have identical truth tables we say they are **logically equivalent** and write $f \equiv g$.

(ALTERNATIVE DEFINITION: When f, g are statement forms and $f \leftrightarrow g$ is a tautology, we say that f and g are **logically equivalent** and write $f \equiv g$.)

Knowing some logical equivalences allows us to replace complicated statement forms by simpler, logically equivalent, statement forms.

Some well-known logical equivalences are shown on the next slide.

From page 49 of the optional text

Theorem 2.1.1 Logical Equivalences

Given any statement variables p , q , and r , a tautology t and a contradiction c , the following logical equivalences hold.

1. Commutative laws:	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
2. Associative laws:	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
3. Distributive laws:	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
4. Identity laws:	$p \wedge t \equiv p$	$p \vee c \equiv p$
5. Negation laws:	$p \vee \sim p \equiv t$	$p \wedge \sim p \equiv c$
6. Double negative law:	$\sim(\sim p) \equiv p$	
7. Idempotent laws:	$p \wedge p \equiv p$	$p \vee p \equiv p$
8. Universal bound laws:	$p \vee t \equiv t$	$p \wedge c \equiv c$
9. De Morgan's laws:	$\sim(p \wedge q) \equiv \sim p \vee \sim q$	$\sim(p \vee q) \equiv \sim p \wedge \sim q$
10. Absorption laws:	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
11. Negations of t and c :	$\sim t \equiv c$	$\sim c \equiv t$

Note: The text uses \sim for negation instead of \neg .

Example: An equivalent form for XOR

Q: Justify the earlier claim that $p \oplus q \equiv (p \vee q) \wedge \neg(p \wedge q)$.

Example: An equivalent form for XOR

Q: Justify the earlier claim that $p \oplus q \equiv (p \vee q) \wedge \neg(p \wedge q)$.

We compute a truth table:

p	q	$p \oplus q$	$p \vee q$	$p \wedge q$	$\neg(p \wedge q)$	$(p \vee q) \wedge \neg(p \wedge q)$
T	T	F	T	T	F	F
T	F	T	T	F	T	T
F	T	T	T	F	T	T
F	F	F	F	F	T	F

Because the entries in column 3 and column 7 agree in every row, the truth table above establishes that $p \oplus q \equiv (p \vee q) \wedge \neg(p \wedge q)$.

Example: Another equivalent form for XOR

Q: Show that $p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$

Example: Another equivalent form for XOR

Q: Show that $p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$

We compute a truth table:

p	q	$p \oplus q$	$\neg p$	$\neg q$	$p \wedge \neg q$	$\neg p \wedge q$	$(p \wedge \neg q) \vee (\neg p \wedge q)$
T	T	F	F	F	F	F	F
T	F	T	F	T	T	F	T
F	T	T	T	F	F	T	T
F	F	F	T	T	F	F	F

Because the entries in column 3 and column 8 agree in every row, the truth table above establishes that $p \oplus q \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$.

Your homework between now and lecture 2

- Master the vocabulary introduced in this lecture and practice making truth tables for complex statement forms
- Read the first 4 sections (pp. 323-329) of Knuth's paper referenced on slide #5 (the paper is available in Wattle), and prepare bullet points in response to the following “Identify three interesting observations/points made by Knuth in the first 4 sections of the paper.”
- Use Sudoku.com. Announce your best time (and the level of difficulty) on the course Wattle forum.

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section A: The language of mathematics and computer science

Part 1: Logic (continued)

A scenario (from p. 54 of our optional text) to explain why our definition of implies is reasonable

Suppose you go to interview for a job at a store and the owner of the store makes you the following promise:

If you show up for work Monday morning, then you will get the job.

In which of the following situations has the store owner been proven a liar?

- A. You show up for work Monday morning and you get the job.
- B. You show up for work Monday morning and you don't get the job.
- C. You do not show up for work Monday morning and you get the job.
- D. You do not show up for work Monday morning and you don't get the job.

A scenario (cont.)

The store owner did not say anything about what would happen if you don't show up for work Monday morning. Therefore, the only situation in which the owner has been proven a liar is (B).

With

p : You show up for work Monday morning

q : You get the job,

the store owner's promise is represented by $p \rightarrow q$.

The store owner is proven a liar only if the statement they made has been demonstrated to be false. Now $p \rightarrow q$ is false exactly when $\neg(p \rightarrow q)$ is true. It therefore seems reasonable to say that $\neg(p \rightarrow q)$ is true only in situation (B).

A scenario (cont.)

The scenario demonstrates that the following truth table is “reasonable”:

p	q	$\neg(p \rightarrow q)$
T	T	F
T	F	T
F	T	F
F	F	F

It follows that the truth table for \rightarrow as given on slide #22 is “reasonable.”

More vocabulary associated to $p \rightarrow q$

The statement $p \rightarrow q$ is read aloud in each of the following ways:

- p implies q
- if p then q
- p only if q
- q if p

Please note the potential for confusion.

It is helpful to have words to describe the relationships between certain implications. We have

- The **converse** of $p \rightarrow q$ is $q \rightarrow p$
- The **inverse** of $p \rightarrow q$ is $\neg p \rightarrow \neg q$
- The **contrapositive** of $p \rightarrow q$ is $\neg q \rightarrow \neg p$

Examples using the language of contrapositive, converse and inverse

Consider the statement:

P: You work hard
q: you will pass the exam

If you work hard, then you will pass the exam.

$$P \rightarrow q$$

The contrapositive is:

If you fail the exam, then you have not worked hard.

$$\neg q \rightarrow \neg p$$

The converse is:

If you pass the exam, then you have worked hard.

$$q \rightarrow p$$

The inverse is:

If you don't work hard, then you will fail the exam.

$$\neg p \rightarrow \neg q$$

More vocabulary associated to $p \rightarrow q$

Which, if any, of the following statements concerning $p \rightarrow q$ are true:

- A. The converse of the inverse is the contrapositive.
- B. The inverse of the converse is the contrapositive.
- C. The inverse of the contrapositive is the converse.
- D. The converse of the contrapositive is the inverse.
- E. The statement is logically equivalent to its contrapositive.
- F. The inverse is logically equivalent to the converse.
- G. The statement is not logically equivalent to its converse.

More vocabulary associated to $p \rightarrow q$

Which, if any, of the following statements concerning $p \rightarrow q$ are true:

- A. The converse of the inverse is the contrapositive.
- B. The inverse of the converse is the contrapositive.
- C. The inverse of the contrapositive is the converse.
- D. The converse of the contrapositive is the inverse.
- E. The statement is logically equivalent to its contrapositive.
- F. The inverse is logically equivalent to the converse.
- G. The statement is not logically equivalent to its converse.

Answer: All of the above are true. Statements E, F, and G are particularly important.

Necessary and sufficient conditions

p is a **sufficient condition** for q means $p \rightarrow q$.

p is a **necessary condition** for q means $\neg p \rightarrow \neg q$.

Since $(\neg p \rightarrow \neg q) \equiv (q \rightarrow p)$, we can also say:

p is a **necessary condition** for q means $q \rightarrow p$;

p is a **necessary and sufficient condition** for q means
 $p \leftrightarrow q$.

In view of the above terminology, statements of the form $p \rightarrow q$ and $p \leftrightarrow q$ are often referred to as **conditional statements**.

Examples using the language of necessary and sufficient conditions

Examples:

- Being older than 16 is a necessary condition for being allowed to drive. It is not sufficient.
- Having a drivers license is a necessary and sufficient condition for being allowed to drive.
- Being a taxi driver is a sufficient condition for being allowed to drive. It is not necessary.

Negating compound statements

For statement variables p and q , the following logical equivalences hold:

$$\neg(\neg p) \equiv p \tag{1}$$

$$\neg(p \wedge q) \equiv \neg p \vee \neg q \tag{2}$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q \tag{3}$$

$$\neg(p \oplus q) \equiv (p \wedge q) \vee (\neg p \wedge \neg q) \tag{4}$$

$$\neg(p \rightarrow q) \equiv p \wedge \neg q \tag{5}$$

$$\neg(p \leftrightarrow q) \equiv (p \wedge \neg q) \vee (\neg p \wedge q) \tag{6}$$

RECALL: Each of these logical equivalences may be confirmed by demonstrating that the truth tables for the left-hand side and right-hand side match in every row.

Using logical equivalences to “simplify”

The right-hand side of each logical equivalence on the previous slide has the pleasant property that \neg only appears in front of statement variables. Using these rules, we can arrange to replace any compound statement by one in one \neg only appears in front of statement variables.

Example: Consider the compound statement

$$\begin{aligned} & \neg(\neg p \wedge q) \wedge (p \vee q) \\ & \equiv (\neg(\neg p) \vee \neg q) \wedge (p \vee q) && \text{Using (2)} \\ & \equiv (p \vee \neg q) \wedge (p \vee q) && \text{Using (1)} \end{aligned}$$

Using logical equivalences to “simplify”

Using the logical equivalences on slide #32 and #46 together (there is overlap), we can simplify further:

Example:

$$\begin{aligned}& \neg(\neg p \wedge q) \wedge (p \vee q) \\&\equiv (\neg(\neg p) \vee \neg q) \wedge (p \vee q) \\&\equiv (p \vee \neg q) \wedge (p \vee q) \\&\equiv p \vee (\neg q \wedge q) \\&\equiv p \vee (q \wedge \neg q) \\&\equiv p \vee c \\&\equiv p\end{aligned}$$

Using (2)

Using (1)

Using a Distributive law

Using a Commutative law

Using a Negation law

Using an Identity law

In the above, c represent a contradiction.

Arguments

Valid arguments

An **argument** is a sequence of statements, one of which (usually the last) is flagged as the **conclusion**. The other statements, called **premises**, are alleged to justify the conclusion.

Example: *The coffee argument*

If I arrive early at Uni, I have a coffee. If I have a coffee, I focus on the lecture. I do focus on the lecture because I arrive early at Uni.

Notation for describing the form of an argument

Consider an argument with premises p_1, \dots, p_n and conclusion q . The **form** of an argument is the expression

$$\underbrace{[p_1, p_2, \dots, p_n]}_{\text{List the premises}} \therefore q.$$

or is written

$$\begin{array}{c} p_1 \\ p_2 \\ \vdots \\ \hline p_n \\ q \end{array}$$

Let's see how this exposes the nature of an argument.

Identifying the structure of the coffee argument

The premises and conclusion of the coffee argument are:

p_1 : If I arrive early at Uni, I have a coffee

p_2 : If I have a coffee, I focus on the lecture.

p_3 : I arrive early at Uni

q : I focus on the lecture

and the form of the coffee argument is:

$$[p_1, p_2, p_3 \therefore q]$$

This doesn't expose the reasoning very effectively, but if we break down the premises by recognising them as compound statements...

Representing the structure of the coffee argument

With

- a: I arrive early at Uni
- b: I have a coffee
- q : I focus on the lecture

the form of the argument is

$$[a \rightarrow b, b \rightarrow q, a \therefore q]$$

or

$$\begin{array}{c} a \rightarrow b \\ b \rightarrow q \\ \hline a \\ \hline q \end{array}$$

Valid and invalid arguments

The purpose of representing the form of an argument is often to examine the validity of the argument.

An argument with the form $[p_1, p_2, \dots, p_n \therefore q]$ is **valid** if the statement form

$$(p_1 \wedge p_2 \wedge \cdots \wedge p_n) \rightarrow q$$

is a tautology.

An informal interpretation of this is that an argument is valid when the truth of the conclusion is guaranteed by the truth of the premises.

Establishing the validity of the coffee argument

a	b	q	$((a \rightarrow b) \wedge (b \rightarrow q) \wedge a) \rightarrow q$
T	T	T	$T \quad T \quad T \quad F \quad T$
T	T	F	$T \quad F \quad T \quad T \quad F$
T	F	T	$F \quad T \quad T \quad T \quad T$
T	F	F	$F \quad T \quad T \quad T \quad F$
F	T	T	$T \quad T \quad T \quad T \quad T$
F	T	F	$T \quad F \quad T \quad T \quad F$
F	F	T	$F \quad T \quad T \quad T \quad T$
F	F	F	$F \quad T \quad T \quad F$

So \rightarrow

Time saving tips:

- If all premises are T , we need the conclusion T .
- Once we find a false premise, we can stop work on the row. (also, if the conclusion is true we can stop work on the row)

Another Example

Consider the following argument:

If fair elections are held and the unemployment rate is below 10%, the civil unrest disappears. But civil unrest is on the rise, so the elections were unfair!

Let's examine the structure. With:

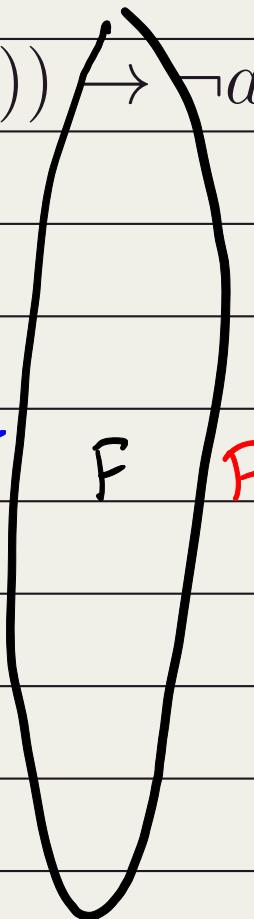
- a: Fair elections are held
- b: The unemployment rate is below 10%
- c: Civil unrest disappears

the argument has the form

$$[a \wedge b \rightarrow c, \neg c \therefore \neg a]$$

Checking the validity of the civil unrest argument

a	b	c	$((a \wedge b \rightarrow c) \wedge (\neg c)) \rightarrow \neg a$
T	T	T	
T	T	F	
T	F	T	
T	F	F	F T T F F
F	T	T	
F	T	F	
F	F	T	
F	F	F	



Checking the validity of the civil unrest argument

a	b	c	$((a \wedge b \rightarrow c) \wedge (\neg c)) \rightarrow \neg a$
T	T	T	
T	T	F	
T	F	T	
T	F	F	F \top \top F F
F	T	T	
F	T	F	
F	F	T	
F	F	F	

Since the fourth line of the truth table evaluated to F , the statement is not a tautology and the argument is invalid.

Fixing the civil unrest argument

Consider the following argument:

If fair elections are held and the unemployment rate is below 10%, then civil unrest disappears. But civil unrest is on the rise, so the elections were unfair or the unemployment rate is at least 10%.

Let's examine the structure. With:

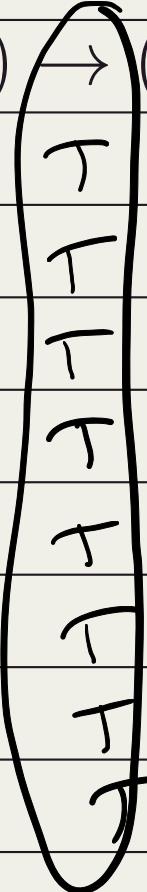
- a: Fair elections are held
- b: The unemployment rate is below 10%
- c: Civil unrest disappears

the amended argument has the form

$$[a \wedge b \rightarrow c, \neg c \therefore (\neg a \vee \neg b)]$$

Checking the validity of the civil unrest argument

a	b	c	$((a \wedge b \rightarrow c) \wedge (\neg c)) \rightarrow (\neg a \vee \neg b)$
T	T	T	$T \ T \ F \ F \quad T$
T	T	F	$T \ F \ F \ T \quad T$
T	F	T	$F \ T \ F \ F \quad T$
T	F	F	$F \ T \ T \ T \quad T \ FTT$
T	T	T	$T \ T \ F \ F \quad T$
T	T	F	$T \ F \ F \ T \quad T$
T	F	T	$F \ T \ F \ F \quad T$
T	F	F	$\neg \vdash T \ T \ T \quad T \ FTT$



Checking the validity of the civil unrest argument

a	b	c	$((a \wedge b \rightarrow c) \wedge (\neg c)) \rightarrow (\neg a \vee \neg b)$
T	T	T	
T	T	F	
T	F	T	
T	F	F	
T	T	T	
T	T	F	<i>see previous slide</i>
T	F	T	
T	F	F	

The truth table demonstrates that the statement is a tautology and the amended argument is valid.

Some standard arguments that are valid

Modus ponens	Modus tollens	Division into cases
$\begin{array}{c} p \rightarrow q \\ p \\ \hline q \end{array}$	$\begin{array}{c} p \rightarrow q \\ \neg q \\ \hline \neg p \end{array}$	$\begin{array}{c} p \vee q \\ p \rightarrow r \\ q \rightarrow r \\ \hline r \end{array}$
VALID	VALID	VALID
The simplest argument form. The others are based on this.	An implication is equivalent to its contrapositive.	At least one of the cases holds, and each case alone implies the conclusion.

Some standard errors (invalid arguments)

Converse error	Inverse error
$\begin{array}{c} p \rightarrow q \\ q \\ \hline p \end{array}$	$\begin{array}{c} p \rightarrow q \\ \neg p \\ \hline \neg q \end{array}$
INVALID	INVALID

An implication is **not** equivalent to its converse.

An implication is **not** equivalent to its inverse.

Identify the structure and determine validity

Example:

If interest rates rise, so does unemployment.

Interest rates are rising.

Therefore unemployment is rising.

P: interest rates rise
q: unemployment rises

$$P \rightarrow q$$

P

—————
q

Example:

If you studied hard for the exam you will have passed.

You did pass.

So you must have studied hard for the exam.

p: You studied hard for the exam

q: You passed the exam

q

—————
P

This is an example to the converse error

The argument is

INVALID

Identify the structure and determine validity

Example:

If Ned robbed a bank in Euroa, then he was in Victoria.

$$P \rightarrow Q$$

But he was in New South Wales.

$$\neg Q$$

So he didn't rob the bank.

$$\hline$$

P: Ned robbed a bank in Euroa

Q: Ned was in Victoria

Example:

The user has changed their status or they have uploaded pictures.

If status has been changed, the page needs to be refreshed.

If pictures have been uploaded, then the page needs to be refreshed too.

So the page needs to be refreshed.

See next page

hence

- j.: The user has changed their status
- k.: The user has uploaded pictures
- l.: The page needs to be refreshed

The argument was the form:

$$\frac{P \vee q}{\begin{array}{c} P \rightarrow r \\ q \rightarrow r \end{array}} r$$

This is an example of
Division into cases. The argument
is VALID

Identify the structure and determine validity

Example:

If we don't ban plastic bags, pollution will increase.

We have banned plastic bags, so pollution will not increase.

P: we don't ban plastic bags
q: Pollution will increase

The argument has the form

$$\frac{P \rightarrow q}{\neg P}$$

This is an example of the inverse error.

The argument is INVALID

Some final notes

The validity/invalidity of an argument says nothing about whether or not the premises are true. An argument can be valid even though some of the premises are false. The validity of the argument merely assures us that if the premises are true, then the conclusion will be true. (See Modus ponens).

You can reach the right conclusion by an invalid argument. The conclusion to an invalid argument may be true, even when the premises are all true. The conclusion is just not guaranteed to be true by the argument in this case. (See the converse error)

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section A: The language of mathematics and computer science

Part 1: Logic (continued)

Predicate logic

Predicates

A **predicate** is a sentence containing one or more variables, with the property that, when a value from a specified **domain** is given to each variable, the sentence becomes a statement. The specified domain is the **domain** of the predicate.

Example: Consider the predicate

$$p(x) = \text{"}x \text{ is a bird",}$$

defined over the domain of animals. If $x = \text{cockatoo}$, then $p(x)$ is true. We write $p(\text{cockatoo}) = T$. Further, $p(\text{shark}) = F$ and $p(17@\#)$ is undefined.

More examples

Perhaps the predicate $q(x)$ = “ x is allowed to view this file”, defined over a list of users, sounds more relevant to computer scientists.

For $r(x, y)$ = “ x has been at war with y ”, with x and y taking values from the set of countries, we have

$$r(\text{Iraq}, \text{USA}) = T$$

and

$$r(\text{Australia}, \text{Indonesia}) = F.$$

Quantifiers

There are two ways to turn a predicate $p(x)$ into a statement:

- specify a value for x ; or
- *quantify* x . (“quantify” = “express the ‘quantity’ of”)

The universal quantifier

The **universal quantifier**, \forall , is read “for all” (or “for every”, “for each”, “for any” etc.).

The **universal statement** $\forall x p(x)$ is read aloud “for all x (in the domain), $p(x)$ is true” or “ $p(x)$ is true for all x (in the domain)”.

The existential quantifier

The **existential quantifier**, \exists , is read “there exists” (or “for at least one”, etc.)

The **existential statement** $\exists x p(x)$ is read aloud “There exists an x (in the domain) such that $p(x)$ is true” or “ $p(x)$ is true for at least one x (in the domain)” or “ $p(x)$ is true for some x (in the domain)”.

The existential quantifier with uniqueness

The **existential quantifier with uniqueness**, $\exists!$, is read “there exists a unique” (or “for exactly one”, etc.)

The **existential statement** $\exists!x p(x)$ is read aloud “There exists a unique x (in the domain) such that $p(x)$ is true” or “ $p(x)$ is true for exactly one x (in the domain)”.

Examples

Q: Let $p(x)$ be the predicate “ x is a bird”, with x taking values from the domain {cockatoo, parrot, shark}. For each of the following statements, write out in words a translation of the statements and evaluate it.

1. $\forall x p(x)$
2. $\exists x p(x)$
3. $\exists!x p(x)$
4. $\exists!x \neg p(x)$

Examples

Recall that $p(x)$ is the predicate “ x is a bird”, with x taking values from the domain $\{\text{cockatoo, parrot, shark}\}$.

1. $\forall x p(x)$

The statement reads: “For all x in the set $\{\text{cockatoo, parrot, shark}\}$, x is a bird”. This is false because a shark is not a bird ($p(\text{shark}) = F$).

2. $\exists x p(x)$

The statement reads: “There is at least one x in the set $\{\text{cockatoo, parrot, shark}\}$ such that x is a bird”. This is true because a cockatoo is a bird ($p(\text{cockatoo}) = T$).

Examples

Recall that $p(x)$ is the predicate “ x is a bird”, with x taking values from the domain {cockatoo, parrot, shark}.

3. $\exists!x p(x)$

The statement reads: “There is exactly one x in the set {cockatoo, parrot, shark} such that x is a bird”.

This is false because there are two animals in the domain that are birds ($p(\text{cockatoo}) = p(\text{parrot}) = T$).

4. $\exists!x \neg p(x)$

The statement reads: “There is exactly one x in the set {cockatoo, parrot, shark} such that x is not a bird”. This is true because only one of the animals in the domain (shark) is not a bird.

Another example

Q: Let $q(x)$ be the predicate “ x is in Australia”, with x taking values from the domain

$$D = \{\text{Brisbane, Sydney, Melbourne, Adelaide, Perth}\}.$$

Evaluate each of the following statements

1. $\forall x q(x)$
2. $\exists x q(x)$
3. $\exists!x q(x)$
4. $\exists!x \neg q(x)$

Examples

Recall that $q(x)$ is the predicate “ x is in Australia”, with x taking values from the domain

$$D = \{\text{Brisbane, Sydney, Melbourne, Adelaide, Perth}\}.$$

1. $\forall x q(x)$ is true because every city in D is in Australia
2. $\exists x q(x)$ is true because $q(\text{Brisbane}) = T$.
3. $\exists!x q(x)$ is false because more than one city in D is in Australia
4. $\exists!x \neg q(x)$ is false because there are no cities in D that are not in Australia.

Example

With domain the set of users (of some online system), express the statements below symbolically, using the following notation:

$o(x)$: x is online.

$c(x)$: x has changed status.

$u(x)$: x has uploaded pictures.

1. All users are online.
2. No user has changed status.
3. All users who have changed status have uploaded pictures.
4. Some users have changed status.
5. Only one user has not uploaded pictures.

Example

With domain the set of users (of some online system), express the statements below symbolically, using the following notation:

$o(x)$: x is online.

$c(x)$: x has changed status.

$u(x)$: x has uploaded pictures.

1. All users are online. $\forall x o(x)$
2. No user has changed status. $\forall x \neg c(x)$
3. All users who have changed status have uploaded pictures. $\forall x c(x) \rightarrow u(x)$
4. Some users have changed status. $\exists x c(x)$
5. Only one user has not uploaded pictures. $\exists!x \neg u(x)$.

Notation: \Rightarrow and \Leftrightarrow

Let $p(x)$ and $q(x)$ be predicates and suppose the common domain of x is D .

- The notation $p(x) \Rightarrow q(x)$ is short for

$$\forall x \ p(x) \rightarrow q(x)$$

- The notation $p(x) \Leftrightarrow q(x)$ is short for

$$\forall x \ p(x) \leftrightarrow q(x)$$

WARNING: It is not uncommon for mathematicians to use \rightarrow and \Rightarrow interchangeably, as if they mean the same thing.

Order of precedence and quantification

This is perhaps best explained by example: the expression

$$\forall x \ p(x) \rightarrow q(x)$$

means

$$\forall x \ (p(x) \rightarrow q(x)).$$

Example

What is the negation of “all users are online”?

Answer: Not all users are online, *i.e.* at least one user is offline.

Symbolically: $\neg(\forall x p(x)) \equiv \exists x \neg p(x)$.

What is the negation of “some users have changed status”?

Answer: No user has changed status, *i.e.* all users have not changed status.

Symbolically: $\neg(\exists x q(x)) \equiv \forall x \neg q(x)$.

Example

Here is a more complicated example from mathematical analysis. The variables x and y take values from the set of real numbers; the variable ϵ and δ take values from the set of positive real numbers.

$$\begin{aligned} & \neg \left(\forall \epsilon \exists \delta \forall x \forall y \quad |x - y| < \delta \implies |x^2 - y^2| < \epsilon \right) \\ & \equiv \exists \epsilon \forall \delta \exists x \exists y \neg \left(|x - y| < \delta \implies |x^2 - y^2| < \epsilon \right) \\ & \equiv \exists \epsilon \forall \delta \exists x \exists y \quad |x - y| < \delta \wedge |x^2 - y^2| \geq \epsilon \end{aligned}$$

The order of quantification matters

$$(\forall x \exists y p(x, y)) \not\equiv (\exists y \forall x p(x, y))$$
$$(\exists x \forall y p(x, y)) \not\equiv (\forall y \exists x p(x, y))$$

Example:

Domains: a set of people and a set of countries.

$p(x, y)$ = “ x is a tall inhabitant of y ”

$\forall y \exists x p(x, y)$ says that each country has at least one tall inhabitant.

$\exists x \forall y p(x, y)$ says that there is a tall individual who lives in every country.

These two statements are not equivalent!

Logic circuits

A question

Do we have all of the logical connectives we need?

That is, suppose I present you with a truth table but I do not label the right-hand column with a compound statement. Can you construct a compound statement for which the given truth table is correct?

Said another way: Is every compound statement logically equivalent to a compound statement made using only AND, OR and NOT?

An example

Q: Find a compound statement to replace ?

p	q	r	?
T	T	T	F
T	T	F	T
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

In the truth table table, ? can be replaced by:

An example

Q: Find a compound statement to replace ?

p	q	r	?
T	T	T	F
T	T	F	T
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

In the truth table table, ? can be replaced by:

$$(p \wedge q \wedge \neg r) \vee (\neg p \wedge q \wedge r) \vee (\neg p \wedge \neg q \wedge r)$$

An example

A compound statement like the one we just wrote is said to be **disjunctive normal form**.

Following the plan used in the previous example, we see that every compound statement is logically equivalent to a compound statement made using only parentheses and the logical connectives \wedge , \vee , \neg .

We say that the set $\{\wedge, \vee, \neg\}$ is **functionally complete**.

CLAIM: The set $\{\wedge, \neg\}$ is functionally complete.

Q: How can we show this?

An example

A compound statement like the one we just wrote is said to be **disjunctive normal form**.

Following the plan used in the previous example, we see that every compound statement is logically equivalent to a compound statement made using only parentheses and the logical connectives \wedge , \vee , \neg .

We say that the set $\{\wedge, \vee, \neg\}$ is **functionally complete**.

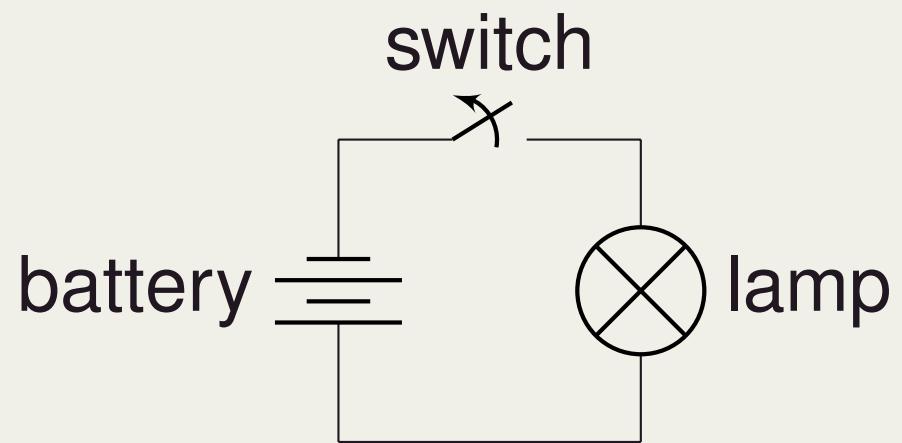
CLAIM: The set $\{\wedge, \neg\}$ is functionally complete.

Q: How can we show this?

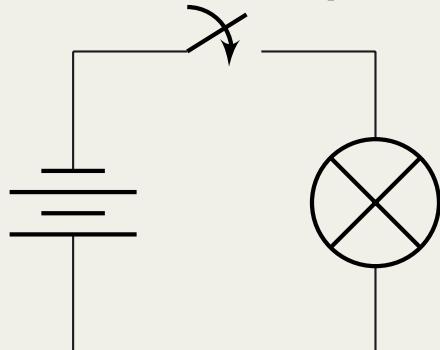
A: Show that $p \vee q \equiv \neg(\neg p \wedge \neg q)$

Circuits

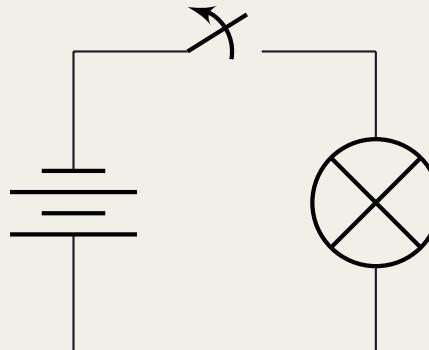
Here is a simple circuit:



It has two possible states:



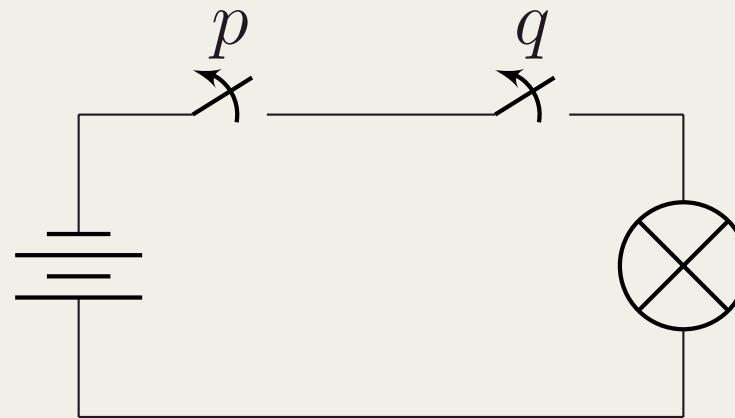
Light ON



Light OFF

The states correspond to the values **True** and **False** of the statement “the light is ON”.

Circuits: AND

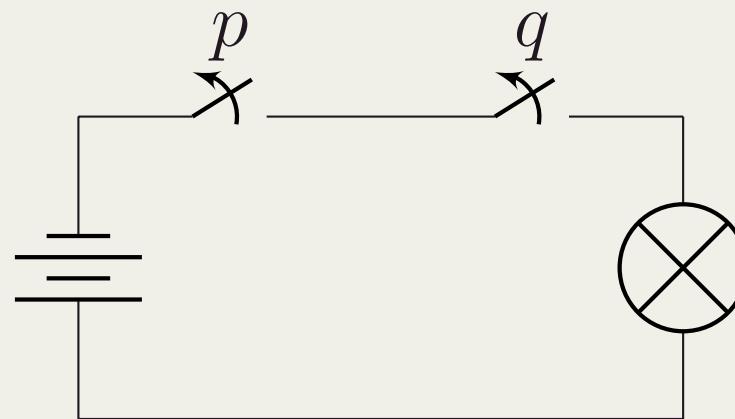


The behaviour of this circuit can be represented by a truth table (which coincides with the truth table for AND):

Switch p is ON	Switch q is ON	Light is ON
T	T	T
T	F	F
F	T	F
F	F	F

The AND gate

The circuit



is called an AND gate.

It takes two inputs:

- the state of switch p
(denoted by 1 for ON and 0 for OFF)
- the state of switch q

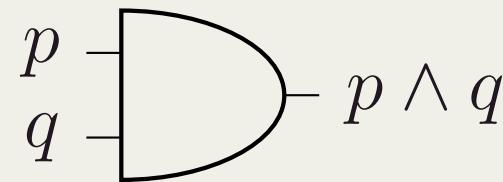
and produces an output:

- the state of the light bulb.

The AND gate

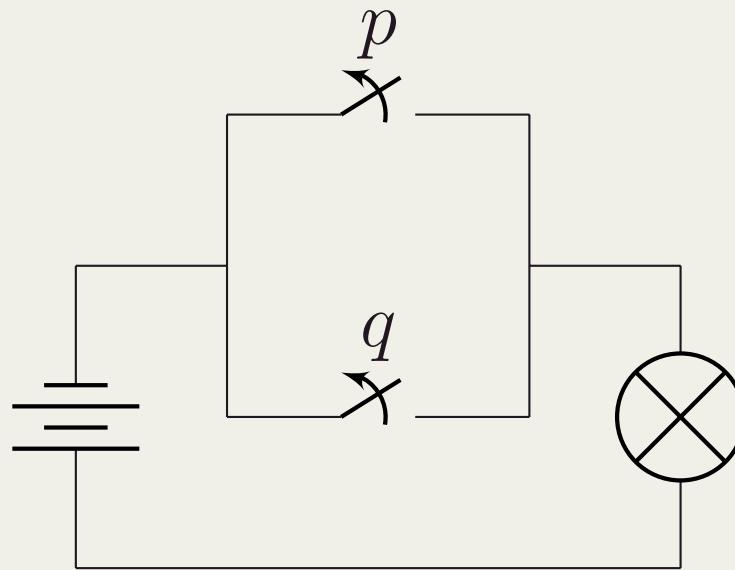
The AND gate is represented by the following input-output table and symbol:

p	q	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0



The OR gate

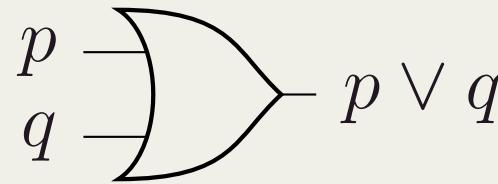
The circuit



gives an OR gate.

It is represented by the following table and symbol:

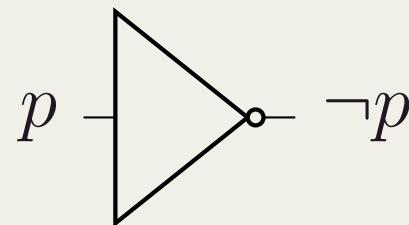
p	q	$p \vee q$
1	1	1
1	0	1
0	1	1
0	0	0



The NOT gate.

The NOT gate has the following table and symbol:

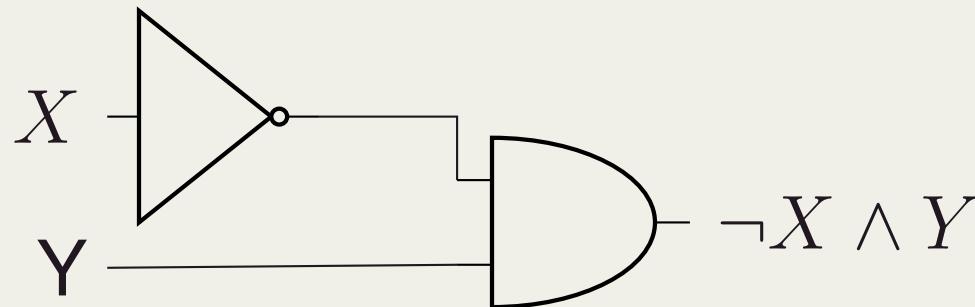
p	$\neg p$
1	0
0	1



Combining gates

Gates can be combined to create a circuit corresponding to a given compound statement.

Example:



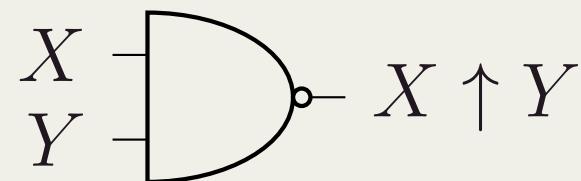
NAND and the NAND Gate

The logical connective **NAND** is a shorthand for “NOT AND”.

p NAND q is denoted by $p \uparrow q$ (sometimes $p|q$ is used instead of $p \uparrow q$). So $p \uparrow q \equiv \neg(p \wedge q)$.

A corresponding **NAND gate** is defined as follows:

X	Y	$X Y$
1	1	0
1	0	1
0	1	1
0	0	1



The functional completeness of NAND

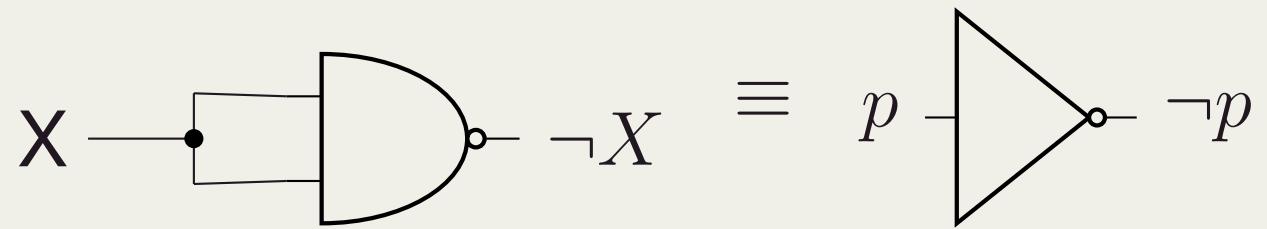
CLAIM: Every gate is equivalent to one that can be constructed by combining NAND gates alone.

How can we establish that this is true?

NOT from NAND

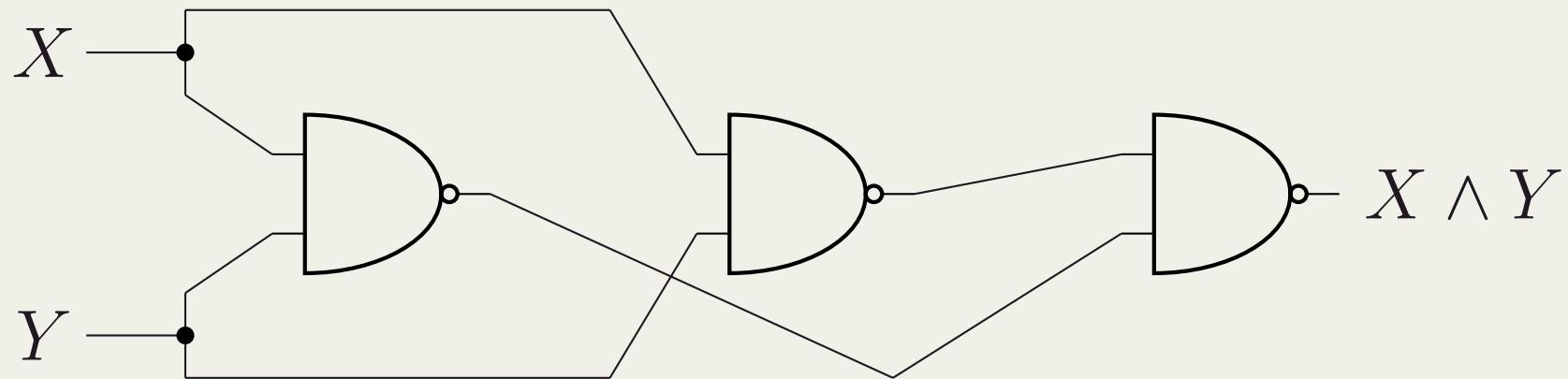
Example:

$$\neg X \equiv X \uparrow X$$



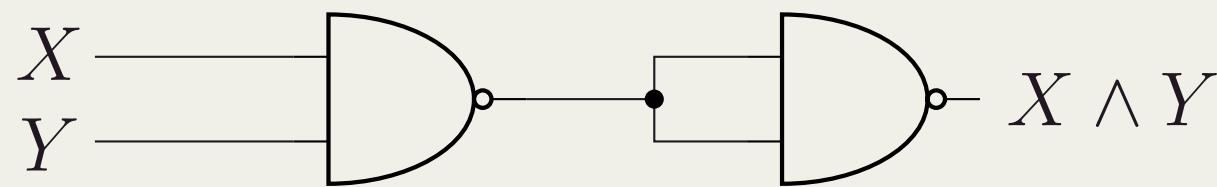
AND from NAND

$$X \wedge Y \equiv \neg(X \uparrow Y) \equiv (X \uparrow Y) \uparrow (X \uparrow Y).$$



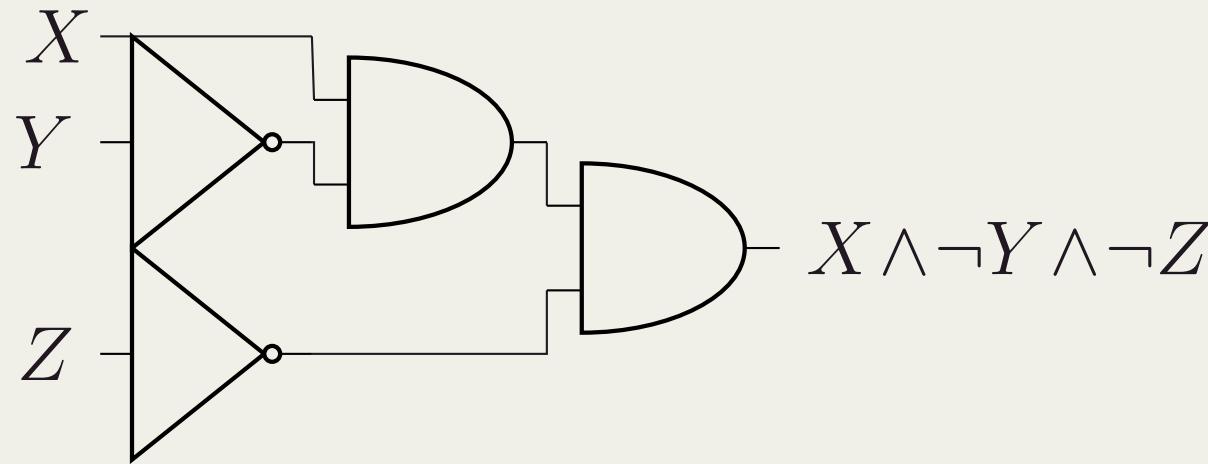
is equivalent to an AND gate.

The circuit can be simplified to use one less NAND gate:



Recogniser Circuits

Consider the circuit and corresponding table below:



X	Y	Z	$X \wedge \neg Y \wedge \neg Z$
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

We say that the circuit recognises the input combination $(X, Y, Z) = (1, 0, 0)$ because that's the only input combination that generates an output of 1.

Similarly a circuit for $\neg X \wedge Y \wedge Z$ would recognise the input combination $(X, Y, Z) = (0, 1, 1)$.

Recogniser circuits

Definition: A circuit that outputs 1 for only one input combination is called a recogniser for that input combination.

Q: Design a circuit to output 1 only for

$$(X, Y, Z) = (1, 1, 1) \& (1, 0, 0)$$

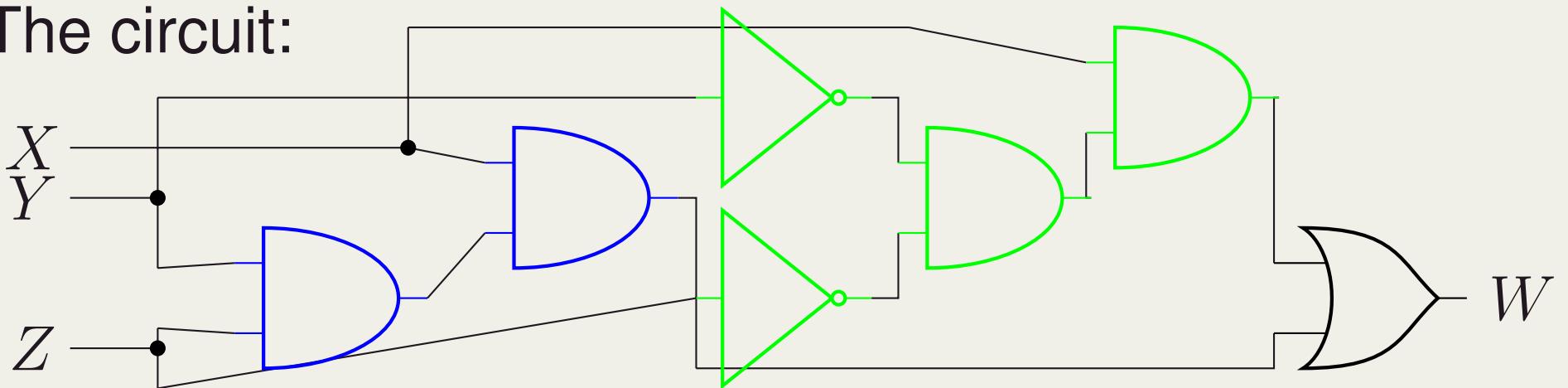
Method: Apply a method like the construction of a compound statement in disjunctive normal form. For outputs equal to 1 express inputs with AND. Join the resulting expressions with OR.

Example: From input-output tables to circuits

The table:

X	Y	Z	output	$X \wedge Y \wedge Z$	$X \wedge \neg Y \wedge \neg Z$	$W = (X \wedge Y \wedge Z) \vee (X \wedge \neg Y \wedge \neg Z)$
1	1	1	1	1	0	1
1	1	0	0	0	0	0
1	0	1	0	0	0	0
1	0	0	1	0	1	1
0	1	1	0	0	0	0
0	1	0	0	0	0	0
0	0	1	0	0	0	0
0	0	0	0	0	0	0

The circuit:



Thus ends Section A1

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section A: The language of mathematics and computer science

Part 2: Sets

References

The ideas about sets that we cover in the next four lectures are covered in the following sections in our optional text:

3ed: Chapter 5

4ed: Chapter 6

5ed: Sections 1.2-1.3 and Chapter 6

Lecture 4: Introduction to sets

Sets and elements

A **set** is a collection of **elements**.

This is an intuitive statement, but cannot be considered a definition because we do not give a precise definition of “collection” or “element” (other than saying they are the things that are in sets). Even so, this is where we start.

The notation $a \in S$ is read “ a is an element of S ”.

The notation $a \notin S$ is read “ a is not an element of S ”.

Axiom of extensionality

A set is determined by what its elements are. No importance is placed on the order in which elements are considered, or how many times an element appears in the set. Membership is the only things that matters.

Methods for describing a set

To effectively communicate which set you are talking about, you may:

- Describe S with language that communicates the precise nature of the set
- Use set-roster notation
- Use set-builder notation

Some important sets described with language

We write \emptyset for the **emptyset**. It is the set with no elements

Let \mathbb{Z}^+ denote the set of **positive integers** (sometimes called the natural numbers).

Let \mathbb{N} denote the set of **non-negative integers** (sometimes called the natural numbers).

Note that $0 \notin \mathbb{Z}^+$ but $0 \in \mathbb{N}$.

Let \mathbb{Z} denote the set of **integers**.

Let \mathbb{Q} denote the set of **rational numbers**.

Let \mathbb{R} denote the set of **real numbers**.

Example: Specifying membership with precision

Which, if any, of the following sentences define sets:

- A. Let E denote the set of Australian species that are endangered.
- B. Let E denote the set of Australian species that are *officially* endangered.
- C. Let E denote the set of species that are currently listed under Section 178 of the Environment Protection and Biodiversity Conservation Act 1999 (EPBC Act).

Example: Specifying membership with precision

Which, if any, of the following sentences define sets:

- A. Let P denote the set of all (computer) programs.
- B. Let P denote the set of all programs written in the language C .
- C. Let P denote the set of all *correct* programs written in the language C.
- E. Let P denote the set of programs written in the language C that terminate in finite time.
- F. Let P denote the set of programs written in the language C that accept no input from the user and will run to completion without a run-time error in finite time.

Set-roster notation

A set S may be specified using the set-roster notation by:

- writing all of its elements, with elements separated by commas, and the entire collection enclosed by braces;
- writing some elements and ellipses (ellipses are ... that read “and so on”), enclosing the entire description between braces. The elements written must establish enough of a pattern, in a way that is obvious to the reader, that it becomes clear how the pattern continues and whether or not the pattern ends at some point.

Examples

Describe each of the following sets in words.

$$S = \{\text{Mr Cookie, Lewis, Goblin}\}$$

$$S = \{\dots, -3, -1, 1, 3, 5, \dots\}$$

$$T = \{2, 3, 5, 7, \dots\}$$

$$U = \{3, 5, 7, \dots, 19\}$$

Examples

Describe each of the following sets in words.

$$S = \{\text{Mr Cookie, Lewis, Goblin}\}.$$

The names of guinea pigs that live in Adam's house.

$$S = \{\dots - 3, -1, 1, 3, 5, \dots\}.$$

The odd integers.

$$T = \{2, 3, 5, 7, \dots\}.$$

The prime numbers.

$$U = \{3, 5, 7, \dots, 19\}.$$

This could be “The odd primes less than 20” or “The odd integers between 2 and 20.” As soon as we identify two reasonable interpretations, we know this description needs improvement to be effective.

Axiom of extensionality again

Recall: A set is determined by what its elements are. No importance is placed on the order in which elements are considered, or how many times an element appears in the set.

So $\{a, b, c\} = \{c, b, a\} = \{a, a, a, a, b, b, c\}$

Set-builder notation

We use a predicate to describe membership. Recall that a predicate is a sentence that involves at least one variable, and the domain of the variable must be specified. Let $p(x)$ be a predicate defined over a domain D . Then we write

$$\{x \in D \mid p(x)\}$$

for “the set of all x in D for which $p(x)$ is true.”

Note: The use of \mid to separate the specification of the domain from the predicate is not universal. Other commonly used symbols include ‘ $:$ ’ and ‘ $,$ ’. In all cases, the symbol may be read as “such that” or “for which”.

Universe of discourse

In any mathematical context, there is usually a type of object you are thinking about. We will usually consider sets that are subsets of a set U , called the **universe of discourse**, or **universal set**. This often forms the domain of the predicates we use to describe sets in a given situation.

For example, in one context the universe of discourse may be the set of positive integers, while in another it may by the set of connected graphs (we will find out about these later).

The importance of having a universe of discourse will be explored later.

An Example

Example:

$$S = \{ \underbrace{x \in \mathbb{Z}}_{\text{domain}} \mid \underbrace{x^2 + 2x - 15 = 0}_{\text{predicate}} \}.$$

The expression may be read aloud as:

“ S is equal to the set of all x from the set of integers such that $x^2 + 2x - 15 = 0$.” or

“ S is equal to the set of all integers x such that $x^2 + 2x - 15 = 0$.”

“ S is equal to the set of all integers x for which $x^2 + 2x - 15 = 0$.”

Sets can be easy to define, but hard to understand

Example: Let $T = \{p \in \mathbb{Z}^+ \mid p \text{ and } p + 2 \text{ are both prime}\}$.

I can, for example, tell you that $3, 11, 17 \in T$ and $6, 7 \notin T$.

If you give me any positive integer n , and leave me alone long enough, I can tell you whether or not $n \in T$.

The statement:

The set T has infinitely many elements

is known as the Twin Prime Conjecture. It has been studied since 1849. At the time of writing, no one knows whether the statement is true or false.

Subsets

Let A and B be sets. We say that A is a **subset** of B , or A is contained in B , and we write $A \subseteq B$, when every element of A is also an element of B . Symbolically,

$$A \subseteq B \Leftrightarrow \forall x (x \in A \rightarrow x \in B).$$

You may say that \subseteq is the set theoretic analogue of IMPLIES in logic.

Understanding $A \not\subseteq B$

We often indicate that the negation of a statement is true by placing a diagonal slash through the key symbol in the statement. Now

Understanding $A \not\subseteq B$

We often indicate that the negation of a statement is true by placing a diagonal slash through the key symbol in the statement. Now

$$\begin{aligned} A \not\subseteq B &\Leftrightarrow \neg(A \subseteq B) \\ &\Leftrightarrow \neg\forall x (x \in A \rightarrow x \in B) \\ &\Leftrightarrow \exists x \neg(x \in A \rightarrow x \in B) \\ &\Leftrightarrow \exists x x \in A \wedge (\neg(x \in B)) \\ &\Leftrightarrow \exists x x \in A \wedge x \notin B. \end{aligned}$$

Understanding $A \not\subseteq B$

We often indicate that the negation of a statement is true by placing a diagonal slash through the key symbol in the statement. Now

$$\begin{aligned} A \not\subseteq B &\Leftrightarrow \neg(A \subseteq B) \\ &\Leftrightarrow \neg\forall x (x \in A \rightarrow x \in B) \\ &\Leftrightarrow \exists x \neg(x \in A \rightarrow x \in B) \\ &\Leftrightarrow \exists x x \in A \wedge (\neg(x \in B)) \\ &\Leftrightarrow \exists x x \in A \wedge x \notin B. \end{aligned}$$

We can interpret this intuitively: $A \not\subseteq B$ means that there is at least one element in A that is not in B .

Proper containment

We say that A is a **proper subset** of B , or A is properly contained in B , and write $A \subsetneq B$, when every element of A is in B but there is at least one element of B that is not in A . Symbolically:

$$A \subsetneq B \Leftrightarrow (A \subseteq B) \wedge (B \not\subseteq A)$$

You are well acquainted with the following proper containments:

$$\mathbb{Z}^+ \subsetneq \mathbb{N} \subsetneq \mathbb{Z} \subsetneq \mathbb{Q} \subsetneq \mathbb{R} \subsetneq \mathbb{C}.$$

BEWARE: Some mathematicians use \subset for \subsetneq . I don't do this because others mathematicians use \subset for \subseteq .

Set equality

Let A and B be sets. We say that A **equals** B , written $A = B$, when $A \subseteq B$ and $B \subseteq A$. Symbolically,

$$\begin{aligned} A = B &\Leftrightarrow (A \subseteq B) \wedge (B \subseteq A) \\ &\Leftrightarrow \forall x (x \in A \Leftrightarrow x \in B) \end{aligned}$$

You may say that $=$ is the set theoretic analogue of IF AND ONLY IF in logic.

An Example

A set may be described in more than one way. Some ways say more about the set, perhaps by saying why it is interesting, while others make it more obvious which things are elements of the set. For example:

$$\{x \in \mathbb{Z} \mid x^2 + 2x - 15 = 0\} = \{-5, 3\}$$

The first description tells me why the set is interesting in a given context, while the second description gives me a particularly clear understanding of the which integers are in the set and which integers are not in the set.

Set equality as a type of problem

Many problems in mathematics (and computer science) can be framed as: we know one way to describe a particular set because we know what makes the set interesting, but we want to know another way to describe the same set that makes membership easier to recognise or understand. Thus we wish to show a set equality.

For example: The problem of solving the equation $x^2 + 2x - 15 = 0$ over the domain of integers is essentially the following:

Show that

$$\{x \in \mathbb{Z} \mid x^2 + 2x - 15 = 0\} = \{-5, 3\}?$$

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section A: The language of mathematics and computer science

Part 2: Sets (continued)

Making new sets from old

Suppose that A and B are subsets of a universe U .

The **union** of A and B , denoted $A \cup B$, is the set

$$\{x \in U \mid (x \in A) \vee (x \in B)\}.$$

The **intersection** of A and B , denoted $A \cap B$, is the set

$$\{x \in U \mid (x \in A) \wedge (x \in B)\}.$$

The **difference** of B minus A , or B **without** A , denoted $B - A$ or $B \setminus A$, is the set

$$\{x \in U \mid (x \in B) \wedge (x \notin A)\}.$$

Making new sets from old

Suppose that A and B are subsets of a universe U .

The **complement** of A (in U), denoted A^c , is the set

$$\{x \in U \mid x \notin A\}$$

The complement of A cannot be understood unless the universe of discourse has been communicated.

The **symmetric difference** of A and B , denoted $A \Delta B$, is the set

$$\{x \in U \mid (x \in A) \oplus (x \in B)\}.$$

Some examples

Suppose that the universe of discourse is the set \mathbb{Z} and let

O be the set of odd integers

E be the set of even integers

P be the set of primes

C be the set of composite numbers.

A **composite number** is a positive integer that can be formed by multiplying two smaller positive integers.

Find simple expressions for: $O \cup E$, $O \cap E$, $E \cap P$,
 $O \cap P$, $P \cup C$, O^c , P^c , $E \Delta P$, $(O \Delta P) \cap \mathbb{Z}^+$

Some examples

$$O \cup E = \mathbb{Z}$$

$$O \cap E = \emptyset$$

$$E \cap P = \{2\}$$

$$O \cap P = P \setminus \{2\}$$

$$P \cup C = \{2, 3, 4, 5, \dots, \}$$

$$= \{x \in \mathbb{Z} \mid x \geq 2\}$$

$$O^c = E$$

$$P^c = \{\dots, -3, -2, -1, 0, 1\} \cup C$$

$$= \{z \in \mathbb{Z} \mid z \leq 1\} \cup C$$

$$E \Delta P = (E \cup P) \setminus \{2\}$$

$$(O \Delta P) \cap \mathbb{Z}^+ = (O \cap C) \cup \{1, 2\}.$$

Using logic to prove set identities

Since the set operations \cup , \cap , \setminus , \subseteq , c and Δ are defined using logical connectives, logical equivalences can be used to proved set theoretic identities (an identity is a relationship that holds not matter which substitutions are made for the variables).

An example

Let A , B and C be subsets of a universe U . Then

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

Proof.

$$\begin{aligned} & x \in A \cap (B \cup C) \\ \Leftrightarrow & (x \in A) \wedge (x \in B \cup C) && \text{Defn of } \cup \\ \Leftrightarrow & (x \in A) \wedge (x \in B \vee x \in C) && \text{Defn of } \cap \\ \Leftrightarrow & ((x \in A) \wedge (x \in B)) \vee ((x \in A) \wedge (x \in C)) && \text{Distr.} \\ \Leftrightarrow & (x \in A \cap B) \vee (x \in A \cap C) && \text{Defn of } \cap \\ \Leftrightarrow & x \in (A \cap B) \cup (A \cap C) && \text{Defn of } \cup \end{aligned}$$



Another way to construct a new set from an old set

For any set A , the power set of A , denoted $\mathcal{P}(A)$, is the set of all subsets of A .

For example, if $A = \{1, 2, 3\}$, then

$$\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

Q: If A has n elements, how many elements does $\mathcal{P}(A)$ have?

Another way to construct a new set from an old set

For any set A , the power set of A , denoted $\mathcal{P}(A)$, is the set of all subsets of A .

For example, if $A = \{1, 2, 3\}$, then

$$\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

Q: If A has n elements, how many elements does $\mathcal{P}(A)$ have?

A: $\mathcal{P}(A)$ has 2^n elements... for reasons we will explain when we discuss counting techniques later in the course.

Cartesian products: Another
way to make new sets from
old

Order and multiplicity

In sets, there is no sense of the order in which elements appear and there is no idea of how many times an element appears.

However, in many situations the order in which data appears is important, and the same data sometimes appears multiple times.

We now look at a construction that allows us to represent order and multiplicity.

Ordered n -tuples (see p.11 of our optional text).

Let n be a positive integer and let x_1, x_2, \dots, x_n be (not necessarily distinct) elements. The **ordered n -tuple** (x_1, x_2, \dots, x_n) consists of x_1, x_2, \dots, x_n together with the ordering: first x_1 , then x_2 , and so forth up to x_n . An ordered 2-tuple is called an **ordered pair**, and an ordered 3-tuple is called an **ordered triple**.

Two ordered n -tuples are **equal** when their elements match up exactly in order. Symbolically:

$$\begin{aligned}(x_1, x_2, \dots, x_n) &= (y_1, y_2, \dots, y_n) \\ \Leftrightarrow (x_1 = y_1) \wedge (x_2 = y_2) \wedge \cdots \wedge (x_n = y_n).\end{aligned}$$

An ordered m -tuple and an ordered n -tuple cannot be equal if $m \neq n$.

Examples

$(a, b, c) \neq (b, c, a)$ because their first elements differ.

$(a, a, b, c) \neq (a, b, c)$ because one is an ordered 4-tuple and the other is an ordered triple.

The elements in ordered n -tuples do not need to be of the same type. For example, $(\text{cat}, \text{car}, 1, \$)$ is an ordered 4-tuple.

We are, however, usually interested in sets of ordered n -tuples where all of the elements in, say, the i -th position are of the “same type”...

Cartesian product

Given (not necessarily distinct sets A_1, A_2, \dots, A_n , the **Cartesian product** of A_1, A_2, \dots, A_n , denoted $A_1 \times A_2 \times \dots \times A_n$, is the set of all ordered n -tuples (a_1, a_2, \dots, a_n) where $a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n$.

$$\{(a_1, a_2, \dots, a_n) \mid a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}.$$

This last expression may be read aloud as:

Cartesian product

Given (not necessarily distinct sets A_1, A_2, \dots, A_n , the **Cartesian product** of A_1, A_2, \dots, A_n , denoted $A_1 \times A_2 \times \dots \times A_n$, is the set of all ordered n -tuples (a_1, a_2, \dots, a_n) where $a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n$.

$$\{(a_1, a_2, \dots, a_n) \mid a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}.$$

This last expression may be read aloud as: “the set of all ordered n -tuples with elements a_1, a_2 , through, a_n such that a_1 comes from A_1 , a_2 comes from A_2 , through a_n comes from A_n .”

A word about the notation just used

The expression

$$\{(a_1, a_2, \dots, a_n) \mid a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}.$$

does not appear to conform to the rules of set-builder notation we laid out in the last lecture because

- the domain part introduces variables but does not specify a domain for each;
- the “predicate” does not appear to be a single predicate.

We can fix the second concern easily by making a rule that in a predicate, each comma is read and understood as “and”. It is usually better to use \wedge .

What I have written is an entirely standard way to describe a Cartesian product, even though it seems like a poor use of set-builder notation.

Examples

Let

$$A = \{\text{cat, dog, chicken}\}$$

$$B = \{\text{yes, no}\}$$

$$C = \{100, 300\}$$

Then

$$\begin{aligned} A \times B = & \{(\text{cat, yes}), (\text{cat, no}), (\text{dog, yes}), (\text{dog, no}), \\ & (\text{chicken, yes}), (\text{chicken, no})\}. \end{aligned}$$

and

$$C \times C = \{(100, 100), (100, 300), (300, 100), (300, 300)\}.$$

How to prove things
Putting our logic to work.

How to prove things I

- To prove a statement of the form $\forall x p(x)$, one may follow this plan:
Let x be a (fixed but arbitrary) element of the predicate domain. Argue that $p(x)$ is true.
- To disprove a statement of the form $\forall x p(x)$, one should prove the statement $\exists x \neg p(x)$. (This is called providing a **counterexample**)
- To prove a statement of the form $\exists x p(x)$, one may identify a particular element of the predicate domain and establish that $p(x)$ is true.
- To disprove a statement of the form $\exists x p(x)$, one should prove the statement $\forall x \neg p(x)$.

How to prove things II

To prove $\forall x p(x) \rightarrow q(x)$ you may:

- Let x be an arbitrary element of the domain.
- Suppose that $p(x)$ is true.
- Deduce by valid reasoning that $q(x)$ must be true (using the truth of $p(x)$ along the way).

This is called **arguing directly**.

You may also:

- Let x be an arbitrary element of the domain.
- Suppose that $\neg q(x)$ is true.
- Deduce by valid reasoning that $\neg p(x)$ must be true (using the truth of $\neg q(x)$ along the way).

This is called **arguing via the contrapositive**.

How to prove things III

Some advice:

1. Before starting a proof, clearly identify the logical structure of the statement to be proved.
2. Write down the logical structure of your argument so that the reader knows what is going on.
3. When deciding between a direct argument and an argument via the contrapositive, try whichever direction appears to allow you to make the strongest supposition first.

Example: Show $\{x \in \mathbb{Z} \mid x^2 + 2x - 15 = 0\} = \{-5, 3\}$

Let $A = \{x \in \mathbb{Z} \mid x^2 + 2x - 15 = 0\}$ and $B = \{-5, 3\}$. To show that $A = B$, we must show that $A \subseteq B$ and $B \subseteq A$.

Example: Show $\{x \in \mathbb{Z} \mid x^2 + 2x - 15 = 0\} = \{-5, 3\}$

Let $A = \{x \in \mathbb{Z} \mid x^2 + 2x - 15 = 0\}$ and $B = \{-5, 3\}$. To show that $A = B$, we must show that $A \subseteq B$ and $B \subseteq A$.

First we show that $B \subseteq A$. Let $x \in \mathbb{Z}$. To show that $x \in B \rightarrow x \in A$, we argue directly. Suppose that $x \in B$. Then $x = -5$ or $x = 3$. We use a proof by cases.

Example: Show $\{x \in \mathbb{Z} \mid x^2 + 2x - 15 = 0\} = \{-5, 3\}$

Let $A = \{x \in \mathbb{Z} \mid x^2 + 2x - 15 = 0\}$ and $B = \{-5, 3\}$. To show that $A = B$, we must show that $A \subseteq B$ and $B \subseteq A$.

First we show that $B \subseteq A$. Let $x \in \mathbb{Z}$. To show that $x \in B \rightarrow x \in A$, we argue directly. Suppose that $x \in B$. Then $x = -5$ or $x = 3$. We use a proof by cases.

Case $x = -5$: Then

$$x^2 + 2x - 15 = (-5)^2 + 2(-5) - 15 = 25 - 10 - 15 = 0.$$

Case $x = 3$: Then

$$x^2 + 2x - 15 = (3)^2 + 2(3) - 15 = 9 + 6 - 15 = 0.$$

In all cases, $x^2 + 2x - 15 = 0$ and hence $x \in A$.

Example continued

Now let $x \in \mathbb{Z}$. To show that $x \in A \rightarrow x \in B$, we argue via the contrapositive. Suppose that $x \notin B$. Then $x < -5$ or $-5 < x < 3$ or $x > 3$. We use a proof by cases.

Example continued

Now let $x \in \mathbb{Z}$. To show that $x \in A \rightarrow x \in B$, we argue via the contrapositive. Suppose that $x \notin B$. Then $x < -5$ or $-5 < x < 3$ or $x > 3$. We use a proof by cases.

Case $x < -5$: Then $x + 1 < -4$ and $(x + 1)^2 > 16$. Then

$$x^2 + 2x - 15 = x^2 + 2x + 1 - 16 = (x + 1)^2 - 16 > 0.$$

Case $-5 < x < 3$: Then $-4 < x + 1 < 4$ and $(x + 1)^2 < 16$. Then

$$x^2 + 2x - 15 = x^2 + 2x + 1 - 16 = (x + 1)^2 - 16 < 0.$$

Example continued

Case $x > 3$: Then $x + 1 > 4$ and $(x + 1)^2 > 16$. Then

$$x^2 + 2x - 15 = x^2 + 2x + 1 - 16 = (x + 1)^2 - 16 > 0.$$

In all cases, $x^2 + 2x - 15 \neq 0$ and hence $x \notin A$. \square

Example continued

Case $x > 3$: Then $x + 1 > 4$ and $(x + 1)^2 > 16$. Then

$$x^2 + 2x - 15 = x^2 + 2x + 1 - 16 = (x + 1)^2 - 16 > 0.$$

In all cases, $x^2 + 2x - 15 \neq 0$ and hence $x \notin A$. \square

The \square at the end used to be read as ‘quod erat demonstrandum’ or Q.E.D. for short; now it is more often read as “boom!”.

A counterexample disproves a \forall

Prove or disprove the following: The square root of an integer is always an integer.

This has the form of a universally quantified statement over the domain of all real numbers

$$\forall x (x \in \mathbb{Z} \rightarrow \sqrt{x} \in \mathbb{Z}).$$

We wish to show that the statement is false, so we wish to prove

$$\exists x (x \in \mathbb{Z} \wedge \sqrt{x} \notin \mathbb{Z}).$$

That is, we wish to provide a **counterexample** to the original statement.

Example: What I would write

Prove or disprove the following: The square root of an integer is always an integer.

The statement is false. The number 2 is an integer, but its square root is not. Therefore, 2 is a counterexample.



Example 6.1.2 on p.378 of our optional text.

Let

$$A = \{m \in \mathbb{Z} \mid \exists r \in \mathbb{Z} m = 6r + 12\}$$
$$B = \{n \in \mathbb{Z} \mid \exists s \in \mathbb{B} n = 3s\}$$

Prove that $A \subsetneq B$.

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section A: The language of mathematics and computer science

Part 2: Sets (continued)

Partitions: A structure for
recognising that a
classification works well

Motivation

A common task in any discipline (science, mathematics, philosophy, humanities, ...) is that of classifying things of a certain type into various sub-types. Thanks to our development of set theoretic tools, we have a way to formalise what it means for such a classification scheme to work really well.

Q: What properties do you think an excellent classification scheme will have?

An example

Which, if any, of the following classification schemes works well?

- We classify each integer as positive, negative or 0.
- We classify each song on the charts as pop, rock or urban.
- We classify each student enrolled in this course as a mathematician or a computer scientist or a physicist.

Disjoint sets

Sets A, B are called **disjoint** when $A \cap B = \emptyset$.

Given a set of sets \mathcal{S} , the sets in \mathcal{S} are said to be **pairwise disjoint** when

$$\forall A, B \in \mathcal{S} \quad A \neq B \rightarrow A \cap B = \emptyset.$$

An example

Let P be the set of prime numbers, let C the set of composite numbers, and let E be the set of even integers. Consider the sets

$$\mathcal{A} = \{\{1\}, P, C\} \text{ and } \mathcal{B} = \{\{1\}, P, E \cap \mathbb{N}\}$$

Since

$$\{1\} \cap P = \emptyset, \{1\} \cap C = \emptyset \text{ and } P \cap C = \emptyset,$$

the sets in \mathcal{A} are pairwise disjoint.

Since $(E \cap \mathbb{N}) \cap P = \{2\}$, the sets in \mathcal{B} are not pairwise disjoint.

A formal interpretation of an ‘excellent’ classification scheme

Let S be a set and $\mathcal{A} \subseteq \mathcal{P}(S)$ (so \mathcal{A} is a set, the elements of which are subsets of S). We say that \mathcal{A} is a **partition** of S when each of the following statements is true:

1. $\emptyset \notin \mathcal{A}$
2. every element of s is an element of some set in \mathcal{A} (that is, $\forall s \in S \exists A \in \mathcal{A} s \in A$)
3. the sets in \mathcal{A} are pairwise disjoint.

A formal interpretation of an ‘excellent’ classification scheme

Let S be a set and $\mathcal{A} \subseteq \mathcal{P}(S)$ (so \mathcal{A} is a set, the elements of which are subsets of S). We say that \mathcal{A} is a **partition** of S when each of the following statements is true:

1. $\emptyset \notin \mathcal{A}$
2. every element of s is an element of some set in \mathcal{A} (that is, $\forall s \in S \exists A \in \mathcal{A} s \in A$)
3. the sets in \mathcal{A} are pairwise disjoint.

Q: Do you agree or disagree that the three properties listed in the definition of a partition are a reasonable interpretation of what it means for a classification scheme (that classifies the elements of S) to be

Some examples

- $\mathcal{A} = \{\{1\}, P, C\}$ is a partition of \mathbb{Z}^+
- $\mathcal{B} = \{\{1\}, P, E \cap \mathbb{N}\}$ is not a partition of \mathbb{N} because the sets in \mathcal{B} are not pairwise disjoint.
- $\mathcal{A} = \{\{1\}, P, C\}$ is not a partition of \mathbb{N} because $0 \in \mathbb{N}$ but 0 is not in any set in \mathcal{A} .
- Let P, C, E, O be as above. Then $\{P \cap C, P \cap E, P \cap O\}$ is not a partition of P , because $P \cap C = \emptyset$.
- Let P, C, E, O be as above. Then $\{P \cap E, P \cap O\}$ is a partition of P .

Section A: The language of mathematics and computer science

Part 3: Relations and functions

Relations

Relations

Let A, B be sets. Any subset of $A \times B$ is called a **relation from A to B** . A relation from A to A is called a **relation on A** .

Given a relation R from A to B and an element $(a, b) \in A \times B$, we usually write $a R b$ instead of $(a, b) \in R$ and we usually write $a \not R b$ instead of $(a, b) \notin R$.

An example will help us see why relations are important, and why these choices of notation are made.

An example

Let

$$B = \{\text{One Direction}, \text{5SOS}, \text{Cody Simpson}, \text{BTS}\}$$

$L = \{\text{My daughter is a fan, I've seen them perform live,}$
 $\text{I would slay their tunes on Just Dance}\}$

$$\begin{aligned} \sim = & \{(\text{One Direction}, \text{My daughter is a fan}), \\ & (\text{5SOS}, \text{My daughter is a fan}), \\ & (\text{Cody Simpson}, \text{My daughter is a fan}), \\ & (\text{5SOS}, \text{I've seen them perform live}) \\ & (\text{Cody Simpson}, \text{I've seen them perform live}) \\ & (\text{BTS}, \text{I would slay their tunes on Just Dance})\} \end{aligned}$$

Then \sim is a relation from B to L .

Example: From arithmetic to inequalities

We define

$$< = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid b - a \in \mathbb{Z}^+\}.$$

We have defined a relation $<$ on \mathbb{Z} .

We could write $(5, 12) \in <$, but we usually prefer $5 < 12$.

Example: From arithmetic to inequalities

We define

$$< = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid b - a \in \mathbb{Z}^+\}.$$

We have defined a relation $<$ on \mathbb{Z} .

We could write $(5, 12) \in <$, but we usually prefer $5 < 12$.

Another way to write the same definition: For all integers a and b ,

$$((a, b) \in <) \Leftrightarrow (b - a \in \mathbb{Z}^+).$$

Example: From arithmetic to inequalities

We define

$$< = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid b - a \in \mathbb{Z}^+\}.$$

We have defined a relation $<$ on \mathbb{Z} .

We could write $(5, 12) \in <$, but we usually prefer $5 < 12$.

Another way to write the same definition: For all integers a and b ,

$$((a, b) \in <) \Leftrightarrow (b - a \in \mathbb{Z}^+).$$

Another way to write the same definition: For all integers a and b ,

$$(a < b) \Leftrightarrow (b - a \in \mathbb{Z}^+).$$

Another familiar example recast

Q: Define the familiar relation \geq on \mathbb{Z}

Another familiar example recast

Q: Define the familiar relation \geq on \mathbb{Z}

A: We define

$$\geq = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid a - b \in \mathbb{N}\}.$$

A: For all integers a and b ,

$$(a \geq b) \Leftrightarrow (a - b \in \mathbb{N}).$$

Inverse relation

The **inverse** R^{-1} of a relation $R \subseteq A \times B$ is the relation $R^{-1} \subseteq B \times A$ defined by

$$R^{-1} = \{(b, a) \in B \times A ; (a, b) \in R\}.$$

Inverse relation

The **inverse** R^{-1} of a relation $R \subseteq A \times B$ is the relation $R^{-1} \subseteq B \times A$ defined by

$$R^{-1} = \{(b, a) \in B \times A ; (a, b) \in R\}.$$

Thus $b R^{-1} a \iff a R b$.

Inverse relation

The **inverse** R^{-1} of a relation $R \subseteq A \times B$ is the relation $R^{-1} \subseteq B \times A$ defined by

$$R^{-1} = \{(b, a) \in B \times A ; (a, b) \in R\}.$$

Thus $b R^{-1} a \iff a R b$.

Example: Let A be a set of canned soup suppliers, let B be a set of supermarkets, and let $R \subseteq A \times B$ defined by $a R b \iff a \text{ sells to } b$. Then

Inverse relation

The **inverse** R^{-1} of a relation $R \subseteq A \times B$ is the relation $R^{-1} \subseteq B \times A$ defined by

$$R^{-1} = \{(b, a) \in B \times A ; (a, b) \in R\}.$$

Thus $b R^{-1} a \iff a R b$.

Example: Let A be a set of canned soup suppliers, let B be a set of supermarkets, and let $R \subseteq A \times B$ defined by $a R b \iff a \text{ sells to } b$. Then

$$\begin{aligned} b R^{-1} a &\iff a R b \\ &\iff a \text{ sells to } b \\ &\iff b \text{ buys from } a. \end{aligned}$$

Diagram of a relation

For small sets, relations can be expressed with **arrow diagrams**.

Example: Let

$$A = \{\text{Yummy!}, \text{Scrummy!}, \text{Yuck!}\}$$

$$B = \{\text{Big Shop}, \text{Food Place}, \text{Fresh Stuff}\}$$

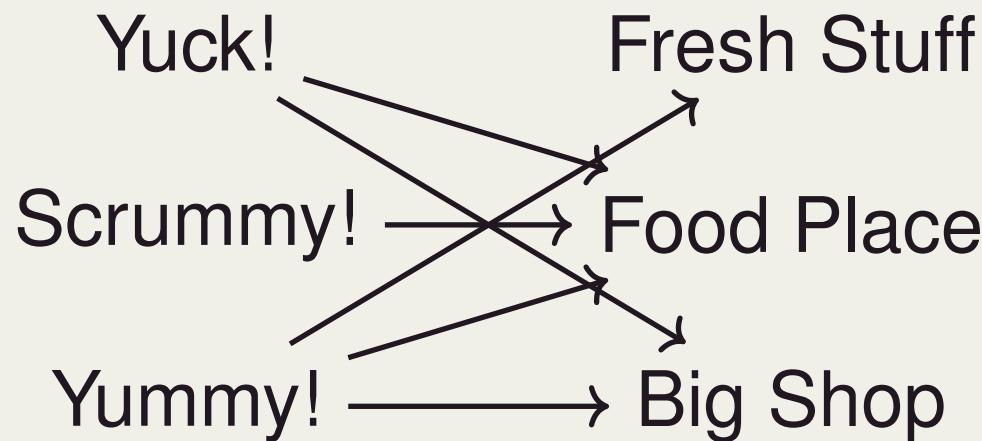
$$R = \{(\text{Yummy!}, \text{Big Shop}), (\text{Yummy!}, \text{Food Place}), (\text{Yummy!}, \text{Fresh Stuff}), (\text{Scrummy!}, \text{Food Place}), (\text{Yuck!}, \text{Food Place}), (\text{Yuck!}, \text{Big Shop})\}$$

Perhaps aRb means “a sells to b.”

Q: How would you represent R in a diagram?

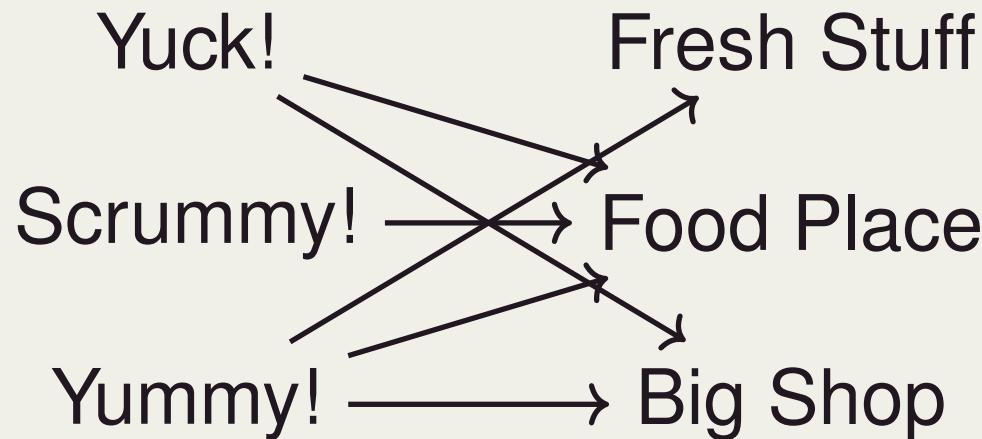
Example

Directed arrows from elements in set A to elements in set B can be used to show which elements are related.



Example

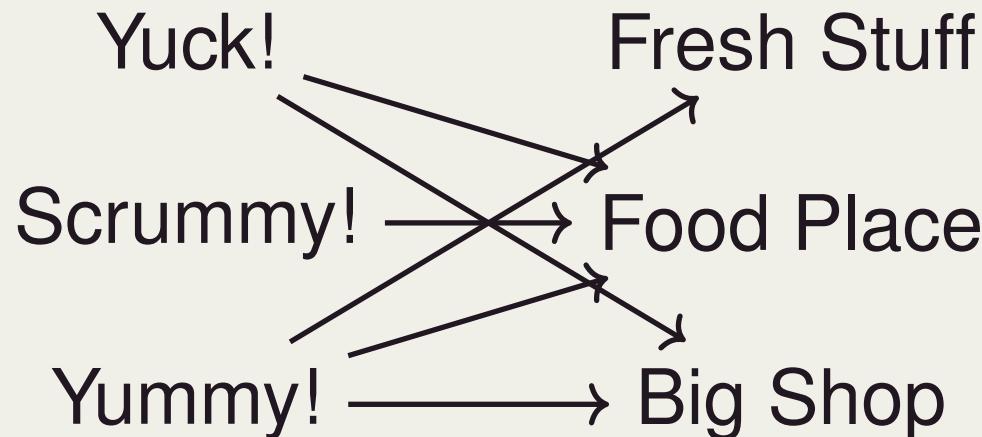
Directed arrows from elements in set A to elements in set B can be used to show which elements are related.



Q: How would you change the diagram above to represent the inverse relation R^{-1} ?

Example

Directed arrows from elements in set A to elements in set B can be used to show which elements are related.



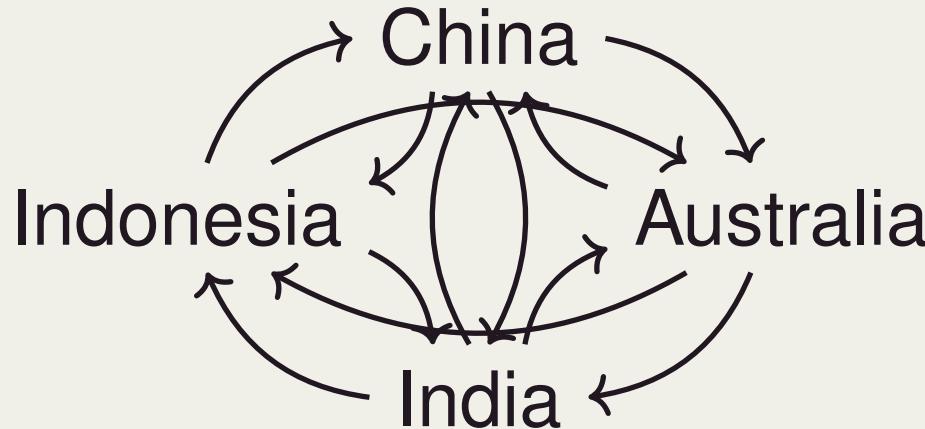
Q: How would you change the diagram above to represent the inverse relation R^{-1} ?

A: Change the direction of the arrows.

Directed Graphs

A **directed graph** or **digraph** is a set A of **vertices** together with a subset $R \subseteq A \times A$ of **directed edges**. If $(x, y) \in R$ we say ‘there is a directed edge from x to y ’. When A is small, a digraph can be drawn with the vertices as points and directed edges as arrows.

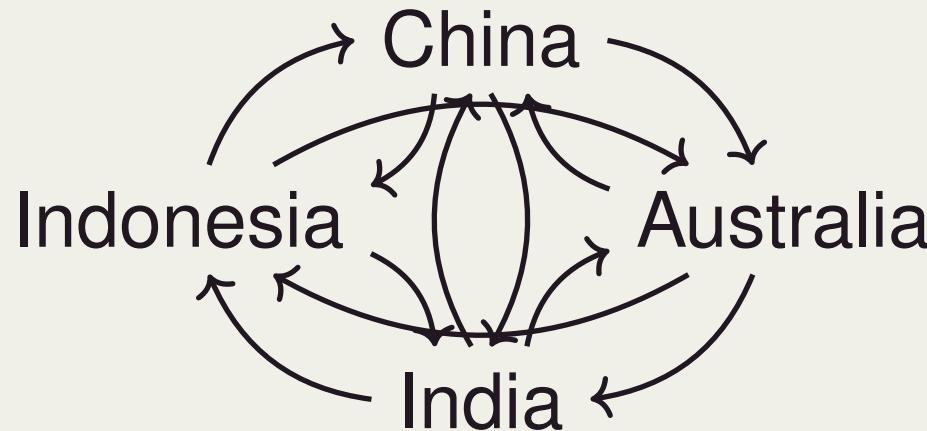
Example: A a set of countries. $a R b$ means a exports to b .



Directed Graphs

A **directed graph** or **digraph** is a set A of **vertices** together with a subset $R \subseteq A \times A$ of **directed edges**. If $(x, y) \in R$ we say ‘there is a directed edge from x to y ’. When A is small, a digraph can be drawn with the vertices as points and directed edges as arrows.

Example: A a set of countries. $a R b$ means a exports to b .



Each of the four countries exports to all the other three.

Example: Directed Graphs

Friends Ami, Bo, Chi and Di took photos of themselves visiting Parliament House.

- Ami took photos of Bo and Chi.
- Bo took photos of each of the others.
- Chi didn't take any photos.
- Di just took a selfie.

Example: Directed Graphs

Friends Ami, Bo, Chi and Di took photos of themselves visiting Parliament House.

- Ami took photos of Bo and Chi.
- Bo took photos of each of the others.
- Chi didn't take any photos.
- Di just took a selfie.

Draw a digraph for the relation P on $\{\text{Ami}, \text{Bo}, \text{Chi}, \text{Di}\}$ given by

$$x P y \iff x \text{ photographed } y.$$

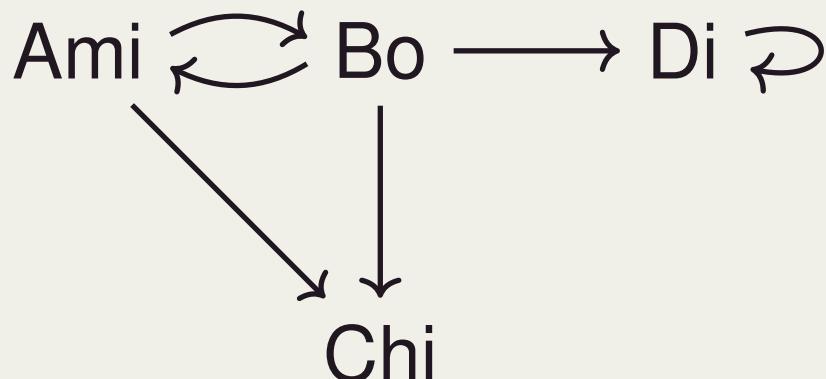
Example: Directed Graphs

Friends Ami, Bo, Chi and Di took photos of themselves visiting Parliament House.

- Ami took photos of Bo and Chi.
- Bo took photos of each of the others.
- Chi didn't take any photos.
- Di just took a selfie.

Draw a digraph for the relation P on $\{\text{Ami}, \text{Bo}, \text{Chi}, \text{Di}\}$ given by

$x P y \iff x \text{ photographed } y.$



(You can position the vertices how you like, of course.)

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section A: The language of mathematics and computer science

Part 3: Relations and functions (cont.)

Functions

Relations (recap)

Let A, B be sets. Any subset of $A \times B$ is called a **relation from A to B** . A relation from A to A is called a **relation on A** .

Given a relation R from A to B and an element $(a, b) \in A \times B$, we usually write $a R b$ instead of $(a, b) \in R$ and we usually write $a \not R b$ instead of $(a, b) \notin R$.

Functions

Let A, B be sets. A relation f from A to B is called a **function from A to B** when

$$\forall a \in A \exists !b \in B (a, b) \in f$$

The set A is called the **domain** of the function; the set B is called the **codomain** of the function. The **range** of f is the set

$$\{b \in B \mid \exists a \in A (a, b) \in f\}.$$

We write $f : A \rightarrow B$ to say that f is a function from A to B . Even though a function is a relation, we usually write $f(a) = b$ instead of $(a, b) \in f$ or $a \mathrel{fb}$.

Functions as machines that take inputs and produce outputs

You may like to think of a function as an abstract machine (this is an example of a ‘model’). It takes inputs. When given an input, the machine runs and produces an output according to some process. Using this model, we may interpret the function $f : A \rightarrow B$ as follows:

- the domain A is the set of allowed inputs to the function f
- the codomain B is the set from which outputs are selected by f
- the range is the subset of the codomain comprising the elements that are actually selected as outputs of the function.

About functions

We use the language of inputs and outputs to observe that:

- every input is assigned exactly one output—this is exactly the criteria under which a relation is a function;
- in general, the same output may be assigned to any number of input
- in general, some elements of the codomain may not be selected as the output for any input—that is, the range and the codomain may be different.

Example: Consider the function $s : \mathbb{Z} \rightarrow \mathbb{Z}$ that squares each input. Then $s(-3) = 9 = s(3)$, and -2 is in the codomain but not the range.

How to specify a function

To describe a function f you must:

- Describe the domain A
- Describe the codomain B
- Specify enough about “how the machine works” that for all $a \in A$, the predicate

$$p_a(b) : f(a) = b$$

is true for exactly one $b \in B$. It does not have to be easy to describe how to get $f(a)$ from a (in particular, the description may be very complicated), but the description must effectively select one output for each a .

Equal functions (a technical point)

Function f and g are **equal** if they have the same domain, the same codomain, and $f(x) = g(x)$ for all x in the domain.

Q: Are the following functions equal?

$f : \mathbb{Z} \rightarrow \mathbb{Z}$ defined by $f(z) = z^2$

$g : \mathbb{Z} \rightarrow \mathbb{N}$ defined by $f(z) = z^2$

Equal functions (a technical point)

Function f and g are **equal** if they have the same domain, the same codomain, and $f(x) = g(x)$ for all x in the domain.

Q: Are the following functions equal?

$f : \mathbb{Z} \rightarrow \mathbb{Z}$ defined by $f(z) = z^2$

$g : \mathbb{Z} \rightarrow \mathbb{N}$ defined by $f(z) = z^2$

A: No, because they have different codomains.

Another example

Q: Are the following functions equal?

$$f : \mathbb{Z} \rightarrow \mathbb{Z} \text{ defined by } f(z) = \frac{z^2 + 2z + 1}{(z + 1)}$$

$$g : \mathbb{Z} \rightarrow \mathbb{Z} \text{ defined by } f(z) = z + 1$$

Another example

Q: Are the following functions equal?

$$f : \mathbb{Z} \rightarrow \mathbb{Z} \text{ defined by } f(z) = \frac{z^2 + 2z + 1}{(z + 1)}$$

$$g : \mathbb{Z} \rightarrow \mathbb{Z} \text{ defined by } f(z) = z + 1$$

A: A TRICK QUESTION! The first function is not well-defined because the rule does not make sense when $z = -1$. *I refuse to answer your ill-posed problem!*

Q: Are the following functions equal?

$$f : \mathbb{N} \rightarrow \mathbb{N} \text{ defined by } f(z) = \frac{z^2 + 2z + 1}{(z + 1)}$$

$$g : \mathbb{N} \rightarrow \mathbb{N} \text{ defined by } f(z) = z + 1$$

Another example

Q: Are the following functions equal?

$$f : \mathbb{Z} \rightarrow \mathbb{Z} \text{ defined by } f(z) = \frac{z^2 + 2z + 1}{(z + 1)}$$

$$g : \mathbb{Z} \rightarrow \mathbb{Z} \text{ defined by } f(z) = z + 1$$

A: A TRICK QUESTION! The first function is not well-defined because the rule does not make sense when $z = -1$. *I refuse to answer your ill-posed problem!*

Q: Are the following functions equal?

$$f : \mathbb{N} \rightarrow \mathbb{N} \text{ defined by } f(z) = \frac{z^2 + 2z + 1}{(z + 1)}$$

$$g : \mathbb{N} \rightarrow \mathbb{N} \text{ defined by } f(z) = z + 1$$

A: Yes, because $\frac{z^2+2z+1}{(z+1)} = \frac{(z+1)^2}{(z+1)} = z + 1$.

Examples: Different notation to define a function

Here are several ways to define the same function:

- Let f be the function from \mathbb{Z} to \mathbb{Z} that squares its input.
- Let $f: \mathbb{Z} \rightarrow \mathbb{Z}$ be defined by the rule $f(z) = z^2$.
- Let $f = \{(a, b) \in \mathbb{Z} \times \mathbb{Z} \mid b = a^2\}$.
- Consider the function $\mathbb{Z} \rightarrow \mathbb{Z}$ given by $z \xrightarrow{f} z^2$
- Let $f: \mathbb{Z} \rightarrow \mathbb{Z}$ be $z \mapsto z^2$

In the above, the arrow \rightarrow is read ‘to’ and the arrow \mapsto is read ‘maps to’.

Another example: multiple inputs

Here is a function that takes an input from a Cartesian product. This is a sneaky way to take two inputs, yet still fit the definition of a function.

Let $A = \{\text{cat}, \text{dog}, \text{chicken}\}$ and $B = \{+, -\}$ and $C = \{1, 2, 3\}$ and let $R : A \times B \rightarrow C$ be defined by the rule:

$$R((\text{pet}, \text{test})) = \begin{cases} 1, & \text{if test is } + \\ 2, & \text{if test is } - \text{ and pet is not cat} \\ 3, & \text{if test is } - \text{ and pet is cat.} \end{cases}$$

In a mild abuse of notation, for such functions we tend to write $R(\text{cat}, +)$ instead of $R((\text{cat}, +))$.

Functions can be well-defined, but slow to compute

Let $f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ be defined by the rule

$$f(z) = \begin{cases} 1, & \text{if } z = 1 \\ \text{the smallest prime that divides } z, & \text{if } z \neq 1 \end{cases}$$

We do not know ‘fast’ ways to compute $f(z)$ when z is large.

This example illustrates that the ‘rule of the function’ (a description of the process by which the output is determined by the input) does not have to be easily expressible in a single simple formula. As long as each element of the domain is related to exactly one element of the codomain, a function has been defined.

Functions can be well-defined, but difficult to compute

Let $\cos : \mathbb{R} \rightarrow \mathbb{R}$ be the function such that θ maps to the x -coordinate of the point reached by starting at the point $(1, 0)$ in the Euclidean plane and travelling counter-clockwise θ units around the unit circle centred at $(0, 0)$.

Let $\sin : \mathbb{R} \rightarrow \mathbb{R}$ be the function such that θ maps to the y -coordinate of the point reached by starting at the point $(1, 0)$ in the Euclidean plane and travelling counter-clockwise θ units around the unit circle centred at $(0, 0)$.

A very difficult question (without a computer):
Evaluate $\cos(2.3)$ and $\sin(-1.45)$.

Properties of functions: injective

Let $f : A \rightarrow B$ be a function. We say that f is **one-to-one** or f is **injective** or f is an **injection** when

$$\forall a_1, a_2 \in A (a_1 \neq a_2) \rightarrow (f(a_1) \neq f(a_2))$$

So f is injective when different inputs must give different outputs.

Properties of functions: surjective

Let $f : A \rightarrow B$ be a function. We say that f is **onto** or f **maps onto** B or f is **surjective** or f is a **surjection** when

$$\forall b \in B \exists a \in A f(a) = b.$$

So f is surjective when its codomain and range are equal; f is surjective when every element of the codomain is an actual output of the function).

Properties of functions: surjective

Let $f : A \rightarrow B$ be a function. We say that f is **onto** or f **maps onto** B or f is **surjective** or f is a **surjection** when

$$\forall b \in B \exists a \in A f(a) = b.$$

So f is surjective when its codomain and range are equal; f is surjective when every element of the codomain is an actual output of the function).

To determine whether or not a function is surjective, the codomain must be explicitly understood.

Example: Consider the function f that takes any integer as input, and outputs the absolute value of the input. Then f is surjective if the codomain is \mathbb{N} , but not surjective if the codomain is \mathbb{Z} .

Properties of functions: bijective

Let $f : A \rightarrow B$ be a function. We say that f is **bijective** or f is a **bijection** when f is injective and surjective.

We say that A and B are in one-to-one correspondence when there exists a bijection $f : A \rightarrow B$.

Composition

Let A, B and C be subsets of a universe U . If $f : A \rightarrow B$ and $g : B \rightarrow C$ are functions, then the rule $a \mapsto g(f(a))$ defines a function called the **composition of g and f** and denoted $g \circ f$. Note

$$(g \circ f)(a) = g(f(a)).$$

The order of composition matters...

Order of composition matters

Let $A = \{\text{cat, dog, chicken}\}$ and let $f : A \rightarrow \mathbb{N}$ be defined by the rule

$$\begin{aligned} f \\ \text{cat} &\mapsto 70 \\ \text{dog} &\mapsto 90 \\ \text{chicken} &\mapsto 50, \end{aligned}$$

and let $g : \mathbb{N} \rightarrow \mathbb{N}$ be defined by the rule

$$g(z) = 3z.$$

Then $g \circ f$ is defined, but $f \circ g$ is undefined.

Inverse function

Let A, B be subsets of a universe U . Recall that if R is a relation from A to B , then the inverse relation R^{-1} from B to A is determined by the rule

$$aRb \Leftrightarrow bR^{-1}a.$$

Theorem: Let A, B be subsets of a universe U , and let $f : A \rightarrow B$ be a function. The inverse relation f^{-1} is a function from B to A (called the **inverse of** f) if and only if f is a bijection.

If f and f^{-1} are both functions, we call them **inverse functions** and we say that f is **invertible**.

Inverse functions and identity functions

For any set A , the identity function on A is the function $i_A : A \rightarrow A$ defined by the rule $a \mapsto a$.

If $f : A \rightarrow B$ is a bijection, then $f^{-1} \circ f = i_A$ and $f \circ f^{-1} = i_B$.

An example

Let's unpack something written using the vocabulary we have defined above.

Let $B = \{0, 1\}$ and $n \in \mathbb{Z}^+$. We define a set

$$B_n = \underbrace{B \times B \times \cdots \times B}_{n \text{ times}},$$

and a function $H_n : B_n \times B_n \rightarrow \mathbb{N}$ by the rule

$H_n(s, t) =$ the number of coordinate (bit) positions
where s and t differ.

Q: Explain, in your own words, what has been defined by the above.

Example continued

An infinite number of sets B_1, B_2, B_3, \dots and functions H_1, H_2, H_3, \dots (called the **Hamming functions**) have been defined. For each positive integer n , B_n is the set of n -tuples of binary digits (bits). So, for example, $(0, 1, 1, 0, 0, 1) \in B_6$.

The function H_n takes as input two n -tuples of bits (in the form of an ordered pair), compares them to see in which places they agree, and outputs the number of coordinates in which they disagree. For example,

$$H_5((0, 0, 0, 1, 1), (1, 0, 1, 0, 1)) = 3$$

because the n -tuples given disagree in the first, third and fourth coordinates.

Challenge

Describe, if possible, a way to arrange the elements of B_n around a circle so that $H_n(s, t) = 1$ whenever s and t are adjacent in your arrangement.

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section A: The language of mathematics and computer science

Bringing it all together

Let's start with some practice

Proving set theoretic identities ... by unpacking and repacking the logic

Prove that, for all subsets A, B of a universal set U ,

$$(A \cap B)^c = A^c \cup B^c.$$

Proving that a function is injective/surjective/bijective

Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be the function defined by

$$f(x) = 5x + 13.$$

Prove that f is bijective.

Proving/disproving

Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be the function defined by

$$f(x) = 10 - 2x^2.$$

Determine whether or not f is bijective.

Capturing an idea of ‘same size’

Let U be a universal set, and let A, B be subsets of U . We say that $A \sim B$ when there exists a bijection $f : A \rightarrow B$.

EXAMPLE: $\{\text{cat, dog, chicken}\} \sim \{50, 60, 25\}$.

EXAMPLE: $\{\text{cat, dog, chicken}\} \not\sim \{50, 60, 25, 110\}$.

Some awesome results (due to Cantor (1845-1918))

$$\mathbb{Z}^+ \sim \mathbb{N}$$

$$\mathbb{Z}^+ \sim \mathbb{Z}$$

$$\mathbb{Z}^+ \sim \mathbb{Q}$$

$$\mathbb{Z}^+ \not\sim \mathbb{R}$$

Some awesome results (due to Cantor (1845-1918))

$$\mathbb{Z}^+ \sim \mathbb{N}$$

$$\mathbb{Z}^+ \sim \mathbb{Z}$$

$$\mathbb{Z}^+ \sim \mathbb{Q}$$

$$\mathbb{Z}^+ \not\sim \mathbb{R}$$

Some awesome results (due to Cantor (1845-1918))

$$\mathbb{Z}^+ \sim \mathbb{N}$$

$$\mathbb{Z}^+ \sim \mathbb{Z}$$

$$\mathbb{Z}^+ \sim \mathbb{Q}$$

$$\mathbb{Z}^+ \not\sim \mathbb{R}$$

Some awesome results (due to Cantor (1845-1918))

$$\mathbb{Z}^+ \sim \mathbb{N}$$

$$\mathbb{Z}^+ \sim \mathbb{Z}$$

$$\mathbb{Z}^+ \sim \mathbb{Q}$$

$$\mathbb{Z}^+ \not\sim \mathbb{R}$$

Some important definitions

Definition: A set A is **countable** when there exists a bijection from A to a subset of \mathbb{Z}^+ .

Definition: A set A is **countably infinite** when there exists a bijection from A to \mathbb{Z}^+ .

Definition: A set A is **uncountable** when it is not countable.

What is this course about again? (see lecture 1)

Discrete mathematical models are abstract representations of processes and objects, the steps or units of which can be indexed by the non-negative integers. In particular, we avoid continua (like the open interval $(0, 1)$).

What is this course about again? (see lecture 1)

Discrete mathematical models are abstract representations of processes and objects, the steps or units of which can be indexed by the non-negative integers. In particular, we avoid continua (like the open interval $(0, 1)$).

Can be restated

Discrete mathematical models are abstract representations of processes and objects, the steps or units of which form a countable set.

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section B: Digital Information

Representing numbers

Positional notation in base b

Let b , called the **base**, be an integer such that $b \geq 2$, and let d_0, d_1, \dots, d_{b-1} be symbols, called **digits**, that represent the first b non-negative integers in order.

Any non-negative integer number can be written in exactly one way using these digits and **positional notation** as follows: For any

$$x_{i_s}, x_{i_{s-1}}, \dots, x_{i_0} \in \{d_0, d_1, \dots, d_{b-1}\},$$

the expression

$$(x_{i_s} x_{i_{s-1}} \dots x_{i_0})_b$$

means

$$x_{i_s} \times b^s + x_{i_{s-1}} \times b^{s-1} + \dots + x_{i_0} \times b^0$$

Decimal

$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is the set of **decimal digits**.
These digits are used for representing numbers in base 10.

$$(63403)_{10}$$

$$= (6 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 0 \times 10^1 + 3 \times 10^0)_{10}$$

Binary

$\{0, 1\}$ is the set of **binary digits**, usually called **bits** for short. These digits are used for representing numbers in base 2.

$$(10111)_2$$

$$= (1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10}$$

$$= (16 + 0 + 4 + 2 + 1)_{10}$$

$$= (23)_{10}$$

Octal

$\{0, 1, 2, 3, 4, 5, 6, 7\}$ is the set of **octal digits**. These digits are used for representing numbers in base 8.

$(63403)_8$

$$= (6 \times 8^4 + 3 \times 8^3 + 4 \times 8^2 + 0 \times 8^1 + 3 \times 8^0)_{10}$$

$$= (24576 + 1536 + 256 + 0 + 3)_{10}$$

$$= (26371)_{10}$$

Hexadecimal

$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$ is the set of **hexadecimal digits**. These digits are used for representing numbers in base 16.

$$(3AD0F)_{16}$$

$$= (3 \times 16^4 + 10 \times 16^3 + 13 \times 16^2 + 0 \times 16^1 + 15 \times 16^0)_{10}$$

$$= (196608 + 40960 + 3328 + 0 + 15)_{10}$$

$$= (240911)_{10}$$

Notational conventions

We often omit the parentheses, or omit both the parentheses and the subscript indicating the base when the omission is unlikely to cause misunderstanding.

Example: We often write 11010100_2 instead of $(11010100)_2$. In the middle of a page of computations in binary, we might write simply write 11010100.

In the slides above you also saw that we sometimes group sums and products inside a single set of parentheses to indicate the base in which all the numbers are represented.

Converting from decimal to binary

Suppose that $n \in \mathbb{N}$ is represented in decimal. To write n in binary: write n as a sum of powers of 2. Start by finding the largest power of 2 that does not exceed n .

$$\begin{aligned}(71)_{10} &= (64 + 7)_{10} \\&= (64 + 4 + 3)_{10} \\&= (64 + 4 + 2 + 1)_{10} \\&= (2^6 + 2^2 + 2^1 + 2^0)_{10} \\&= (1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 \\&\quad + 1 \times 2^1 + 1 \times 2^0)_{10} \\&= (1000111)_2\end{aligned}$$

An example

$$\begin{aligned}(347)_{10} &= (256 + 91)_{10} \\&= (256 + 64)_{10} \\&= (256 + 64 + 27)_{10} \\&= (256 + 64 + 16 + 11)_{10} \\&= (256 + 64 + 16 + 8 + 3)_{10} \\&= (256 + 64 + 16 + 8 + 2 + 1)_{10} \\&= (2^8 + 2^6 + 2^4 + 2^3 + 2^1 + 2^0)_{10} \\&= (1 \times 2^8 + 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 \\&\quad + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10} \\&= (101011011)_2\end{aligned}$$

Converting between binary to octal

Suppose that $n \in \mathbb{N}$ is represented in binary. To write n in octal: collect the bits into groups of 3, then replace each group of 3 bits by the appropriate octal digit.

$$\begin{aligned}(1011110)_2 &= (\underbrace{1}_{\text{001}} \underbrace{011}_{\text{011}} \underbrace{110}_{\text{110}})_2 \\ &= (136)_8\end{aligned}$$

Reverse the process the go from octal to binary.

Converting from binary to hexadecimal

Suppose that $n \in \mathbb{N}$ is represented in binary. To write n in hexadecimal: collect the bits into groups of 4, then replace each group of 4 bits by the appropriate hexadecimal digit.

$$\begin{aligned}(1011110)_2 &= (\underbrace{101}_{} \underbrace{1110}_{})_2 \\&= (\underbrace{0101}_{} \underbrace{1110}_{})_2 \\&= (5E)_{16}\end{aligned}$$

Reverse the process the go from hexadecimal to binary.

Why binary?

Many real work phenomena exist in one of two states: current flowing or no current flowing; a north pole or south pole of a magnet; at each location on a laser disc, we either burned a tiny hole or we did not burn a tiny hole. By interpreting one of the states as 0 and the other as 1, we can represent information. We simply need to agree upon how to encode information in the form of 0's and 1's. It is therefore natural to represent numbers in binary, rather than encode decimal digits as strings of 0's and 1's and then try to make numerical meaning from them.

Why octal and hexadecimal?

A page of binary is difficult for humans to read.

The same information is written in a more compact form on a page, and is more easily read by humans, in octal or hexadecimal. Conversions between binary, octal and hexadecimal are convenient.

Some vocabulary

A digit in base 2 is called a **bit**. This is simply a contraction of **binary digit**. It was first used in information theory by Tukey in 1948.

A block of 8 bits is called a **byte**. It was first used by IBM around the 60's when they started using an 8-bit character code known as EBCDIC.

A block of 4 bits is called a **nibble**.

A sequence of several adjacent bytes is called a **word**. The number of bytes varies, depending on the purpose of the word. For example, a 2-byte word can store non-negative integers in the range from 0 to $2^{16} - 1 = 65535$.

Addition and subtraction in decimal

In base 10:

$$123 + 678 : \begin{array}{r} 1 & 2 & 3 \\ + & 6 & 7 & 8 \\ \hline 8 & 0 & 1 \end{array}$$

$$123 - 78 : \begin{array}{r} 1 & 2 & 3 \\ - & 7 & 8 \\ \hline 4 & 5 \end{array}$$

Addition and subtraction in binary

In base 2:

$$111_2 + 10_2 : \begin{array}{r} 1\ 1\ 1 \\ + \quad 1\ 0 \\ \hline 1\ 0\ 0\ 1 \end{array}$$

$$1010_2 - 111_2 : \begin{array}{r} 1\ 0\ 1\ 0 \\ - \quad 1\ 1\ 1 \\ \hline 0\ 0\ 1\ 1 \end{array}$$

Addition and subtraction in hexadecimal

In base 16:

$$456_{16} + ABC_{16} : \begin{array}{r} 4 \quad 5 \quad 6 \\ + \quad A \quad B \quad C \\ \hline F \quad 1 \quad 2 \end{array}$$

$$F12_{16} - ABC_{16} : \begin{array}{r} F \quad 1 \quad 2 \\ - \quad A \quad B \quad C \\ \hline 4 \quad 5 \quad 6 \end{array}$$

Representing positives, negatives and zero

When writing positive and negative integers in binary, we may use a - or + in front of the integer to represent its sign.

We may write $+(1101)_2$ for $(13)_{10}$
and $-(1101)_2$ for $-(13)_{10}$.

How can we, in a computer, represent integers in a range that includes negative integers?

Representing positives, negatives and zero in a computer

How can we, in a computer, represent integers in to a range that includes negative integers?

FIRST IDEA: Use an extra **sign bit**. Fix a number of bits to use for each integer, and the left-most bit is 0 if the number is positive, and 1 is the number is negative.

In such a scheme with say, a byte to represent an integer:

$(00001101)_2$ would represent $(13)_{10}$

$(10001101)_2$ would represent $(-13)_{10}$

and $(00000000)_2$ and $(10000000)_2$ would both represent zero.

This is **NOT** how integers are usually represented.

Representing positives, negatives and zero in a computer

How can we, in a computer, represent integers in to a range that includes negative integers?

BETTER IDEA: Use an extra **sign bit**, but the binary representation of a negative is chosen according to a scheme called two's complement...

Representing positives, negatives and zero in a computer

We fix a number of bits, say t , to use for each integer. A string of t bits $d_1d_2 \dots d_t$ is interpreted as a number in one of two ways, depending on the value of d_1 :

- if $d_1 = 0$, then the bit string $(0d_2d_3 \dots d_t)$ represents $(d_2d_3 \dots d_t)_2$
- if $d_1 = 1$, then the bit string $(1d_2d_3 \dots d_t)$ represents $(d_2d_3 \dots d_t)_2 - 2^{t-1}$, which is equivalent to $-[(e_2e_3 \dots e_t)_2 + 001_2]$ where

$$e_i = \begin{cases} 1, & \text{if } d_i = 0 \\ 0, & \text{if } d_i = 1. \end{cases}$$

The process of evaluation $(1d_2d_3 \dots d_t)_2$ is often described as: ‘toggle’ all bits, add one, then negate.

Example: 4-bit signed integers

When we use nibbles to represent integers, then $t = 4$ and we have:

nibble	decimal value	nibble	decimal value
0000	0	1000	-8
0001	1	1001	-7
0010	2	1010	-6
0011	3	1011	-5
0100	4	1100	-4
0101	5	1101	-3
0110	6	1110	-2
0111	7	1111	-1

In this case, we say we are using **4-bit signed integers**.

Example: 8-bit signed integers

When we use bytes to represent integers, then $t = 8$ and we have:

nibble	decimal value	nibble	decimal value
00000000	0	10000000	-128
00000001	1	10000001	-127
00000010	2	10000010	-126
:	:	:	:
01111110	126	11111110	-2
01111111	127	11111111	-1

In this case, we say we are using **8-bit signed integers**.

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section B: Digital Information

Representing numbers (cont.)

Addition and subtraction in decimal

In base 10:

$$123 + 678 : \begin{array}{r} 1 & 2 & 3 \\ + & 6 & 7 & 8 \\ \hline 8 & 0 & 1 \end{array}$$

$$123 - 78 : \begin{array}{r} 1 & 2 & 3 \\ - & 7 & 8 \\ \hline 4 & 5 \end{array}$$

Addition and subtraction in binary

In base 2:

$$111_2 + 10_2 : \begin{array}{r} 1\ 1\ 1 \\ + \quad 1\ 0 \\ \hline 1\ 0\ 0\ 1 \end{array}$$

$$1010_2 - 111_2 : \begin{array}{r} 1\ 0\ 1\ 0 \\ - \quad 1\ 1\ 1 \\ \hline 0\ 0\ 1\ 1 \end{array}$$

Addition and subtraction in hexadecimal

In base 16:

$$456_{16} + ABC_{16} : \begin{array}{r} 4 \quad 5 \quad 6 \\ + \quad A \quad B \quad C \\ \hline F \quad 1 \quad 2 \end{array}$$

$$F12_{16} - ABC_{16} : \begin{array}{r} F \quad 1 \quad 2 \\ - \quad A \quad B \quad C \\ \hline 4 \quad 5 \quad 6 \end{array}$$

Representing positives, negatives and zero

When writing positive and negative integers in binary, we may use a - or + in front of the integer to represent its sign.

We may write $+(1101)_2$ for $(13)_{10}$
and $-(1101)_2$ for $-(13)_{10}$.

How can we, in a computer, represent integers in a range that includes negative integers?

Representing positives, negatives and zero in a computer

How can we, in a computer, represent integers in to a range that includes negative integers?

FIRST IDEA: Use an extra **sign bit**. Fix a number of bits to use for each integer, and the left-most bit is 0 if the number is positive, and 1 is the number is negative.

In such a scheme with say, a byte to represent an integer:

$(00001101)_2$ would represent $(13)_{10}$

$(10001101)_2$ would represent $(-13)_{10}$

and $(00000000)_2$ and $(10000000)_2$ would both represent zero.

This is **NOT** how integers are usually represented.

Representing positives, negatives and zero in a computer

How can we, in a computer, represent integers in to a range that includes negative integers?

BETTER IDEA: Use an extra **sign bit**, but the binary representation of a negative is chosen according to a scheme called two's complement...

Representing positives, negatives and zero in a computer

We fix a number of bits, say t , to use for each integer. A string of t bits $d_1d_2 \dots d_t$ is interpreted as a number in one of two ways, depending on the value of d_1 :

- if $d_1 = 0$, then the bit string $(0d_2d_3 \dots d_t)$ represents $(d_2d_3 \dots d_t)_2$
- if $d_1 = 1$, then the bit string $(1d_2d_3 \dots d_t)$ represents $(d_2d_3 \dots d_t)_2 - 2^{t-1}$, which is equivalent to $-[(e_2e_3 \dots e_t)_2 + 001_2]$ where

$$e_i = \begin{cases} 1, & \text{if } d_i = 0 \\ 0, & \text{if } d_i = 1. \end{cases}$$

The process of evaluation $(1d_2d_3 \dots d_t)_2$ is often described as: ‘toggle’ all bits, add one, then negate.

Example: 4-bit signed integers

When we use nibbles to represent integers, then $t = 4$ and we have:

nibble	decimal value	nibble	decimal value
0000	0	1000	-8
0001	1	1001	-7
0010	2	1010	-6
0011	3	1011	-5
0100	4	1100	-4
0101	5	1101	-3
0110	6	1110	-2
0111	7	1111	-1

In this case, we say we are using **4-bit signed integers**.

Example: 8-bit signed integers

When we use bytes to represent integers, then $t = 8$ and we have:

nibble	decimal value	nibble	decimal value
00000000	0	10000000	-128
00000001	1	10000001	-127
00000010	2	10000010	-126
:	:	:	:
01111110	126	11111110	-2
01111111	127	11111111	-1

In this case, we say we are using **8-bit signed integers**.

NOTE: To go from the bit string representing an integer y to the bit string representing $-y$, simply toggle all bits, add one (and throw away any carry digit from the most significant place after the addition).

Adding 1-bit numbers p and q

p	q	$p + q$	
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

Adding 1-bit numbers p and q

p	q	$p + q$	
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

p	q	$p + q$	
		LB	RB
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

Adding 1-bit numbers p and q

p	q	$p + q$
1	1	1 0
1	0	0 1
0	1	0 1
0	0	0 0

p	q	$p + q$	
		LB	RB
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

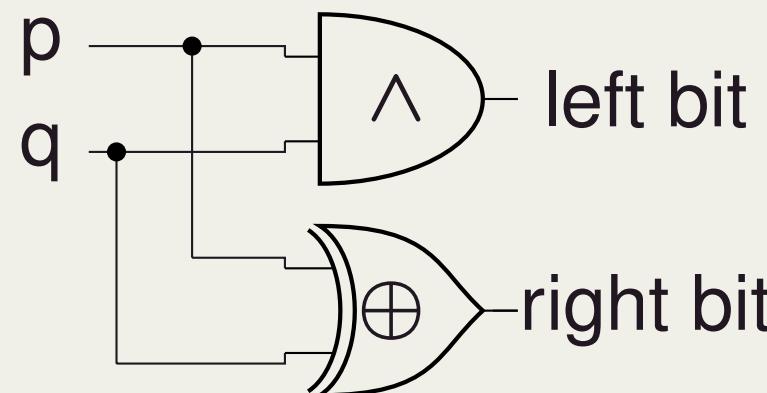
p	q	$p + q$	
		$p \wedge q$	$p \oplus q$
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

Adding 1-bit numbers p and q

p	q	$p + q$
1	1	1 0
1	0	0 1
0	1	0 1
0	0	0 0

p	q	$p + q$	
		LB	RB
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0

p	q	$p + q$	
		$p \wedge q$	$p \oplus q$
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0



This circuit is called a **half adder** (2 bits in, 2 bits out)

Adding 1-bit numbers p , q and r

p	q	r	$p + q + r$	
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

Adding 1-bit numbers p , q and r

p	q	r	$p + q + r$
1	1	1	1 1
1	1	0	1 0
1	0	1	1 0
1	0	0	0 1
0	1	1	1 0
0	1	0	0 1
0	0	1	0 1
0	0	0	0 0

p	q	r	$p + q + r$	
				LB
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

CHALLENGE: Construct a circuit diagram for a circuit which implements the above (3 bits in, 2 bits out). We shall call such a circuit a **full adder**.

Meaningful renaming of bits in a full adder

p	q	carry in	carry out	sum out
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

Example: 4-bit addition via a cascade of full adders

$$\begin{array}{r} A = 0110 \\ + B = 0011 \end{array}$$

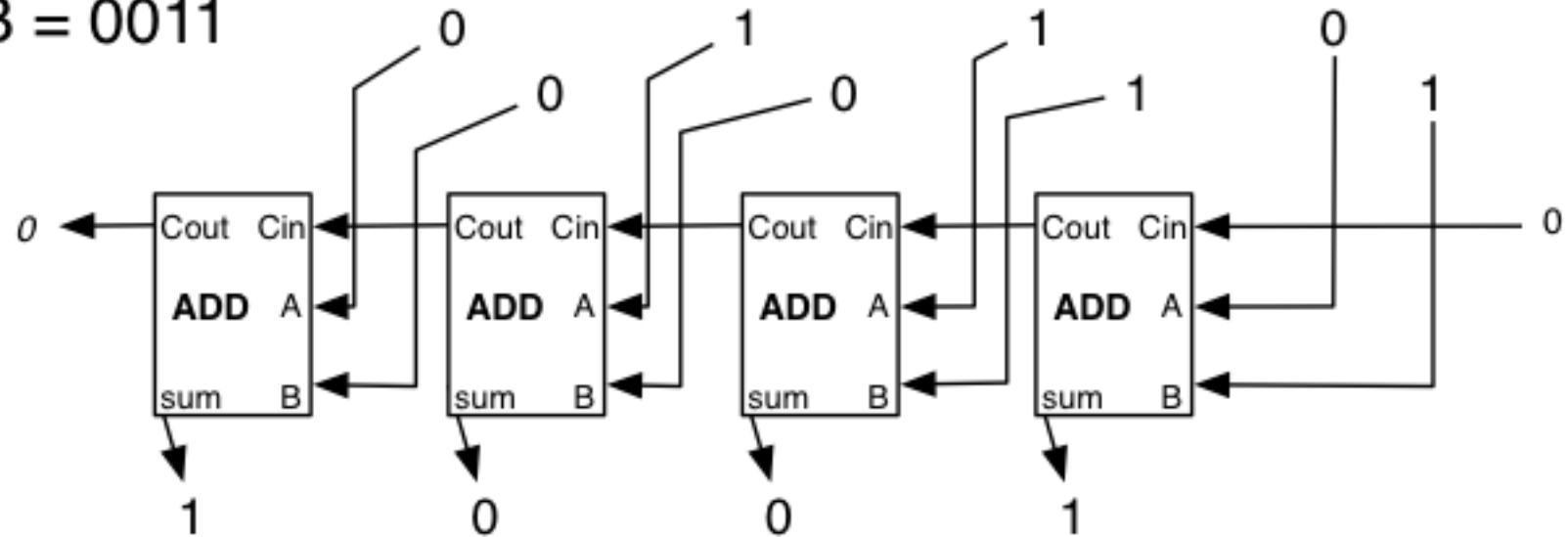


Diagram extracted from “Concepts in Computing” course notes at Dartmouth College, Hanover, New Hampshire, USA, 2009.

<http://www.cs.dartmouth.edu/~cbk/classes/4/notes/19.php>

Circuits for integer subtraction

Since $x - y$ is the same as $x + (-y)$, we can accomplish integer subtraction provided we can convert y to $-y$ (because we already know how to add).

Our clever way of representing negative numbers means

- the integer representation of an integer y can be converted to the integer representation of $-y$ by toggling all bits and adding one.
- A NOT gate will toggle a single bit, and we already know how to build a circuit that adds, so we can build a circuit that adds one.

Something amazingly neat

Our clever way of representing negative numbers means

- the usual algorithm (the step-by-step process) that works for adding two non-negative integers, works for adding two negative integers and for adding a non-negative integer and a negative integer
- the carry bit from the most significant place in an addition can be ignored.
- this will work provided that the sum is in the range of integers that can be represented by the number of bits we have chosen.

Examples

4-bit examples:

$$\begin{array}{r}
 4 & & 0 & 1 & 0 & 0 \\
 -6 & & + & 1 & 0 & 1 & 0 \\
 \hline
 -2 & & & 1 & 1 & 1 & 0
 \end{array}$$

$$\begin{array}{r}
 7 \\
 -3 \\
 \hline
 4
 \end{array}
 \qquad
 \begin{array}{r}
 0 & 1 & 1 & 1 \\
 + & 1 & 1 & 0 & 1 \\
 \hline
 (1) & 0 & 1 & 0 & 0
 \end{array}$$

The carry bit in () is ignored.

8-bit example:

$$\begin{array}{r}
 81 \\
 -98 \\
 \hline
 -17
 \end{array}
 \qquad
 \begin{array}{r}
 01010001 \\
 +10011110 \\
 \hline
 11101111
 \end{array}$$

Multiplication

Example:

$$\begin{array}{r} & 1 & 0 & 1 & 1 \\ \times & 1 & 1 & 0 & 0 \\ \hline & 0 & 0 & 0 & 0 \\ + & 0 & 0 & 0 & 0 \\ + & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline + & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & 1 & 1 & 1 & 1 \end{array}$$

Multiplication

Example:

$$\begin{array}{r} & 1 & 0 & 1 & 1 \\ \times & 1 & 1 & 0 & 0 \\ \hline & 0 & 0 & 0 & 0 \\ + & 0 & 0 & 0 & 0 \\ + & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline + & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & 1 & 1 & 1 & 1 \end{array}$$

Notice that no ‘multiplication tables’ are required since the only multiplications used are ‘times 0’ which results in all zeroes, and ‘times 1’ which has no effect.

Multiplication

Example:

$$\begin{array}{r} & 1 & 0 & 1 & 1 \\ \times & 1 & 1 & 0 & 0 \\ \hline & 0 & 0 & 0 & 0 \\ + & 0 & 0 & 0 & 0 \\ + & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline + & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & 1 & 1 & 1 & 1 \end{array}$$

Notice that no ‘multiplication tables’ are required since the only multiplications used are ‘times 0’ which results in all zeroes, and ‘times 1’ which has no effect.

This is one of the bonuses of using binary in computers.

Multiplication is accomplished by shifts and additions

For computer implementation of multiplication (within \mathbb{N}) we only need a ‘shift’ circuit (to move bits left and append a zero on the right) and addition circuits.

But we also need an *algorithm* to control the process.

An **algorithm** is a sequence of operations on an input.
The result is called an output.

Examples: Addition, subtraction, multiplication, division.

Example

Here is very simple list processing algorithm, set out using a standard format for displaying algorithms:

Input: L : a list of students.

Output: O : list of students who have submitted their assignment.

Method:

Set $j = 1$, $O =$ empty list. (Initialization phase)

Loop: If $j = \text{length}(L) + 1$, stop.

If $L[j]$ (the j -th element in L) has submitted the assignment, append $L[j]$ to O .

Replace j by $j + 1$.

Repeat loop.

Algorithm for binary addition

This algorithm formalizes the method described earlier, which is implemented with a cascade of full adders.

Contrary to notation used in some earlier examples, in this context it is usual to *number the bits starting from the rightmost bit, which is called the 0-th bit*.

So the leftmost bit of an n -bit number is its $(n - 1)$ -th bit.

Algorithm for binary addition

Input: Two numbers p, q in base 2 with n bits.

Output: The sum $s = p + q$ in base 2 with $n + 1$ bits.

Method:

Set $j = 0$, $c = 0$, and the n -th bits of p and q to 0.

Loop: If $j = n + 1$ stop.

Add the j -th bits of p and q plus c .

Store the result part as the j -th bit of s .

Store the carry part as c (replacing the old value of

Replace j by $j + 1$.

Repeat loop

Algorithm for binary multiplication

This algorithm formalises the method described earlier, except that addition is done progressively rather than at the end.

The additions may be done by the algorithm of the previous slide.

This algorithm also assumes the existence of a shift algorithm, which shifts all the bits of a number one place to the left and then fills the vacant rightmost place with a 0.

Algorithm for binary multiplication

Input: Two numbers x, y in base 2 with n bits.

Output: The product $p = x \times y$ in base 2 with $2n$ bits.

Method:

Set $j = 0$ and $p = 0$.

Loop: If $j = n$ stop.

If the j -th bit of y is 1 then replace p by $p + x$.

Shift x .

Replace j by $j + 1$.

Repeat loop.

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

Section B: Digital Information

Representing numbers (cont.)

What is a rational number?

Recall that Q is the set of **rational numbers**. A rational number is a number that can be represented as the ratio of two integers.

EXAMPLE $\frac{2}{3}$ is a rational number.

Please note that every integer is a rational number as, for example $6 = \frac{6}{1}$.

What is a rational number?

Recall that Q is the set of **rational numbers**. A rational number is a number that can be represented as the ratio of two integers.

EXAMPLE $\frac{2}{3}$ is a rational number.

Please note that every integer is a rational number as, for example $6 = \frac{6}{1}$.

Are you happy with this definition?

What is a rational number ? (again)

Let $Q = \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$.

What does an element of Q look like?

What is a rational number ? (again)

Let $Q = \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$.

What does an element of Q look like?

The set Q may be partitioned so that any elements (n_1, d_1) and (n_2, d_2) of Q are in the same partition set if and only if $n_1 d_2 = n_2 d_1$.

So, for example,

$\{(2, 3), (-2, -3), (4, 6), (-4, -6), (6, 9), (-6, -9), \dots\}$ is one of the sets in the partition

The sets in the partition may themselves be considered rational numbers. We usually write $\frac{2}{3}$ instead of $\{(2, 3), (-2, -3), (4, 6), (-4, -6), (6, 9), (-6, -9), \dots\}$.

Representing a rational number in a computer

For computer storage of any **non-zero** rational number q we need to express it using **scientific notation** with base 2. For any base b this is

where

$$q = (-1)^s \times m \times b^n$$

- $q \in \mathbb{Q}$, $q \neq 0$;
- $b \in \mathbb{Z}^+$, $b \geq 2$;
- $s \in \{0, 1\}$, (s is the **sign bit**)
- $m \in \mathbb{Q}$, $1 \leq m < b$, (m is the '**mantissa**') and
- $n \in \mathbb{Z}$ (n is the **exponent**).

Given q and b , the values of s , m and n are uniquely determined by these conditions.

Example in base 10: $13.5 = (-1)^0 \times 1.35 \times 10^1$.

A technicality and more examples

Example in base 10: $13.5 = (-1)^0 \times 1.35 \times 10^1$.

To save space, we store 135 instead of 1.35 because 1.35 is the only number between 1 and 10 with digits 1, 3, 5.

A technicality and more examples

Example in base 10: $13.5 = (-1)^0 \times 1.35 \times 10^1$.

To save space, we store 135 instead of 1.35 because 1.35 is the only number between 1 and 10 with digits 1, 3, 5.

Examples in base 10:

- $-154 = (-1)^1 \times 1.54 \times 10^2$. Store m as 154.
- $0.031 = (-1)^0 \times 3.1 \times 10^{-2}$. Store m as 31.

The number of digits used to store m is called the **precision**.

An example in binary

Example in base 2, precision 4, 4-bit exponent n

(As we have seen, using a 4-bit signed integer means that $-8 \leq n \leq 7$):

$\underbrace{1}_{s} \quad \underbrace{0101}_{m} \quad \underbrace{0011}_{n}$

or, alternatively,

$\underbrace{1}_{s} \quad \underbrace{1010}_{m} \quad \underbrace{0011}_{n}$

$$m = 1.01_2 = 1(2^0) + 0(2^{-1}) + 1(2^{-2}) = 1 + \frac{1}{4} = \frac{5}{4}.$$

$$n = 3_{10}.$$

$$\text{So } q = -\frac{5}{4} \times 2^3 = -10_{10}.$$

NOTE: In practice, when using binary, the “1.” part of the mantissa m is not stored, since it is implied. So in 4-bit precision 1.01_2 would be stored as 0100 (with *no* alternative).

A warning

WARNING: With limits on precision and exponent size, some rational numbers can only be stored inaccurately, if at all.

Of course, the same sort of thing is true for integers. But with integers we can represent ALL of the integers close enough to 0, so it is easier to understand which integers we can and cannot represent.

If you have a reason to represent rational numbers accuracy beyond the accuracy provided by some sort of standard set up, you can write dedicated software to represent numbers with greater precision.

Modular arithmetic

A Theorem

Theorem

$$\forall n \in \mathbb{Z} \ \forall d \in \mathbb{Z}^+ \ \exists!q \in \mathbb{Z} \ \exists!r \in \mathbb{N} (n = qd + r) \wedge (0 \leq r < d)$$

RECALL: In the lecture slides we use the notation $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ and $\mathbb{Z}^+ = \{1, 2, 3, \dots\}$.

A Theorem

Theorem

$$\forall n \in \mathbb{Z} \ \forall d \in \mathbb{Z}^+ \ \exists! q \in \mathbb{Z} \ \exists! r \in \mathbb{N} (n = qd + r) \wedge (0 \leq r < d)$$

RECALL: In the lecture slides we use the notation $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ and $\mathbb{Z}^+ = \{1, 2, 3, \dots\}$.

We can say the same thing in words.

Theorem: Given any integer n and given any positive integer d , there is exactly one way to express n as an integer multiple qd of d plus a non-negative ‘remainder’ r less than the ‘divisor’ d .

This theorem is called the **Quotient-Remainder Theorem**.

A way to understand q and r

Fix a choice of $n \in \mathbb{Z}$ and $d \in \mathbb{Z}^+$.

Now picture a number line with the integers marked in some dramatic way.

Now: qd is the integer multiple of d that is closest to n but NOT to the right of n ; and r is the distance between qd and n .

A picture will help...

The ‘mod’ and ‘div’ operations

We define: $q = n \text{ div } d$; $r = n \text{ mod } d$.

You may like to say that:

- $n \text{ div } d$ gives the **quotient** when n is divided by d ;
- $n \text{ mod } d$ gives the **remainder** when n is divided by d .

MATH1005/MATH6005:
Discrete Mathematical
Models

Adam Piggott

Semester 1, 2021

An announcement/correction

Earlier in the course, we adopted the notation

$$\mathbb{Z}^+ = \{1, 2, 3, \dots\} \text{ and } \mathbb{N} = \{0, 1, 2, 3, \dots\}$$

In previous versions of the course, the notation

$$\mathbb{N} = \{1, 2, 3, \dots\} \text{ and } \mathbb{N}^* = \{0, 1, 2, 3, \dots\}$$

has been used.

It seems that I grossly underestimated the amount of work involved in changing all of the optional problems, worksheets, and assignments (5 variations per week) to account for my preferred notation...

Our new convention

From now on, in all conversations we have (lectures, worksheets, assignments, etc) we will adopt the notation

$$\mathbb{N} = \{1, 2, 3, \dots\} \text{ and } \mathbb{N}^* = \{0, 1, 2, 3, \dots\}.$$

Modular arithmetic

Another way to say the Q-R Theorem

Theorem: Given any integer n and given any positive integer d , there exist unique integers q and r such that

$$n = dq + r \text{ and } 0 \leq r < d.$$

We say that q is the quotient, and r the remainder, when n is divided by d .

We define: $q = n \text{ div } d$; $r = n \text{ mod } d$.

Please note that the remainder is ALWAYS non-negative and less than d .

Examples

Q: Evaluate the following expressions:

$$87 \bmod 13$$

$$-100 \text{ div } 13$$

A:

Since $87 = 6(13) + 9$, $87 \bmod 13 = 9$.

Since $-100 = (-8)(13) + 4$, $-100 \text{ div } 13 = -8$.

WARNING: Please pay careful attention to the second example. Using the Q-R Theorem when n is negative is often a source of confusion. Note that, even when n is negative, r is non-negative.

The division algorithm

The ‘primary school’ method of finding quotient and remainder is to use *repeated subtraction*. This only works for non-negative n .

Input: $n \in \mathbb{N}^*$ and $d \in \mathbb{N}$.

Output: $q = n \text{ div } d$ and $r = n \text{ mod } d$.

Method:

Set $r = n$, $q = 0$.

Loop: If $r < d$ stop.

Replace r by $r - d$.

Replace q by $q + 1$.

Repeat loop

Some small modifications to the algorithm allow it cope also with negative n .

Congruence modulo d

Let $d \in \mathbb{N}$. The **congruence modulo d** relation $R_d \subseteq \mathbb{Z} \times \mathbb{Z}$ is defined by

$$aR_d b \Leftrightarrow \exists k \in \mathbb{Z} ; a = b + kd.$$

We have unusual notation for this relation. We write $a \equiv b \pmod{d}$ to mean $aR_d b$

Two ways to understand the relation

Lemma 1: Two integers are congruent modulo d if and only if they leave the same remainder upon division by d . That is

$$\forall a, b \in \mathbb{Z} \forall d \in \mathbb{N} [a \equiv b \pmod{d}] \Leftrightarrow [a \bmod d = b \bmod d]$$

Lemma 2: Two integers are congruent modulo d if and only if their difference is a multiple of d

$$\forall a, b \in \mathbb{Z} \forall d \in \mathbb{N} [a \equiv b \pmod{d}] \Leftrightarrow [\exists k \in \mathbb{Z} b - a = kd]$$

Examples

Example: $-17 \equiv 15 \pmod{8}$ since

$$-17 - 15 = -32 = (-4)8$$

Example: $-17 \equiv 15 \pmod{8}$ since

$$-17 = (-3) \times 8 + 7$$

and

$$15 = 1 \times 8 + 7,$$

so -17 and 15 leave the same remainder upon division by 8.

\equiv partitions the integers

For any $d \in \mathbb{N}$ and any $a \in \mathbb{Z}$ the **congruence class** $[a]_d$ (or ‘equivalence class’) of a modulo d is defined by

$$[a]_d = \{m \in \mathbb{Z} \mid m \equiv a \pmod{d}\}.$$

EXAMPLE: The elements of $[2]_7$, for example, are all the integers that leave remainder 2 upon division by 7.

Lemma: R_d induces the partition $\{[0]_d, [1]_d, \dots, [d-1]_d\}$ on \mathbb{Z}

Modular arithmetic: Something amazing

Lemma: Let $d \in \mathbb{N}$ and let $a_1, b_1, a_2, b_2 \in \mathbb{Z}$. If

$$a_1 \equiv a_2 \pmod{d} \text{ and } b_1 \equiv b_2 \pmod{d},$$

then the following are all true:

1. $a_1 + b_1 \equiv a_2 + b_2 \pmod{d}$
2. $a_1 - b_1 \equiv a_2 - b_2 \pmod{d}$
3. $a_1 \times b_1 \equiv a_2 \times b_2 \pmod{d}$

We will prove the first assertion together. The second and the third are left as an exercise for the reader.

Proof of the first assertion

Suppose that $a_1 \equiv a_2 \pmod{d}$ and $b_1 \equiv b_2 \pmod{d}$. By definition, there exist integers j, k such that

$$a_1 = a_2 + dj \text{ and } b_1 = b_2 + dk.$$

Now

$$\begin{aligned}(a_1 + b_1) - (a_2 + b_2) &= (a_2 + dj + b_2 + dk) - (a_2 + b_2) \\&= dj + dk \\&= d(j + k).\end{aligned}$$

It follows by Lemma 2 that

$$(a_1 + b_1) \equiv (a_2 + b_2) \pmod{d}. \quad \square$$

The upshot

When working with arithmetic expressions modulo d , we may at any moment replace a number by another number that is equivalent to it, modulo d . This allows a number of fast calculations in modular arithmetic.

EXAMPLE:

$$\begin{aligned} 24 + 71 \times 13 &\equiv 2 + 5 \times 2 \pmod{11} \\ &\equiv 12 \pmod{11} \\ &\equiv 1 \pmod{11}. \end{aligned}$$

WARNING: Division is not as easy when working in modular arithmetic. Which is great!

Applications

Modular arithmetic is often used in cryptography, to generate pseudo-random numbers (necessary for running simulations), for error-detecting codes (used in credit card numbers, for example) and many other applications.

An example

A 10-digit string $d_1d_2\dots d_{10}$ (where the last digit may be X , which is considered to have the value 10) is a **valid ISBN-10** if

$$1 \times d_1 + 2 \times d_2 + \dots + 10 \times d_{10} \equiv 0 \pmod{11}$$

For example, the ISBN-10 for our optional text is 0357114086 and

$$\begin{aligned} 1(0) + 2(3) + 3(5) + 4(7) + 5(1) + 6(1) + 7(4) + 8(0) + \\ 9(8) + 10(6) = 220 \equiv 0 \pmod{11}. \end{aligned}$$

Why is it worth making sure book identifiers have this property?

A flavour of number theory: Two statements about primes and modular arithmetic

The **greatest common divisor** of $a, b \in \mathbb{N}$, written **gcd**(a, b), is the largest $n \in \mathbb{N}$ such that $a \bmod n = 0$ and $b \bmod n = 0$.

If $\text{gcd}(a, b) = 1$, a, b are called **relatively prime**.

Examples: $\text{gcd}(30, 75) = 15$; $\text{gcd}(30, 77) = 1$; so 30, 77 are relatively prime.

Fermat's little theorem, (proof omitted):

If p is prime and $a \in \mathbb{N}$ satisfies $\text{gcd}(a, p) = 1$ then $a^{p-1} \bmod p = 1$.

Example: $\text{gcd}(6, 11) = 1$, so $6^{10} \bmod 11 = 1$.

Check: $6^{10} = 60\,466\,176 = 5\,496\,925 \times 11 + 1$.

B2: Sequences

Text Reference (Epp)

- | | |
|---------------|--|
| 3ed: Sections | 4.1-4, 8.1-3 (Sequences and induction),
9.3,5 (Sorting) |
| 4ed: Sections | 5.1-4,6-8, (Sequences and induction),
11.3,5 (Sorting) |
| 5ed: Sections | 5.1-4,6-7, (Sequences and induction),
11.3,5 (Sorting) |

Sequences

Let S be a set and $I \subseteq \mathbb{N}^*$. A function $a : I \rightarrow S$ is called a **sequence in S** . Special **sequence notation** is often used:

Function notation	Sequence notation
$a : I \rightarrow S$ $n \mapsto a(n).$	$(a_n)_{n \in I} \subseteq S$

The notation $(a_n)_{n \in I}$ indicates that the function can be represented as an *ordered -tuple* or, more simply, as a *list*.

(Unlike a *set*, a *list* has an order, and can have repeated entries.)

Examples

- $I = \{1, 2, 3\}$: $(a_n)_{n \in I} = (a_1, a_2, a_3)$.
- $I = \mathbb{N}^*$: $(a_n)_{n \in I} = (a_0, a_1, a_2, \dots \dots)$.

In practice we usually leave out the parentheses and speak of “the sequence a_1, a_2, a_3 ” or “the sequence $a_0, a_1, a_2, \dots \dots$ ”

Examples of Sequences

The “ $\subseteq S$ ” part of the sequence notation $(a_n)_{n \in I} \subseteq S$ indicates that the sequence members belong to S ; i.e. that the range of the sequence function $a : I \rightarrow S$ is a subset of its codomain S .

The sequence *itself* is **not** a subset of S , since it is not a set.

Examples

1. $(a_n)_{n \in \mathbb{N}} \subseteq \mathbb{Q}$ sequence of interests rates. a_n is an interest rate at time n .
2. $(p_n)_{n \in \mathbb{N}} \subseteq \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{N}^*$ sequence of states of an ecosystem. $p_n = (r_n, s_n, t_n)$: population of species r, s, t at time n .
3. $(a_n)_{n \in \mathbb{N}} \subseteq \mathbb{N}^*$ sequence of amplitudes. a_n : amplitude of the harmonic of frequency $n \times f$ (f fundamental frequency).
4. U set of users. $(u_n)_{n \in \{1,2,3,4,5\}} \subseteq U$: a list of 5 users.

Describing sequences: explicit definitions

An **explicit definition** of a sequence is a formula for a_n .

Examples:

1. $\forall n \in \mathbb{N} \quad a_n = 2^n. \quad (a_n)_{n \in \mathbb{N}} = 2, 4, 8, 16, \dots,$
2. $a_1 = \text{Pierre}, a_2 = \text{Julie}, a_3 = \text{Paul}.$
 $(a_n)_{n \in \{1,2,3\}} = \text{Pierre, Julie, Paul.}$

Describing sequences: Implicit definitions

An **implicit definition** of a sequence comprises starting value(s) and a relationship between the a_n 's.

Examples:

$$\begin{cases} a_{n+1} = 2a_n, \\ a_1 = 2. \end{cases}$$

Defines the sequence

$$(a_n)_{n \in \mathbb{N}} = 2, 4, 8, 16, \dots,$$

Another example

$$\begin{cases} a_{n+1} = -a_n + a_{n-1}, \\ a_2 = 1, \\ a_1 = 0. \end{cases}$$

Defines the sequence

$$a_1 = 0$$

$$a_2 = 1$$

$$a_3 = -1 + 0 = -1$$

$$a_4 = -(-1) + 1 = 2$$

$$a_5 = -2 + (-1) = -3$$

⋮ ⋮

B2: Sequences

Text Reference (Epp)

- | | |
|---------------|--|
| 3ed: Sections | 4.1-4, 8.1-3 (Sequences and induction),
9.3,5 (Sorting) |
| 4ed: Sections | 5.1-4,6-8, (Sequences and induction),
11.3,5 (Sorting) |
| 5ed: Sections | 5.1-4,6-7, (Sequences and induction),
11.3,5 (Sorting) |

Sequences

Let S be a set and $I \subseteq \mathbb{N}^*$. A function $a : I \rightarrow S$ is called a **sequence in S** . Special **sequence notation** is often used:

Function notation	Sequence notation
$a : I \rightarrow S$ $n \mapsto a(n).$	$(a_n)_{n \in I} \subseteq S$

The notation $(a_n)_{n \in I}$ indicates that the function can be represented as an *ordered -tuple* or, more simply, as a *list*.

(Unlike a *set*, a *list* has an order, and can have repeated entries.)

Examples

- $I = \{1, 2, 3\}$: $(a_n)_{n \in I} = (a_1, a_2, a_3)$.
- $I = \mathbb{N}^*$: $(a_n)_{n \in I} = (a_0, a_1, a_2, \dots \dots)$.

In practice we usually leave out the parentheses and speak of “the sequence a_1, a_2, a_3 ” or “the sequence $a_0, a_1, a_2, \dots \dots$ ”

An agreed upon abuse of notation

The “ $\subseteq S$ ” part of the sequence notation $(a_n)_{n \in I} \subseteq S$ indicates that the sequence members belong to S ; i.e. that the range of the sequence function $a : I \rightarrow S$ is a subset of its codomain S .

The sequence *itself* is **not** a subset of S , since it is not a set.

Examples

1. Suppose n represents time (in months since January 1, 2000) and a_n is the standard savings account interest rate offered by bank X at time n . Then $(a_n)_{n \in \mathbb{N}} \subseteq \mathbb{Q}$ is a sequence of interests rates since 2000 and into the future!

For example, a_{17} is the standard savings account interest rate offered by bank X on 1 June, 2001.

Examples

2. Suppose n represents time (in months since January 1, 2000) and a_n, f_n, z_n represent the populations of amphibians, fish and zooplankton in a particular lake ecosystem at time n . Let $p_n = (a_n, f_n, z_n)$. Then $(p_n)_{n \in \mathbb{N}} \subseteq \mathbb{N}^* \times \mathbb{N}^* \times \mathbb{N}^*$ is a sequence of states of the ecosystem since 2000 and into the future!

Examples

3. For each $n \in \mathbb{N}$, let a_n denote the amplitude of the harmonic of frequency $n \times f$ (where f is the fundamental frequency). Then $(a_n)_{n \in \mathbb{N}} \subseteq \mathbb{N}^*$ is a sequence of amplitudes.
4. Let U be a set of users, then $(u_n)_{n \in \{1,2,3,4,5\}} \subseteq U$ is a list of 5 users.

In examples 1, 2, 3 the indexing variable n had some intuitive meaning; in example 4 the indexing variable did not necessarily have an intuitive meaning other than we have ordered the 5 interesting users into the first, second, third, fourth and fifth user.

Describing sequences: explicit definitions

An **explicit definition** of a sequence is a formula for a_n .

Examples:

1. For all $n \in \mathbb{N}$, let $a_n = 2^n$. Then

$$(a_n)_{n \in \mathbb{N}} = 2, 4, 8, 16, \dots$$

2. Let $a_1 = \text{Pierre}$, $a_2 = \text{Julie}$, $a_3 = \text{Paul}$. Then
 $(a_n)_{n \in \{1,2,3\}} = \text{Pierre, Julie, Paul.}$

Describing sequences: Implicit definitions

An **implicit definition** of a sequence comprises starting value(s) and a relationship between the a_n 's.

Examples: Let $(a_n)_{n \in \mathbb{N}}$ be the sequence such that:

$$\begin{cases} a_1 = 2, \text{ and} \\ \forall n \in \mathbb{N} \ a_{n+1} = 2a_n. \end{cases}$$

This defines the sequence

$$(a_n)_{n \in \mathbb{N}} = 2, 4, 8, 16, \dots,$$

Another example

Let $(a_n)_{n \in \mathbb{N}}$ be the sequence such that:

$$\begin{cases} a_1 = 0, \\ a_2 = 1, \text{ and} \\ \forall n \in \{2, 3, 4, \dots\} \quad a_{n+1} = -a_n + a_{n-1}. \end{cases}$$

Defines the sequence

$$a_1 = 0$$

$$a_2 = 1$$

$$a_3 = -1 + 0 = -1$$

$$a_4 = -(-1) + 1 = 2$$

$$a_5 = -2 + (-1) = -3$$

⋮ ⋮

Proofs about sequences

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \ P(n) ?$

METHOD 1:

Introduce a fixed but arbitrary variable: Let $n \in \mathbb{N}$.
you are now working with a fixed but arbitrary value of n .

Deduce $P(n)$ from what you know: *Insert
mathemagic here.*

Victory lap: Since $P(n)$ hold for a fixed but arbitrary choice $n \in \mathbb{N}$, $P(n)$ holds for all $n \in \mathbb{N}$. *No one write this,
but this is why the method works.*

Method 2:

The basis step Prove $P(1)$.

The inductive step Prove

$$\forall n \in \mathbb{N} \quad \left(P(1) \wedge P(2) \wedge P(3) \wedge \dots \wedge P(n) \right) \Rightarrow P(n+1)$$

Let $n \in \mathbb{N}$. Suppose that all of the statements $P(1)$, $P(2)$, ..., $P(n)$ are true. Now deduce $P(n+1)$ making use somewhere of one or more of the facts $P(1), \dots, P(n)$.

The victory lap By the Principle of Mathematical Induction, $P(n)$ is true for all $n \in \mathbb{N}$.

(This is also known as *strong mathematical induction*.)

Why does induction work?

Suppose that you have completed the base step and the inductive step.

From the basis step, we know that $P(1)$ is true.

From the inductive step, we know that $P(1) \Rightarrow P(2)$. Since $P(1)$ is true and $P(1) \Rightarrow P(2)$, we deduce that $P(2)$ is true.

From the inductive step, we know that $P(2) \Rightarrow P(3)$. Since $P(2)$ is true and $P(2) \Rightarrow P(3)$, we deduce that $P(3)$ is true.

Continuing to argue in this manner gives $P(n)$ for all $n \in \mathbb{N}$.

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by

$$\begin{cases} a_{n+1} = 3a_n & \forall n \in \mathbb{N}, \\ a_1 = 3. \end{cases}$$

Can we get an explicit definition?

First generate some values:

$$a_1 = 3, a_2 = 9, a_3 = 27, a_4 = 81, \dots$$

Now we make a claim/hypothesis/informed guess:

$$\forall n \in \mathbb{N} \ a_n = 3^n.$$

Proof that the claim is correct

We shall prove the claim using mathematical induction.

Basis step: For $n=1$, formula gives $a_1 = 3^1 = 3$, agreeing with the implicit definition.

Inductive step: Let $n \in \mathbb{N}$. Suppose that the formula is correct for a_1, a_2, \dots, a_n . Then

$$\begin{aligned} a_{n+1} &= 3a_n \quad (\text{from the implicit definition}) \\ &= 3(3^n) \quad (\text{by the inductive assumption}) \\ &= 3^{n+1} \end{aligned}$$

and so the formula is also correct for $n+1$.

By the Principle of Mathematical Induction, the formula is correct for all $n \in \mathbb{N}$.

Sum and products of terms

Terms of a sequence can be summed: $a_1 + a_2 + a_3 + \dots$
or multiplied: $a_1 \times a_2 \times a_3 \times \dots$. We use the special
notation

$$\sum_{n=1}^k a_n = a_1 + a_2 + a_3 + \dots + a_k,$$

$$\prod_{n=1}^k a_n = a_1 \times a_2 \times a_3 \times \dots \times a_k.$$

Examples

$$1. \sum_{n=1}^{10} n = 1 + 2 + 3 + 4 + \dots + 9 + 10 = 55.$$

$$2. \sum_{n=0}^7 2^n = 1 + 2 + 4 + 8 + \dots + 128 = 255.$$

$$3. \prod_{n=1}^5 n = 1 \times 2 \times 3 \times 4 \times 5 = 5! = 120.$$

$$4. \prod_{n=1}^8 n^2 = 4 \times 9 \times 16 \times \dots \times 64 = 1\,625\,702\,400.$$

Geometric sequences

Given a set of integers $K = \{n \in \mathbb{Z} \mid n \geq k\}$, a sequence $(a_n)_{n \in K} \subseteq \mathbb{R}$ is a **geometric sequence** when there exist $a, r \in \mathbb{R}$ such that

$$\begin{cases} a_k = a, \text{ and} \\ \forall k \in K \quad a_{k+1} = r a_k \end{cases}$$

We call a the **first term** and r the **common ratio** of the geometric sequence.

A geometric sequence can also be defined explicitly:

$$\forall n \in K \quad a_n = ar^{n-k}$$

Arithmetic Sequences

Implicit definition: Choose a **first index** $k \in \mathbb{Z}$, a **first term** $a \in \mathbb{R}$, and a **common difference** $d \in \mathbb{R}$. The sequence

$$(a_n)_{n \in \{k, k+1, k+2, \dots\}} \quad (\text{often written } (a_n)_{n \geq k})$$

defined by

$$\begin{cases} a_k = a \\ \forall n \geq k \ a_{n+1} = a_n + d \end{cases}$$

is called a **arithmetic sequence**.

Explicit definition: For the arithmetic sequence above, we have

$$\forall n \geq k \ a_n = a + (n - k)d.$$

Arithmetic Series

When the terms of an arithmetic series are summed, we get an **arithmetic series**.

An arithmetic series of N terms:

$$\begin{aligned} & a + (a + d) + (a + 2d) + \cdots + (a + (N - 1)d) \\ &= \sum_{n=k}^{k+(N-1)} [a + (n - k)d] \\ &= N \left[\frac{a + a + (N - 1)d}{2} \right] \end{aligned}$$

The formula in words: The sum of the first N terms of an arithmetic sequence is equal to N times the average of the first term and the N term.

Geometric Sequences

Implicit definition: Choose a **first index** $k \in \mathbb{Z}$, a **first term** $a \in \mathbb{R}$, and a **common ratio** $r \in \mathbb{R}$. The sequence

$$(a_n)_{n \in \{k, k+1, k+2, \dots\}} \quad (\text{often written } (a_n)_{n \geq k})$$

defined by

$$\begin{cases} a_k = a \\ \forall n \geq k \ a_{n+1} = r a_n \end{cases}$$

is called a **geometric sequence**.

Explicit definition: For the geometric sequence above, we have

$$\forall n \geq k \ a_n = ar^{n-k}.$$

Geometric Series

When the terms of a geometric series are summed, we get a **geometric series**.

An geometric series of N terms:

$$\begin{aligned} & a + ar + ar^2 + ar^3 + \cdots + ar^{N-1} \\ &= \sum_{n=k}^{k+(N-1)} [ar^{n-k}] \\ &= \begin{cases} \frac{a(1 - r^N)}{(1 - r)}, & \text{if } r \neq 1 \\ Na, & \text{if } r = 1. \end{cases} \end{aligned}$$

Mixed Geometric-Arithmetic Sequences

Implicit definition: Choose a **first index** $k \in \mathbb{Z}$, a **first term** $a \in \mathbb{R}$, a **multiplication factor** $r \in \mathbb{R}$ and an **offset** $d \in \mathbb{R}$. The sequence

$$(a_n)_{n \in \{k, k+1, k+2, \dots\}} \quad (\text{often written } (a_n)_{n \geq k})$$

defined by

$$\begin{cases} a_k = a \\ \forall n \geq k \quad a_{n+1} = ra_n + d \end{cases}$$

is called a **mixed geometric-arithmetic sequence**.

Explicit definition: For the geometric sequence above, we have

$$\forall n \geq k \quad a_n = \begin{cases} ar^{n-k} + \left(\frac{1 - r^{n-k}}{1 - r}\right)d, & \text{if } r \neq 1 \\ a + (n - k)d, & \text{if } r = 1 \end{cases}$$

Simplest Malthusian model in population dynamics

(Exponential growth)



T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, i.e. $\forall n \in \mathbb{N}^* \ p_{n+1} = 2p_n$.

Assume the current population is five million. i.e. $p_0 = 5 \times 10^6$.

- Questions:
1. What will the population size be in 10 years time?
 2. When will the population size reach 10^9 ?

We have a geometric sequence with $k = 0$, $a = 5 \times 10^6$, $r = 2$. Hence:

Answers:

- $a_{10} = 5 \times 10^6 \times 2^{10} = 5 \cdot 12 \times 10^9$

2. We need N so that $a_N \geq 10^9 > a_{n_1}$.

From 1. $a_8 = \frac{1}{4}(5 \cdot 12 \times 10^9) = 1.28 \times 10^9$

and $a_7 = \frac{1}{8}(5 \cdot 12 \times 10^9) = 0.64 \times 10^9$.

So $N = 8$ i.e. eight years.

Population Growth

Suppose that, without immigration, a population would double every year; and suppose that there is immigration bringing 10^3 new individuals to the population each year. The population may be modelled by a mixed geometric-arithmetic sequence $(p_n)_{n \in \mathbb{N}^*}$ with

$$\begin{cases} p_{n+1} = 2p_n + 10^3 & \forall n \in \mathbb{N}^*, \\ p_0 = 5 \times 10^6. \end{cases}$$

Applying our formula, we know that

$$\forall n \in \mathbb{N}^* \quad p_n = 2^n p_0 + (2^n - 1)d$$

Sorting algorithms

Let $N \in \mathbb{N}$, S be a set, and $(x_n)_{n \in \{1, \dots, N\}} \subseteq S$.

Remember that this just means that $x_n \in S$ for each $n \in \{1, \dots, N\}$;
it does not imply that all the x_n 's are different.

So *sequences may contain some elements more than once*.

A **sorting algorithm** is a procedure for sorting a sequence into increasing order according to some specified ordering rule (e.g. numerical, alphabetical, etc.) i.e. it replaces $(x_n)_{n \in \{1, \dots, N\}}$ by a rearrangement $(y_n)_{n \in \{1, \dots, N\}}$ with

$$y_1 \leq y_2 \leq y_3 \cdots y_{N-1} \leq y_N$$

where " \leq " denotes the ordering rule.

Example:

$$(x_n)_{n \in \{1, \dots, 5\}} = \text{Jane, Fred, Jo, Jane, Ann}$$

$$(y_n)_{n \in \{1, \dots, 5\}} = \text{Ann, Fred, Jane, Jane, Jo} \quad (\text{in alphabetical order})$$

Sorting preliminaries

An **index set** I is a set of the form

$$I = \{i \in \mathbb{N}^* : s \leq i \leq f\} = \{s, \dots, f\}$$

where $s, f \in \mathbb{N}^*$, $s \leq f$, are the **start index** and the **finish index**.

Example: $I = \{3, 4, 5, 6\}$ ($s = 3, f = 6$)

For $I = \{s, \dots, f\}$ we may denote the sequence $(a_n)_{n \in I}$ by $(a_n)_{s..f}$.

Example: Suppose $\forall n \in \mathbb{N} a_n = 2n + 1$. Then $(a_n)_{3..6} = 7, 9, 11, 13$.

An **index permutation** on an index set I is a bijection (one-to-one correspondence) $\pi : I \rightarrow I$. For $I = \{s, \dots, f\}$ the permutation can be specified using the notation

$$\pi = \begin{pmatrix} s & s+1 & \dots & f \\ \pi(s) & \pi(s+1) & \dots & \pi(f) \end{pmatrix}.$$

Example:

$\pi = \begin{pmatrix} 3 & 4 & 5 & 6 \\ 6 & 4 & 3 & 5 \end{pmatrix}$ means $I = \{3, 4, 5, 6\}$
 $\pi(3) = 6, \pi(4) = 4, \pi(5) = 3, \pi(6) = 5$.

More sorting preliminaries

Using index permutations for sorting has two benefits:

- it allows for more precise algorithm specification: and
- items being sorted do not get moved - only their indices are affected. This is valuable when the items have long and/or variable storage length.

A **reordering** of a sequence $(x_n)_{s..t}$ is a sequence $(y_n)_{s..t}$ where $y_n = x_{\pi(n)}$ for some index permutation π .

The reordering of a sequence $(a_n)_{s..t}$ can be denoted by $(a_{\pi(n)})_{s..t}$.

Example: For $\pi = \begin{pmatrix} 3 & 4 & 5 & 6 \\ 6 & 4 & 3 & 5 \end{pmatrix}$, if $(a_n)_{3..6} = 7, 9, 11, 13$.
then $(a_{\pi(n)})_{3..6} = 13, 9, 7, 11$

Example: (names example recast using an index permutation)

If $(x_n)_{n \in \{1, \dots, 5\}} = \text{Jane, Fred, Jo, Jane, Ann}$

then $(x_{\pi(n)})_{n \in \{1, \dots, 5\}} = \text{Ann, Fred, Jane, Jane, Jo}$

where $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 2 & 1 & 4 & 3 \end{pmatrix}$ sorts the sequence into alphabetical order.

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean “assign a the value b , leaving b unchanged”. (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule “ \leq ” for S and an index function π on $\{s, \dots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \dots, f$.

Method:

$i \leftarrow s + 1$. [Initialisation]
 $m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.
 $i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Example

Example: ($s=1, f=6$)

	i	1	2	3	4	5	6
BEFORE	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C

Trace:	i	2	3	4	5	6	7
	m	1	2	3	3	5	5
	$x_{\pi(i)}$	D	C	E	B	C	-
	$x_{\pi(m)}$	F	D	C	C	B	B

	i	1	2	3	4	5	6
AFTER	$\pi(i)$	5	2	3	4	1	6
	$x_{\pi(i)}$	B	D	C	E	F	C

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$,
an ordering rule " \leq " for
 S and an index function
 π on $\{1, \dots, n\}$.

Output: Modification to π ,
so that $(x_{\pi(i)})_{1..n}$ is in
non-decreasing order
 $x_{\pi(1)} \leq x_{\pi(2)} \leq \dots \leq x_{\pi(n)}$.

Method:

$s \leftarrow 1$ [Initialisation]

Loop: If $s = n$ stop.

Run least element

algorithm on $(x_{\pi(i)})_{s..n}$

$s \leftarrow s + 1$

Repeat loop

Example: $i: 1 2 3 4 5 6$

Input: $(n = 6)$	$\pi(i)$ $x_{\pi(i)}$	1 2 3 4 5 6 F D C E B C
---------------------	--------------------------	----------------------------

$s = 1$

After 1st iteration	$\pi(i)$ $x_{\pi(i)}$	5 2 3 4 1 6 B D C E F C
---------------------	--------------------------	----------------------------

$s = 2$

After 2nd iteration	$\pi(i)$ $x_{\pi(i)}$	5 3 2 4 1 6 B C D E F C
---------------------	--------------------------	----------------------------

$s = 3$

After 3rd iteration	$\pi(i)$ $x_{\pi(i)}$	5 3 6 4 1 2 B C C E F D
---------------------	--------------------------	----------------------------

$s = 4$

After 4th iteration	$\pi(i)$ $x_{\pi(i)}$	5 3 6 2 1 4 B C C D F E
---------------------	--------------------------	----------------------------

$s = 5$

After final iteration	$\pi(i)$ $x_{\pi(i)}$	5 3 6 2 4 1 B C C D E F
-----------------------	--------------------------	----------------------------

Selection sort: number of operations

How many operations are required by Selection Sort?

By *operation* here we mean any comparison step;

i.e. a step of the form “If $x_{\pi(i)} \leq x_{\pi(j)}$ then ...”

The loop of the Selection Sort algorithm is iterated $n-1$ times; once each for $s = 1, \dots, n-1$.

Iteration s runs the Least Element algorithm on $(x_{\pi(i)})_{s..n}$ and so uses $n-s$ comparisons.

So: 1st iteration uses $n-1$ comparisons

 2nd iteration uses $n-2$ comparisons

⋮ ⋮ ⋮ ⋮

 last iteration uses 1 comparison

Hence the total number of comparisons, T_n say, is given by

$$1 + 2 + \cdots + (n-1) = (n-1) \left(\frac{1+(n-1)}{2} \right) \text{ (sum of an arithmetic series).}$$

That is: $\forall n \in \mathbb{N} \quad T_n = \frac{n(n-1)}{2}$.

Other sorting algorithms

There are many different sorting algorithms, with various pros and cons. A full study of the topic belongs in course on algorithms and data structures.

We will look at just one more; “Merge Sort”.

This will provide us with an opportunity to compare two algorithms designed to do the same job – what are their respective advantages and disadvantages?

In order to keep the description simple, I will not use an indexing function π in specifying the algorithm, though it is possible, and often preferable, to do so.

As with Selection Sort, Merge Sort makes use of a sub-algorithm, which we treat first.

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i \in \{1, \dots, n\}} \subseteq S$ and $(b_j)_{j \in \{1, \dots, p\}} \subseteq S$ pre-sorted according to an ordering rule " \leq " on S .

Output: In-order list (sorted sequence) $(z_k)_{k \in \{1, \dots, n+p\}}$ that merges the two input lists.

Method:

$i, j, k \leftarrow 1$. [Initialize the indeces for the a -, b - and z - lists]

Loop: If $k = n+p+1$ stop. [there will be $n+p$ items in the z -list]

If $i = n+1$ then $[z_k \leftarrow b_j, j \leftarrow j+1]$ [a -list empty; take from b -list]

Else if $j = p+1$ then $[z_k \leftarrow a_i, i \leftarrow i+1]$ [b -list empty; take from a -list]

Else if $a_i < b_j$ then $[z_k \leftarrow a_i, i \leftarrow i+1]$ [a -list item less; take it]

Else $[z_k \leftarrow b_j, j \leftarrow j+1]$ [else take item from b -list]

$k \leftarrow k+1$ [prepare to add next item to z -list]

Repeat loop.

Merge algorithm: example of execution

Example: Merge $(1, 3, 7)$ and $(2, 3, 6, 8, 9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	(z_1, \dots, z_{k-1})
0	1	1	1	(1, 3, 7)	(2, 3, 6, 8, 9)	()
1	2	1	2	(1, 3, 7)	(2, 3, 6, 8, 9)	(1)
2	2	2	3	(1, 3, 7)	(2, 3, 6, 8, 9)	(1, 2)
3	2	3	4	(1, 3, 7)	(2, 3, 6, 8, 9)	(1, 2, 3)
4	3	3	5	(1, 3, 7)	(2, 3, 6, 8, 9)	(1, 2, 3, 3)
5	3	4	6	(1, 3, 7)	(2, 3, 6, 8, 9)	(1, 2, 3, 3, 6)
6	4	4	7	(1, 3, 7)	(2, 3, 6, 8, 9)	(1, 2, 3, 3, 6, 7)
7	4	5	8	(1, 3, 7)	(2, 3, 6, 8, 9)	(1, 2, 3, 3, 6, 7, 8)
8	4	6	9	(1, 3, 7)	(2, 3, 6, 8, 9)	(1, 2, 3, 3, 6, 7, 8, 9)

Merge Sort algorithm

Input: $r \in \mathbb{N}$, $(x_n)_{n \in \{1, \dots, 2^r\}} \subseteq S$, and an ordering rule " \leq " for S .
(For lists whose length is not a power of 2, see workshop question.)

Output: In-order list (sorted sequence) $(z_n)_{n \in \{1, \dots, 2^r\}}$ that
is a rearrangement of the input list.

Method: There are r steps.

If $r < 3$ adjust the description below accordingly.

- Step 1: Apply the Merge algorithm 2^{r-1} times with inputs $\{(x_1), (x_2)\}, \{(x_3), (x_4)\}, \dots, \{(x_{2^{r-1}}), (x_{2^r})\}$.
This gives 2^{r-1} in-order lists of length 2.
- Step 2: Apply the Merge algorithm 2^{r-2} times with pairs of these lists as input.
This gives 2^{r-2} in-order lists of length $2 \times 2 = 2^2$.
- Steps 3 to r : Continue in this vein until you have just one ($= 2^{r-r}$) in-order list with 2^r elements.

Merge sort: example

Example: merge sort (1, 2, 6, 1, 7, 9, 4, 5).

Step 1:

Merging $\{(1), (2)\}$ gives (1, 2).

Merging $\{(6), (1)\}$ gives (1, 6).

Merging $\{(7), (9)\}$ gives (7, 9).

Merging $\{(4), (5)\}$ gives (4, 5).

Step 2:

Merging $\{(1, 2), (1, 6)\}$ gives (1, 1, 2, 6).

Merging $\{(7, 9), (4, 5)\}$ gives (4, 5, 7, 9).

Step 3:

Merging $\{(1, 1, 2, 6), (4, 5, 7, 9)\}$ gives (1, 1, 2, 4, 5, 6, 7, 9).

Merge sort: counting comparisons

As with Selection Sort, we can analyze the complexity of Merge sort by counting the number of comparisons involved.

Revisiting the previous example, merge sort (1, 2, 6, 1, 7, 9, 4, 5):

(1) (2) (6) (1) (7) (9) (4) (5)

(1, 2) (1, 6) (7, 9) (4, 5) $1+1+1+1=4$ comps

(1, 1, 2, 6) (4, 5, 7, 9) $3+2=5$ comps

(1, 1, 2, 4, 5, 6, 7, 9) 6 comps

TOTAL: 15 comparisons

Note for example that when merging (7, 9) and (4, 5) only 2 comparisons are used:

7 and 4 are compared; 4 is transferred

7 and 5 are compared; 5 is transferred

7 and 9 are transferred without comparison (other list exhausted.)

Merge sort: number of operations

Since the number of comparisons used to Merge Sort a list of length n depends to some extent on the nature of the list, there is no precise formula for this number as there is with Selection Sort. However an upper bound is given by the number of *transfers*.

Let T_r denote the number of transfers required to sort a sequence of length 2^r using Merge sort.

Now $2^{r-1} + 2^{r-1} = 2^r$ transfers are required for the Merge algorithm to merge two sequences of length 2^{r-1} , so an implicit

definition for T_r is
$$\begin{cases} T_r = 2^r + 2T_{r-1} & \forall r \in \mathbb{N} \setminus \{1\} \\ T_1 = 2. \end{cases}$$

$$\text{So } T_1 = 2, \quad T_2 = 2^2 + 4 = 8, \quad T_3 = 2^3 + 2 \times 8 = 3 \times 2^3,$$

$$T_4 = 2^4 + 2 \times (3 \times 2^3) = 4 \times 2^4, \quad \dots$$

Claim: $\forall r \in \mathbb{N} \quad T_r = r2^r$. Verify by induction!

(The sequence $(T_r)_{r \in \mathbb{N}}$ is neither geometric, arithmetic nor mixed.)

Merge Sort and Selection Sort compared

Merge Sort sorts a sequence of length 2^r with less than $r2^r$ transfers.

For the same length, Selection Sort uses

$$\frac{2^r(2^r - 1)}{2} = \frac{2^2r - 2^r}{2} = 2^{2r} - 2^r \text{ comparisons.}$$

Merge sort is **much faster**, e.g. To sort a list with $N = 1024$ objects ($r = 10$):

Selection sort uses $\frac{2^r(2^r - 1)}{2} = 523\,776$ comparisons.

Merge sort uses $r2^r = 10\,240$ transfers

Notes: Merge sort can be modified to work even faster on machines with parallel processing capabilities; there is no direct way to use parallel processing with Selection sort. Selection Sort may be simpler to program; for short lists on high speed computers, slower speed may not matter.

Companion to Lecture 14

Adam Piggott

March 2021

The following result accompanies slide #1.

Lemma: Let $k \in \mathbb{Z}$, let $a \in \mathbb{R}$, let $d \in \mathbb{R}$, and consider the sequence

$$(a_n)_{n \in \{k, k+1, k+2, \dots\}} \quad (\text{often written } (a_n)_{n \geq k})$$

defined by

$$\begin{cases} a_k = a \\ \forall n \geq k \ a_{n+1} = a_n + d \end{cases}$$

Then

$$\forall n \geq k \ a_n = a + (n - k)d.$$

Proof. Let

$$p(n) : a_n = a + (n - k)d$$

We shall use mathematical induction to prove that $p(n)$ is true for $n \geq k$.

Basis step: The definition of the sequence give that $a_k = a$, whereas $p(k)$ claims

$$a_k = a + (k - k)d = a + (0)d = a.$$

Hence $p(k)$ is true.

Inductive step: Let $n \in \{k, k+1, k+2, \dots\}$ and suppose that $p(k), p(k+1), \dots, p(n)$ are all true. Now

$$\begin{aligned} a_{n+1} &= a_n + d && (\text{using the definition}) \\ &= (a + (n - k)d) + d && (\text{using } p(n)) \\ &= a + (n - k)d + d \\ &= a + ((n + 1) - k)d. \end{aligned}$$

Hence $p(n+1)$ is true.

By the principle of mathematical induction, $p(n)$ is true for all $n \geq k$. \square

The following result accompanies slide #2.

Lemma: Let $k \in \mathbb{Z}$, let $a \in \mathbb{R}$, let $d \in \mathbb{R}$. Then

$$\forall N \in \mathbb{N} \quad \sum_{n=k}^{k+(N-1)} [a + (n - k)d] = N \left[\frac{a + a + (N - 1)d}{2} \right]$$

Proof. Let

$$p(N) : \sum_{n=k}^{k+(N-1)} [a + (n - k)d] = N \left[\frac{a + a + (N - 1)d}{2} \right].$$

We shall use mathematical induction to prove that $p(N)$ is true for all $N \in \mathbb{N}$.

Basis step: When $N = 1$, the left-hand side (LHS) of the formula is

$$\sum_{n=k}^{k+(1-1)} [a + (n - k)d] = \sum_{n=k}^k [a + (n - k)d] = a + (k - k)d = a,$$

and the right-hand side (RHS) of the formula is

$$1 \times \left[\frac{a + a + (1 - 1)d}{2} \right] = \left[\frac{2a}{2} \right] = a.$$

Hence $p(1)$ is true.

Inductive step: Let $N \in \mathbb{N}$. Suppose that $p(1), p(2), \dots, p(N)$ are all true. Now

$$\begin{aligned} & \text{The left-hand side of } p(N+1) \\ &= \sum_{n=k}^{k+((N+1)-1)} [a + (n - k)d] \\ &= \sum_{n=k}^{k+N} [a + (n - k)d] \\ &= \left(\sum_{n=k}^{k+(N-1)} [a + (n - k)d] \right) + a + (k + N - k)d \quad (\text{moving the last term out of the sum}) \\ &= N \left[\frac{a + a + (N - 1)d}{2} \right] + a + (k + N - k)d \quad (\text{using } p(N)) \\ &= N \left[\frac{a + a + (N - 1)d}{2} \right] + a + Nd \\ &= N \left[\frac{a + a + (N - 1)d}{2} \right] + a + Nd \end{aligned}$$

$$\begin{aligned}
&= \frac{Na + Na + N(N-1)d}{2} + \frac{2a + 2Nd}{2} \\
&= \frac{Na + Na + N^2d - Nd + 2a + 2Nd}{2} \\
&= \frac{Na + a + Na + a + N^2d + Nd}{2} \\
&= \frac{(N+1)a + (N+1)a + (N+1)Nd}{2} \\
&= (N+1) \left[\frac{a + a + ((N+1)-1)d}{2} \right]
\end{aligned}$$

The right-hand side of $p(N+1)$

Hence $p(N+1)$ is true.

By the principle of mathematical induction, $p(N)$ is true for all $N \in \mathbb{N}$. \square

The following result accompanies slide #3.

Lemma: Let $k \in \mathbb{Z}$, let $a \in \mathbb{R}$, let $r \in \mathbb{R}$, and consider the sequence

$$(a_n)_{n \in \{k, k+1, k+2, \dots\}}$$

defined by

$$\begin{cases} a_k = a \\ \forall n \geq k \quad a_{n+1} = ra_n \end{cases}$$

Then

$$\forall n \geq k \quad a_n = ar^{n-k}.$$

Proof. Let

$$p(n) : a_n = ar^{n-k}$$

We shall use mathematical induction to prove that $p(n)$ is true for $n \geq k$.

Basis step: The definition of the sequence give that $a_k = a$, whereas $p(k)$ claims

$$a_k = ar^{k-k} = ar^0 = a$$

(because any real number raised to the power 0 is equal to 1). Hence $p(k)$ is true.

Inductive step: Let $n \in \{k, k+1, k+2, \dots\}$ and suppose that $p(k), p(k+1), \dots, p(n)$ are all true. Now

$$\begin{aligned}
a_{n+1} &= ra_n && \text{(using the definition)} \\
&= r(ar^{n-k}) && \text{(using } p(n) \text{)} \\
&= ar^{(n+1)-k}
\end{aligned}$$

Hence $p(n+1)$ is true.

By the principle of mathematical induction, $p(n)$ is true for all $n \geq k$. \square

The following result accompanies slide #4

Lemma: Let $k \in \mathbb{Z}$, let $a \in \mathbb{R}$, let $r \in \mathbb{R}$. Then

$$\forall N \in \mathbb{N} \quad \sum_{n=k}^{k+(N-1)} [ar^{n-k}] = \begin{cases} \frac{a(1-r^N)}{(1-r)}, & \text{if } r \neq 1 \\ Na, & \text{if } r = 1. \end{cases}$$

Proof. Let

$$p(N) : \sum_{n=k}^{k+(N-1)} [ar^{n-k}] = \begin{cases} \frac{a(1-r^N)}{(1-r)}, & \text{if } r \neq 1 \\ Na, & \text{if } r = 1. \end{cases}$$

We shall give a proof by cases. First consider the case that $r = 1$. Then the left-hand side of $p(N)$ simplifies to

$$\sum_{n=k}^{k+(N-1)} [ar^{n-k}] = \sum_{n=k}^{k+(N-1)} [a(1)^{n-k}] = \sum_{n=k}^{k+(N-1)} a = Na$$

and the right-hand side of $p(N)$ is equal to Na . Hence $p(N)$ hold in this case.

Now consider the case that $r \neq 1$. We shall use mathematical induction to prove that $p(N)$ is true for all $N \in \mathbb{N}$.

Basis step: When $N = 1$, the left-hand side (LHS) of the formula is

$$\sum_{n=k}^{k+(1-1)} [ar^{n-k}] = \sum_{n=k}^k [ar^{n-k}] = ar^0 = a,$$

and the right-hand side (RHS) of the formula is

$$\frac{a(1-r^1)}{(1-r)} = a.$$

Hence $p(1)$ is true.

Inductive step: Let $N \in \mathbb{N}$. Suppose that $p(1), p(2), \dots, p(N)$ are all true.

Now

$$\begin{aligned}
 & \text{The left-hand side of } p(N+1) \\
 &= \sum_{n=k}^{k+((N+1)-1)} [ar^{n-k}] \\
 &= \sum_{n=k}^{k+N} [ar^{n-k}] \\
 &= \sum_{n=k}^{k+(N-1)} [ar^{n-k}] + ar^{(k+N)-k} && \text{(taking the last term out of the sum)} \\
 &= \frac{a(1 - r^N)}{(1 - r)} + ar^N && \text{(using } p(N)) \\
 &= \frac{a - ar^N + ar^N - ar^{N+1}}{(1 - r)} \\
 &= \frac{a - ar^{N+1}}{(1 - r)} \\
 &= \frac{a(1 - r^{N+1})}{(1 - r)} \\
 &= \text{The right-hand side of } p(N+1)
 \end{aligned}$$

Hence $p(N+1)$ is true.

By the principle of mathematical induction, $p(N)$ is true for all $N \in \mathbb{N}$. \square

Proving the result stated on Slide #5 is left as a challenge exercise for the reader.

B3. Matrices.

Notes originally prepared by Pierre Portal.
Editing and expansion by Malcolm Brooks.

Text Reference (Epp) 3ed: Section 11.3
 4ed: Section 10.3
 5ed: Section 10.2

Unfortunately these sections are part of chapters on Graph Theory,
that we have not yet covered, so the examples may seem
unfamiliar.

Also they do not go quite as far as we do, in that matrix inverses
are not discussed.

What is a matrix (plural: matrices) ?

Definition: Let S be a set, and $m, n \in \mathbb{N}$.

An **$m \times n$ matrix** (over S) is a rectangular array of members of S , the array having m rows and n columns. The array is enclosed left and right with parentheses or brackets. The expression " $m \times n$ " describes the **shape** of the matrix. Examples of various shapes are:

$$\mathbf{A} = \begin{bmatrix} 1 & -2 & 3 \\ -4 & 5 & -6 \end{bmatrix}$$

\mathbf{A} is a 2×3 matrix over \mathbb{Z}

$$\mathbf{B} = \begin{bmatrix} \pi/2 \\ -\pi/2 \end{bmatrix}$$

\mathbf{B} is a 2×1 matrix over \mathbb{R}

$$\mathbf{C} = \left(\begin{array}{ccc} \frac{1}{5} & \frac{2}{5} & \frac{2}{5} \end{array} \right)$$

\mathbf{C} is a 1×3 matrix over \mathbb{Q}

The set of all $m \times n$ matrices over S is denoted by $M_{m \times n}(S)$, so

$$\mathbf{A} \in M_{2 \times 3}(\mathbb{Z}), \quad \mathbf{B} \in M_{2 \times 1}(\mathbb{R}), \quad \mathbf{C} \in M_{1 \times 3}(\mathbb{Q}).$$

Matrices with an equal number of rows and columns, i.e. $m = n$, play a particularly important rôle in mathematics.

For such a **square matrix** \mathbf{M} over S we simply write $\mathbf{M} \in M_n(S)$.

Examples: $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \in M_2(\mathbb{N}), \quad \begin{pmatrix} a & b & c \\ c & a & b \\ b & c & a \end{pmatrix} \in M_3(\{a, b, c\})$.

Indexing

A generic member of $M_{m \times n}(S)$ is written

$$\mathbf{A} = (a_{i,j}) = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{pmatrix}$$

so that $a_{i,j} \in S$ denotes the entry in row i , column j , of \mathbf{A} .

NB: The **row index** i always comes *before* the **column index** j .

Example: For the matrix $\mathbf{A} = \begin{bmatrix} 2 & 7 \\ 0 & -3 \end{bmatrix}$ we have

$$a_{1,1} = 2, \quad a_{1,2} = 7, \quad a_{2,1} = 0, \quad a_{2,2} = -3.$$

Two dimensional information

Let S be a set, $n \in \mathbb{N}$. Elements of $S^n = \underbrace{S \times S \times \cdots \times S}_{n \text{ copies of } S}$

correspond to sequences $(a_j)_{1..n}$, i.e. functions

$$\begin{aligned} a : \{1, \dots, n\} &\rightarrow S \\ j &\mapsto a_j. \end{aligned}$$

This is 1-dimensional information: information which depends on 1 number, j .

Elements of $M_n(S)$ correspond to functions

$$\begin{aligned} a : \{1, \dots, n\} \times \{1, \dots, n\} &\rightarrow S \\ (i, j) &\mapsto a_{i,j}. \end{aligned}$$

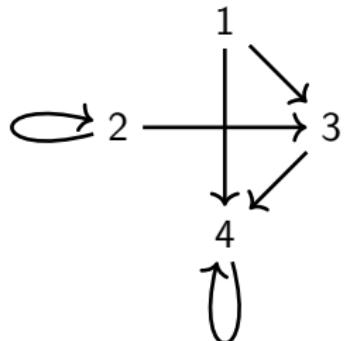
This is 2-dimensional information: information which depends on 2 numbers, i and j .

Examples

- An image can be described by the colour of each pixel.
Let C be the set of colours.
A square 1 megapixel image is an element of $M_{10^3}(C)$.
- A relation $R \subseteq \{1, \dots, n\} \times \{1, \dots, n\}$ can be represented by a matrix $(a_{i,j}) \in M_n(\{0, 1\})$ with

$$a_{i,j} = 1 \iff iRj.$$

Example:



is represented by $\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

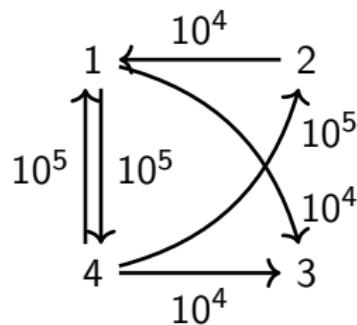
Another example

- A matrix $(a_{i,j}) \in M_n(\mathbb{Q})$ can define a weighted relation. Let us consider 4 companies, called 1,2,3,4, and let $a_{i,j}$ be the money (\$) received by i from j in a year. Then

$$\begin{pmatrix} 0 & 10^4 & 0 & 10^5 \\ 0 & 0 & 0 & 10^5 \\ 10^4 & 0 & 0 & 10^5 \\ 10^5 & 0 & 0 & 0 \end{pmatrix}$$

represents the situation where :

- 1 received $\$10^4$ from 2
and $\$10^5$ from 4,
- 2 received $\$10^5$ from 4,
- 3 received $\$10^4$ from 1
and $\$10^5$ from 4,
- 4 received $\$10^5$ from 1.



Vectors and vector arithmetic

For any $n \in \mathbb{N}$ an element $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{Q}^n$ will be called a **vector**.

A vector $\mathbf{x} \in \mathbb{Q}^n$ can be viewed as

an element of $M_{1 \times n}(\mathbb{Q})$; \mathbf{x} is then called a **row vector**
or as an element of $M_{n \times 1}(\mathbb{Q})$; \mathbf{x} is then called a **column vector**.

The **sum** of two vectors, $\mathbf{x} + \mathbf{y}$, is defined element-wise:

$$\mathbf{x} + \mathbf{y} = (x_1, \dots, x_n) + (y_1, \dots, y_n) = (x_1 + y_1, \dots, x_n + y_n).$$

(When viewed as row or column vectors, \mathbf{x} and \mathbf{y} must be the same shape.)

There are a number of ways to define the product of two vectors (e.g the 'inner' and the 'outer' products) but we will not use them in this course. However we do need to define the product of a number λ and a vector. In this context the number λ is referred to as a **scalar**, to distinguish it from a vector, and the product $\lambda\mathbf{x}$ is called a **scalar product**. It is also defined element-wise:

$$\forall \lambda \in \mathbb{Q} \quad \lambda\mathbf{x} = \lambda(x_1, \dots, x_n) = (\lambda x_1, \dots, \lambda x_n).$$

Examples of vectors and vector arithmetic

- Let $\mathbf{p} = (p_1, p_2, p_3) \in \mathbb{Q}^3$ represent the state of an ecosystem with p_1, p_2, p_3 being the sizes of the populations of three different species.

If p_1 increases by 10 individuals, p_2 loses 20 individuals, and p_3 gains 2, then the new state of the ecosystem is

$$\mathbf{p} + \mathbf{c} = (p_1, p_2, p_3) + (10, -20, 2)$$

- Let $\mathbf{a} = (a_1, \dots, a_n) \in \mathbb{Q}^n$ represent the amplitudes a_k , $1 \leq k \leq n$, of the harmonic frequencies kf of the fundamental frequency f of a note played on a violin. Then

$$3\mathbf{a} = 3(a_1, \dots, a_n),$$

represents to the same sound, but three times stronger.

Addition and scalar multiplication of matrices

The same can be done with matrices.

For matrices $\mathbf{A} = (a_{i,j})$ and $\mathbf{B} = (b_{i,j})$ in $M_n(\mathbb{Q})$, and $\lambda \in \mathbb{Q}$, we define the **sum** $\mathbf{A} + \mathbf{B}$ and **scalar product** $\lambda\mathbf{A}$ by

$$\boxed{\begin{aligned}\mathbf{A} + \mathbf{B} &= (a_{i,j}) + (b_{i,j}) = (a_{i,j} + b_{i,j}). \\ \lambda\mathbf{A} &= \lambda(a_{i,j}) = (\lambda a_{i,j}).\end{aligned}}$$

Examples:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$$

$$5 \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 5 & 10 \\ 15 & 20 \end{pmatrix}.$$

Linear functions

Definition: A function $F : \mathbb{Q}^n \rightarrow \mathbb{Q}^n$ is called **linear** if and only if it satisfies the following two conditions:

- $F(x + y) = F(x) + F(y) \quad \forall x, y \in \mathbb{Q}^n.$
- $F(\lambda x) = \lambda F(x) \quad \forall x \in \mathbb{Q}^n \quad \forall \lambda \in \mathbb{Q}.$

Example: For $n \in \mathbb{N}$ suppose $(a_1, \dots, a_n) \in \mathbb{Q}^n$ represents the amplitudes of the different harmonics of a note played on a violin. Thus a_n is the amplitude of frequency nf , where f is the fundamental frequency of the note.

Then for $m \in \mathbb{N}$ with $m \leq n$ the function F specified by

$$\begin{aligned} F : \mathbb{Q}^n &\rightarrow \mathbb{Q}^n \\ (a_1, \dots, a_n) &\mapsto (a_1, \dots, a_m, 0, 0, \dots 0) \end{aligned} \cdot$$

is called a **filter**. (It filters out the high frequencies).

Filters are linear functions. (Check!)

Linear functions: another example

Let $(p_n)_{n \in \mathbb{N}} \subseteq \mathbb{Q}^2$ represent the state of an ecosystem with two species at time n ; say $p_n = (x_n, y_n)$, where x_n is the size of the population of species 1, and y_n the size of the population of species 2.

Assume that the ecosystem evolves as follows, due to a predator-prey relationship between the two species:

$$\forall n \in \mathbb{N} \quad \begin{cases} x_{n+1} = 4x_n - y_n, \\ y_{n+1} = y_n + 2x_n. \end{cases}$$

Then $p_{n+1} = F(p_n) \quad \forall n \in \mathbb{N}$, where $F(x, y) = (4x - y, 2x + y)$.

The function F is linear. (Check!)

We will return to this example several times in this section on matrices.

Multiplying a vector by a matrix: motivation

We now explore the possibility of expressing the function

$$\begin{aligned} F : \mathbb{Q}^2 &\rightarrow \mathbb{Q}^2 \\ (x, y) &\mapsto (4x - y, 2x + y) \end{aligned}$$

using a matrix. We would like to write $F(x, y) = \mathbf{M}(x, y)$ for some matrix \mathbf{M} and a suitable meaning for the ‘product’ $\mathbf{M}(x, y)$.

Writing (x, y) as the column vector $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$, we want

$$\mathbf{M} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4x - y \\ 2x + y \end{pmatrix},$$

and so the ‘coefficient matrix’ $\begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix}$ looks like a good candidate for \mathbf{M} provided that we define the product $\mathbf{M}\mathbf{x}$ so that

$$\begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4x - y \\ 2x + y \end{pmatrix}.$$

That's exactly what we do next.

Multiplying a vector by a matrix: definition

For a matrix $\mathbf{A} = (a_{i,j}) \in M_n(\mathbb{Q})$ and a vector $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{Q}^n$ we define the **matrix-vector product \mathbf{Ax}** as the vector given by

$$\mathbf{Ax} = \left(\sum_{j=1}^n a_{i,j} x_j \right)_{1 \leq i \leq n} \in \mathbb{Q}^n.$$

By convention, in this context, we normally write the vectors as columns. Thus

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n \end{bmatrix}$$

Example:

$$\begin{pmatrix} 2 & 0 & -1 \\ 0 & -1 & 2 \\ -1 & 2 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} -1 \\ 4 \\ 3 \end{pmatrix}.$$

Linear functions expressed using matrices

Example: $\begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 4x - y \\ 2x + y \end{pmatrix} = F(x, y)$

where, as we have seen, the function $F : \mathbb{Q}^2 \rightarrow \mathbb{Q}^2$ so defined is linear.

This is no coincidence.

Theorem (proof omitted): To each linear function $F : \mathbb{Q}^n \rightarrow \mathbb{Q}^n$ there is a matrix $\mathbf{M} \in M_n(\mathbb{Q})$ such that

$$F(\mathbf{x}) = \mathbf{M}\mathbf{x} \quad \forall \mathbf{x} \in \mathbb{Q}^n.$$

Conversely, every function $F : \mathbb{Q}^n \rightarrow \mathbb{Q}^n$ defined using a matrix in this way is linear.

Matrix multiplication: motivation

Question: Given $\mathbf{M} \in M_n(\mathbb{Q})$, how, if at all, should \mathbf{M}^2 be defined?

Discussion: For any \mathbf{x} in \mathbb{Q}^n , \mathbf{Mx} is also in \mathbb{Q}^n and so we can consider $\mathbf{M}(\mathbf{Mx})$. Surely we would like this to equal to $\mathbf{M}^2\mathbf{x}$.

Example: For $\mathbf{M} = \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix}$:

$$\begin{aligned}\mathbf{M}(\mathbf{Mx}) &= \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \left[\begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \right] \\ &= \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 4x - y \\ 2x + y \end{pmatrix} \\ &= \begin{pmatrix} 14x - 5y \\ 10x - y \end{pmatrix} = \begin{pmatrix} 14 & -5 \\ 10 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}\end{aligned}$$

So we want $\mathbf{M}^2 = \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 14 & -5 \\ 10 & -1 \end{pmatrix}$.

Matrix multiplication: definition

For matrices $\mathbf{A} = (a_{i,j})$ and $\mathbf{B} = (b_{i,j})$ in $M_n(\mathbb{Q})$ the **product** $\mathbf{AB} = \mathbf{C} = (c_{i,j}) \in M_n(\mathbb{Q})$ is defined by

$$c_{i,j} = \sum_{k=1}^n a_{i,k} b_{k,j} \quad \forall i, j \in \{1, \dots, n\}.$$

Two Examples:

- (a) First, let's check that this formula produces what we were looking for with \mathbf{M}^2 on the previous slide:

$$\begin{aligned} \mathbf{M}^2 &= \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 4 \times 4 + (-1) \times 2 & 4 \times (-1) + (-1) \times 1 \\ 2 \times 4 + 1 \times 2 & 2 \times (-1) + 1 \times 1 \end{pmatrix} = \begin{pmatrix} 14 & -5 \\ 10 & -1 \end{pmatrix}. \end{aligned}$$

- (b) This example demonstrates the product formula more clearly:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a + 2c & b + 2d \\ 3a + 4c & 3b + 4d \end{pmatrix}.$$

Identity matrices

Observe that the matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ acts as an ‘identity’ in the sense

that $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ for any matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$.

More generally, for $n \in \mathbb{N}$, we define the $n \times n$ **identity matrix** I_n by

$$I_n = (\delta_{i,j}) \in M_n(\mathbb{Q}) \text{ with } \delta_{i,j} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

$$\text{So } I_1 = \begin{bmatrix} 1 \end{bmatrix}, \quad I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ etc.}$$

By applying the matrix product formula we can immediately establish that, for any $n \in \mathbb{N}$, the identity matrix I_n does indeed have the identity property:

$$\forall n \in \mathbb{N}, \forall \mathbf{M} \in M_n(\mathbb{Q}) \quad I_n \mathbf{M} = \mathbf{M} = \mathbf{M} I_n.$$

Remark: When the value of n is clear from the context, we abbreviate I_n to just I .

Matrix multiplication

is not commutative: I.e. $\mathbf{MN} \neq \mathbf{NM}$ in general.

E.g take $\mathbf{M} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ and $\mathbf{N} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$.

Then $\mathbf{MN} = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$ but $\mathbf{NM} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}$.

is associative: I.e. $\forall n \in \mathbb{N} \ \forall \mathbf{M}, \mathbf{N}, \mathbf{P} \in M_n(\mathbb{Q}) \quad \mathbf{M}(\mathbf{NP}) = (\mathbf{MN})\mathbf{P}$.

This follows from the one-to-one correspondence between matrices and linear functions, the way that matrix multiplication reflects function composition, and the fact that function composition is associative.

has 'zero divisors': I.e. $\mathbf{MN} = \mathbf{O} \Rightarrow (\mathbf{M} = \mathbf{O}) \vee (\mathbf{N} = \mathbf{O})$.

E.g. take $\mathbf{M} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ and $\mathbf{N} = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}$.

Then $\mathbf{MN} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ but $\mathbf{M} \neq \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \neq \mathbf{N}$.

Inverses: motivation

In a certain chemical reaction between two reactants, R1 and R2:

- (a) The quantity of R1 should be twice the quantity of R2.
- (b) The quantity of product is then equal to half the quantity of R1 plus one third of the quantity of R2.

What quantities of R1 and R2 should be used to produce 5 units of the product?

Let x be the quantity of R1, y quantity of R2.

$$\begin{cases} \text{from (a): } x = 2y \\ \text{from (b): } \frac{x}{2} + \frac{y}{3} = 5 \end{cases} \iff \begin{cases} x - 2y = 0 \\ 3x + 2y = 30 \end{cases}$$

We can solve these equations by elimination, but consider the equivalent matrix equation

$$\begin{pmatrix} 1 & -2 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 30 \end{pmatrix}.$$

Q: Can we solve this matrix equation, just using matrices?

Inverses

Question: More generally, can we solve a matrix equation $\mathbf{Ax} = \mathbf{b}$, just using matrices? i.e. Can we “divide” by \mathbf{A} to get $\mathbf{x} = \frac{\mathbf{b}}{\mathbf{A}}$?

Answer: Well, sort of, sometimes.

In some cases there exists a matrix \mathbf{A}^{-1} such that $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

Our chemical reaction example is a case in point:

$$\begin{pmatrix} 1 & -2 \\ 3 & 2 \end{pmatrix}^{-1} = \frac{1}{8} \begin{pmatrix} 2 & 2 \\ -3 & 1 \end{pmatrix} \quad \begin{matrix} \text{(we will see how} \\ \text{to get this later)} \end{matrix}$$

so $\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{8} \begin{pmatrix} 2 & 2 \\ -3 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 30 \end{pmatrix} = \begin{pmatrix} 7.5 \\ 3.75 \end{pmatrix}$.

This matrix \mathbf{A}^{-1} is an ‘inverse’ of \mathbf{A} in the following sense:

An **inverse**, if one exists, of a matrix $\mathbf{A} \in M_n(\mathbb{Q})$ is a matrix $\mathbf{A}^{-1} \in M_n(\mathbb{Q})$ with the property that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{AA}^{-1} = \mathbf{I}_n$.

Note the if an \mathbf{A}^{-1} exists and $\mathbf{Ax} = \mathbf{b}$ then

$$\mathbf{x} = \mathbf{Ix} = (\mathbf{A}^{-1}\mathbf{A})\mathbf{x} = \mathbf{A}^{-1}(\mathbf{Ax}) = \mathbf{A}^{-1}\mathbf{b}.$$

Determinants

Not every square matrix has an inverse.

An obvious example is a zero matrix, but there are also plenty of non-zero examples.

Question: How can we tell (determine) if \mathbf{A} has an inverse?

As we shall see, one answer is that \mathbf{A} has an inverse if and only if its 'determinant' is non-zero.

In this course we only consider the 2×2 case:

The **determinant** of a matrix $\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in M_2(\mathbb{Q})$ is the number

$$\det(A) = ad - bc.$$

Example: $\det \begin{pmatrix} 1 & -2 \\ 3 & 2 \end{pmatrix} = 1 \times 2 - (-2) \times 3 = 8.$

Lemma: For any $\mathbf{A}, \mathbf{B} \in M_2(\mathbb{Q})$, $\det(\mathbf{AB}) = \det(\mathbf{A}) \det(\mathbf{B})$.

Proof: Multiply out both sides.

Calculating Inverses

A matrix \mathbf{A} can have at most one inverse, because if \mathbf{B} and \mathbf{C} are both inverses then $\mathbf{BA} = \mathbf{I}$ and $\mathbf{AC} = \mathbf{I}$ and so

$$\mathbf{B} = \mathbf{BI} = \mathbf{B(AC)} = (\mathbf{BA})\mathbf{C} = \mathbf{IC} = \mathbf{C}.$$

How can we compute (the unique) \mathbf{A}^{-1} when it does exist?

Theorem: A matrix $\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in M_2(\mathbb{Q})$ has an inverse if and only if $\det(\mathbf{A}) \neq 0$ and in this case

$$\boxed{\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}} \quad \text{e.g. } \begin{pmatrix} 1 & -2 \\ 3 & 2 \end{pmatrix}^{-1} = \frac{1}{8} \begin{pmatrix} 2 & 2 \\ -3 & 1 \end{pmatrix}$$

Proof: If \mathbf{A} has an inverse then

$$1 = \det(\mathbf{I}_2) = \det(\mathbf{AA}^{-1}) = \det(\mathbf{A}) \det(\mathbf{A}^{-1}),$$

so $\det(\mathbf{A})$ cannot be zero. But if $\det(\mathbf{A}) \neq 0$ then multiplying out shows that $\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$ when \mathbf{A}^{-1} is given by the formula.

What about $n > 2$? See Math1013 or Math1115.

Back to population dynamics

As a final example involving matrix multiplication and matrix inverses, we return to the simple ecosystem model

$$\forall n \in \mathbb{N} \quad \begin{cases} x_{n+1} = 4x_n - y_n, \\ y_{n+1} = y_n + 2x_n, \end{cases}$$

where x_n, y_n are the populations of two species after n time steps.
We can rewrite this in the form

$$\forall n \in \mathbb{N} \quad \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix}.$$

This is an implicit definition of a sequence of vectors. We will use mathematical induction to establish an explicit formula, stated and proved on the next slide. First two preliminary results

R1: $\begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$ [prove by multiplying out the RHS]

R2: $\begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}^{-1} = \frac{1}{2-1} \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$ [formula for inverse of 2×2 matrix]

$$\text{Claim: } \forall n \in \mathbb{N}^* \quad \begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 3^n & 0 \\ 0 & 2^n \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

Proof:

Basis step: When $n = 0$ the RHS becomes (using R2)

$$\begin{aligned} \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} &= \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \text{LHS.} \end{aligned}$$

Inductive step: Assume the explicit formula holds up to and including some particular n , and consider the case $n + 1$. Then, using the implicit definition, preliminary results R1 and R2, and the inductive assumption,

$$\begin{aligned} \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} &= \begin{pmatrix} 4 & -1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 3^n & 0 \\ 0 & 2^n \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 3^{n+1} & 0 \\ 0 & 2^{n+1} \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \end{aligned}$$

and hence the formula also holds for $n + 1$.

END OF SECTION B3

C1. Counting.

Notes originally prepared by Judy-anne Osborn.
Editing, expansion and additions by Malcolm Brooks.
Substantial tinkering by Adam Piggott.

Text Reference (Epp) 3ed: Sections 6.1-7, 7.3
 4ed: Sections 9.1-7
 5ed: Sections 9.1-7

Principles of counting

Bijections preserve cardinality If A and B are finite sets and there exists a bijection $f : A \rightarrow B$, then $|A| = |B|$.

TO USE THIS PRINCIPLE: Count something easier, and exhibit a bijection between the set you which to count and the set you have counted.

The Pigeonhole Principle If $k + 1$ or more pigeons occupy k pigeonholes, then at least one pigeonhole must contain two or more pigeons.

The Generalised Pigeonhole Principle If N objects are classified in k disjoint categories, then at least one category must contain $\lceil \frac{N}{k} \rceil$ objects. ($\lceil \frac{N}{k} \rceil$ means the least integer that is greater than or equal to $\frac{N}{k}$)

Permutations There are $n!$ ways to arrange n distinct objects in a list.

Principles of counting

r-Permutations There are

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

ways to select and order r out of n distinct objects.

Combinations There are

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

ways to choose a set of r objects from a set of n objects (that is, to select r out of n distinct objects when the order in which objects are selected is not important). The notation $\binom{n}{r}$ is read “ n choose r .”

Principles of counting

Multisets (Stars and Bars) There are $\binom{r+n-1}{r}$ size- r multisets with members from a set of size n . That is, there are $\binom{r+n-1}{r}$ ways to arrange a list of r stars and $n-1$ bars.

Inclusion-Exclusion If A and B are finite sets, then

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

The Sum Rule If A is a finite set and $\{A_1, A_2, \dots, A_m\}$ is a partition of A , then $|A| = |A_1| + |A_2| + \dots + |A_m|$.

The Product Rule If A_1, A_2, \dots, A_m are finite sets, then

$$|A_1 \times A_2 \times \dots \times A_m| = |A_1| \times |A_2| \times \dots \times |A_m|.$$

A mixed example

How many distinguishable ways can the letters of the word
MILLIMICRON
be arranged?

If we were to distinguish between like letters using labels, as in

M₁I₁L₁L₂M₂I₃C R O N

there would be $11! = 39\,916\,800$ different arrangements.

Now we must compensate for the over-counting induced by this distinguishing between indistinguishable arrangements.

Since MILLIMICRON has 2 M's, 3 I's and 2 L's the true answer is :

$$\frac{11!}{2! 3! 2!} = 1\,663\,200.$$

This example generalises both permutations and combinations.
Can you see how?

'Stars and Bars'

Let s and b be positive integers. How many different strings of length $s + b$ can we form out of s stars (*) and b bars (|)?

For example, here are three different strings of length 13 formed out of 10 stars and 3 bars:

	*	*	*	*		*	*		*	*	*	*
*		*	*	*	*	*	*			*	*	*
*	*	*		*	*		*	*		*	*	*

ANSWER: Consider a horizontal arrangement of cells numbered 1 through $s + b$.

1	2	3	...	$s + b$
			...	

Making a string of s stars and b bars is like choosing a set of b numbers from the set $\{1, 2, \dots, s + b\}$, where the set of chosen numbers is the set of positions containing bars. Hence we can form $\binom{s+b}{b}$ different strings of length $s + b$ from s stars and b bars.

'Stars and Bars' example

(Epp(4ed) Q9.6.15)

For how many integers from 1 through 99 999 is the sum of their digits equal to 10?

Proof: By inserting leading zeros if necessary, all the integers to be counted can be considered to be 5-digit strings $abcde$ with $a+b+c+d+e=10$.

Each of these 5-digit strings can be represented as a length-14 pattern of 10 **stars** and 4 **bars**. For example:

$\star\star|\star\star\star\star|*\mid\star\star|*$ represents 24 121.

$\star\star\star||\star\star\star\star|\star\star\star|$ represents 30 430.

$|||\star\star\star\star\star\star\star\star\star\star|*$ represents 00 091.

There are $\binom{14}{4}$ ways to arrange a list of 10-stars and 4-bars. But five of the patterns have ten stars in a row and so don't count (ten is not a digit). So the number of integers is

$$\binom{14}{4} - 5 = \frac{14 \times 13 \times 12 \times 11}{4 \times 3 \times 2 \times 1} - 5 = 996. \square$$

Counting ‘Multisets’

What is a ‘multiset’?

It’s a ‘set’ with multiple copies of elements allowed and acknowledged.

An example is $\{c, b, a, c, a\}$, which has 2 a ’s, 1 b and 2 c ’s.

As for ordinary sets, order is irrelevant: $\{c, b, a, c, a\} = \{a, a, b, c, c\}$.

But the multiplicities **do** matter.

Formally, a **size- r multiset** is a set S together with a ‘multiplicity function’ $m : S \rightarrow \mathbb{N}$, where,

$$\forall s \in S \quad m(s) = \text{number of copies of } s \quad \text{and} \quad r = \sum_{s \in S} m(s).$$

So, for example, $\{c, b, a, c, a\}$ has size $r = 2 + 1 + 2 = 5$.

Counting multisets

How many different size- r multisets can be formed from members of a set S of cardinality n ?

IDEA: A size r -multiset formed from members of a set S of cardinality n can be represented by a pattern of r stars and $n - 1$ bars.

For example if $S = \{a, b, c, d\}$ and $r = 5$ then $\{c, b, a, c, a\}$ is represented by $\star\star|*\star|\star\star$ ($m(a)=2, m(b)=1, m(c)=2, m(d)=0$).

Which multisets, selected from S , are represented by the following arrangements?

$$\begin{array}{c} ||\star\star\star|\star\star \\ \{c, c, c, d, d\} \end{array}$$

$$\begin{array}{c} |\star\star\star|\star\star| \\ \{b, b, b, c, c\} \end{array}$$

$$\begin{array}{c} \star\star|\star|\star|\star \\ \{a, a, b, c, d\} \end{array}$$

There are $\binom{r+n-1}{r}$ size- r multisets with members from a set of size n .

Multisets example

(Epp(4ed) Q9.6.6)

If n is a positive integer, how many 5-tuples of integers from 1 through n can be formed in which the elements of the 5-tuple are written in non-increasing order?

For $n = 9$ some 5-tuples are $(8, 6, 4, 2, 1)$, $(9, 3, 3, 2, 2)$, $(6, 6, 6, 6, 6)$.

There is a bijection (one-to-one correspondence) between the set of all these 5-tuples and the set of all size-5 multisets chosen from $\{1, \dots, n\}$, because the r 'members' of the multiset can only be arranged in one way in non-increasing order.

So by stars-and-bars, there are $\binom{5+n-1}{5} = \binom{n+4}{5}$ of these 5-tuples.

For example for $n = 3$ there are $\binom{7}{5} = \binom{7}{2} = 21$ such 5-tuples:

33333 33332 33331 33322 33321 33311 33222 33221 33211 33111

32222 32221 32211 32111 31111 22222 22221 22211 22111 21111 11111

New counts from old

- The Sum Rule
- The Product Rule
- Inclusion-Exclusion

The Sum Rule

If sets A and B are finite and *disjoint* then the cardinality of their union $A \cup B$ is the sum of the their individual cardinalities, i.e.

$$A \cap B = \emptyset \implies |A \cup B| = |A| + |B|.$$

More generally, if $\{A_1, A_2, \dots, A_m\}$, $m \in \mathbb{N}$, is a *partition* of the finite set A then

$$|A| = |A_1| + |A_2| + \dots + |A_m|.$$

Example:

For $U = \{-10, \dots, 10\} \subseteq \mathbb{Z}$ and $S = \{n \in U : |20 - n^2| > 10\}$, find $|S|$.

Observe that $|20 - n^2| > 10 \iff n^2 < 10 \vee n^2 > 30$.

So $S = \{-3, -2, \dots, 2, 3\} \cup \{6, 7, \dots, 10\} \cup \{-10, -9, \dots, -6\}$.

Hence $|S| = 7 + 5 + 5 = 17$.

The Product Rule

For finite sets A and B the cardinality of their cartesian product $A \times B$ is the product of the their individual cardinalities, i.e.

$$|A \times B| = |A| \times |B|.$$

More generally, for finite sets A_1, A_2, \dots, A_m , $m \in \mathbb{N}$,

$$|A_1 \times A_2 \times \cdots \times A_m| = |A_1| \times |A_2| \times \cdots \times |A_m|.$$

Example:

A regular **cubic die, D6**, has face set $C = \{1, 2, 3, 4, 5, 6\}$.

An **octahedral die, D8**,

has face set $O = \{1, 2, \dots, 8\}$.

A **dodecahedral die, D12**,

has face set $D = \{1, 2, \dots, 12\}$.



The number of possible outcomes from throwing the three dice together is $|C \times O \times D| = |C| \times |O| \times |D| = 6 \times 8 \times 12 = 576$.

The product rule via an outcome construction procedure

Often the product rule is implemented by designing a procedure by which we construct one of the objects to be counted, and then counting the number of different possible outcomes from each step of the procedure. When you do this you must be careful that each outcome can be constructed in only one way.

Suppose that a web-banking password is always 8 characters long and it always comprises two upper case letters, one digit, and 5 lower case characters. How many different passwords can be created that follow these rules?

Counting passwords

A: We construct a password in 11 steps.

In step 1, we choose the two positions in which the upper case letters will be placed. There are $\binom{8}{2}$ ways to make this choice.

In step 2, we choose one integers from the remaining 6 positions. This are $\binom{6}{1}$ ways to make this choice.

In step 3, we choose an upper case letter to go in the first place for an upper case letter. There are 26 ways to make this choice.

In step 4, we choose an upper case letter to go in the second place for an upper case letter. There are 26 ways to make this choice.

In step 5, we choose a digit to go in the digit place. There are 10 ways to make this choice.

In step 6, we choose a lower case letter to go in the first place for a lower case letter. There are 26 ways to make this choice.

Counting passwords (cont.)

In step 7, we choose a lower case letter to go in the first place for a lower case letter. There are 26 ways to make this choice.

In step 8, we choose a lower case letter to go in the first place for a lower case letter. There are 26 ways to make this choice.

In step 9, we choose a lower case letter to go in the first place for a lower case letter. There are 26 ways to make this choice.

In step 10, we choose a lower case letter to go in the first place for a lower case letter. There are 26 ways to make this choice.

In step 11, we choose a lower case letter to go in the first place for a lower case letter. There are 26 ways to make this choice.

By the product rule, we can create

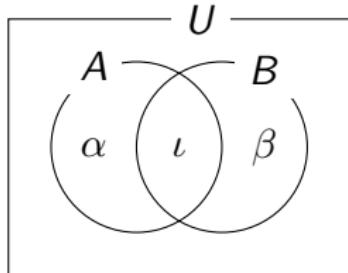
$$\binom{8}{2} \times \binom{6}{1} \times 26 \times 26 \times 10 \times 26^6$$

different passwords. \square

Inclusion-Exclusion

If A and B are finite sets which *may not be disjoint* the sum rule has to be modified to the **inclusion-exclusion rule**:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

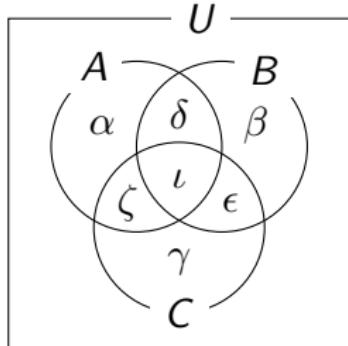


This is because the plain sum rule $|A \cup B| = |A| + |B|$ includes the intersection $A \cap B$ twice, so it has to be excluded once:

$$|A \cup B| = \alpha + \iota + \beta = (\alpha + \iota) + (\iota + \beta) - \iota = |A| + |B| - |A \cap B|.$$

The inclusion-exclusion rule can be generalised to deal with more than two sets, but it quickly gets very messy.

Can you figure out how to extend the rule to deal with just three sets A , B and C ?



Bit-String Example of Inclusion-Exclusion

How many bytes start with '1' or end with '00'?

IDEA: Count the bytes that start with 1, then count the bytes that end in 00, then count the bytes that start with 1 and end with 0, then apply Inclusion-Exclusion.

Task 1: Construct a byte that starts with '1'.

- There is one way to choose the first bit (1)
- There are two ways to choose the second bit (0 or 1)
- There are two ways to choose the third bit (0 or 1)
- :
- There are two ways to choose the eighth bit (0 or 1)

Product Rule: Task 1 can be done in $1 \times 2^7 = 128$ ways.

Bit-String Example of Inclusion-Exclusion

Task 2

Task 2: Construct a byte that ends with '00'.

- There are two ways to choose the first bit (0 or 1)
- There are two ways to choose the second bit (0 or 1)
⋮
- There are two ways to choose the sixth bit (0 or 1)
- There is one way to choose the seventh bit (0)
- There is one way to choose the eighth bit (0)

Product Rule: Task 2 can be done in $2^6 \times 1^2 = 64$ ways.

Bit-String Example of Inclusion-Exclusion

Task 3

Is the answer $128+64 = 196$? **NO!**

That would be overcounting.

We have to subtract off the cases we counted twice.

Task 3: Construct a string of length 8 that both starts with '1' and ends with '00'.

- There is one way to choose the first bit (1)
- There are two ways to choose the second bit (0 or 1)
⋮
- There are two ways to choose the sixth bit (0 or 1)
- There is one way to choose the seventh bit (0)
- There is one way to choose the eighth bit (0)

Product Rule: Task 3 can be done in $1 \times 2^5 \times 1^2 = 32$ ways.

Bit-String Example of Inclusion-Exclusion

Conclusion

Finally, the number of ways to construct a bit string of length 8 that starts with '1' or ends with '00' is equal to:

- the number of ways to do task 1, **plus**
- the number of ways to do task 2, **minus**
- the number of ways to do both at the same time (task 3), *i.e.*

$$128 + 64 - 32 = 160.$$

Note: An alternative, and quite different, way to solve this problem is to use **complementary counting**; *i.e.* calculate $|S^c|$ where S is the set of strings we are interested in and U is the set of all (8-bit) strings. Then $|S| = |U| - |S^c| = 2^8 - 1 \times 2^5 \times 3 = 256 - 96 = 160$.

Can you see how to get the $1 \times 2^5 \times 3$?

Combinations and binomial coefficients

There are $C(n, r) = \frac{P(n, r)}{r!} = \frac{n!}{r!(n-r)!}$ ways to choose a set of r objects from a set of n candidates.

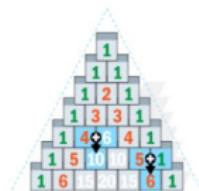
I.e. A set of cardinality n has $\frac{n!}{r!(n-r)!}$ subsets of cardinality r .

The subsets are called **r -combinations**.

We say ' n choose r ' for $C(n, r)$ and often write it $\binom{n}{r}$.

These numbers $\binom{n}{r}$ arise as coefficients in the algebraic expansion of the n -th power of the 'binomial' $(x + y)$ and are consequently also known as **binomial coefficients**. The expansion is

$$(x + y)^n = \sum_{r=0}^n \binom{n}{r} x^{n-r} y^r.$$



Two important properties of $\binom{n}{r}$

$$1. \forall r, n \in \mathbb{N}^* \quad 0 \leq r \leq n \implies$$

$$\binom{n}{r} = \binom{n}{n-r}. \quad \text{e.g. } \binom{5}{3} = \binom{5}{2}.$$

Proof: Choosing the r elements of a subset S of U , with $|U| = n$, is exactly equivalent to choosing the $n-r$ elements of U to be left out.

$$2. \forall r, n \in \mathbb{N}^* \quad 0 < r \leq n \implies$$

$$\text{e.g. } \binom{5}{3} = \binom{4}{3} + \binom{4}{2}.$$

$$\binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}.$$

(Pascal's Triangle Identity)

Proof: Let u be a fixed member of U , with $|U| = n$.

Subsets S of U with $|S| = r$ are of two types; those that don't contain u and those that do.

There are $\binom{n-1}{r}$ of the first kind, since the r members of S are chosen from the $n-1$ members of $U \setminus u$.

There are $\binom{n-1}{r-1}$ of the second kind, since the $r-1$ members of $S \setminus u$ are also chosen from the $n-1$ members of $U \setminus u$.

C1. Counting.

Notes originally prepared by Judy-anne Osborn.
Editing, expansion and additions by Malcolm Brooks.
Substantial tinkering by Adam Piggott.

Text Reference (Epp) 3ed: Sections 6.1-7, 7.3
 4ed: Sections 9.1-7
 5ed: Sections 9.1-7

Cardinality

This section is mostly about calculating the number of objects of some specified type; for example counting all five digit numbers with no repeated digits. Counting like this can be viewed as finding the number of members of some set, also known as finding the *size* of the set.

For a finite set A , this 'size' or 'cardinality' is just the number of members of A . However it can be defined formally as follows:

Let A be a set. Suppose there exists a bijection (one-to-one correspondence) from A to a subset of the natural numbers of the form $\{1, 2, \dots, n\}$ for some $n \in \mathbb{N}$. Then the **cardinality**, or **size** of the set A , written $|A|$, is n . Thus $|A| = n$.

Earlier in the course we saw that this definition is suitable for generalisation to infinite sets (not all infinite sets have the same cardinality.) It also points to some practical counting techniques (see next slide).

Example: numbers in an interval

- What is the cardinality of the set of natural numbers in an interval?
- Example: $S = \{150, 151, 152, \dots, 160\}$
- Subtract 149 from each:

$$\begin{array}{cccccc} 150 & 151 & 152 & \dots & 160 \\ \downarrow & \downarrow & \downarrow & \dots & \downarrow \\ 1 & 2 & 3 & \dots & 11 \end{array}$$

- We have made a bijection to the set $\{1, 2, 3, \dots, 11\}$, so $|S| = 11$.

Example: numbers in an interval, generalized

- Let $S = \{a, a + 1, a + 2, \dots, b\} \subseteq \mathbb{N}$
- A nice bijection subtracts ' $a - 1$ ' from each element of S . We have

$$\begin{array}{ccccccc} a & a+1 & a+2 & \cdots & & b \\ \downarrow & \downarrow & \downarrow & \cdots & & \downarrow \\ 1 & 2 & 3 & \cdots & & b-a+1 \end{array}$$

- Therefore $|S| = b - a + 1$.

A slightly harder but similar example

How many numbers from 150 to 330 inclusive are congruent to 5 mod 7?

$150 \bmod 7 = 3$ so the lowest number is $150 + 2 = 152$.

$330 \bmod 7 = 1$ so the highest number is $330 - 3 = 327$.

Now use the following composition of bijections:

	152	159	166	...	327
subtract 5:	↓	↓	↓	...	↓
	147	154	161	...	322
divide by 7:	↓	↓	↓	...	↓
	21	22	23	...	46
subtract 20:	↓	↓	↓	...	↓
	1	2	3	...	26

Since the composition of bijections is a bijection, the answer is 26.

Finite and infinite sets

The **cardinality of the empty set** is defined to be 0. Thus $|\emptyset| = 0$.

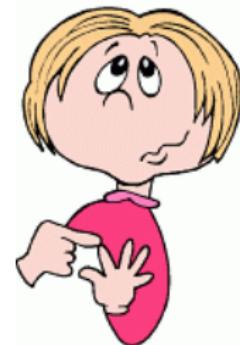
Any set S with cardinality $|S| = n \in \mathbb{N}^*$ it is said to be **finite**.

Any set S that is not finite is said to be **infinite**. We write $|S| = \infty$.

Examples:

Finite Sets	Infinite Sets
$\{1, 2, 3\}$	\mathbb{N} ... natural numbers
$\{\text{red, orange, yellow, green, blue, purple}\}$	\mathbb{Z} ... integers
$\{b: b \text{ is a book in the Hancock library}\}$	\mathbb{Q} ... rational numbers
$\{s: s \text{ is a star in the Milky Way Galaxy}\}$	\mathbb{R} ... real numbers
$\{\}$	$\mathcal{P}(\mathbb{R})$... power set of \mathbb{R}

Countability



A set S is called **countable** when there is a bijection from S to a subset of the set \mathbb{N} of natural numbers.

Examples:

- Any *finite* set is countable.
- The *empty set* is countable (because $\emptyset \subseteq \mathbb{N}$).
- The set \mathbb{P} of all primes is countable (because $\mathbb{P} \subseteq \mathbb{N}$).
- \mathbb{N} itself is countable (because $\mathbb{N} \subseteq \mathbb{N}$).
- The sets \mathbb{N} and \mathbb{P} are each both **countable** and **infinite**.
Such sets are called **countably infinite**.

Comparing cardinalities

Generalising from the case of finite sets, we say that two sets A and B have **the same cardinality**, written $|A| = |B|$, provided that there exists a bijection (one-to-one correspondence) from A to B .

Remember that for $\phi : A \rightarrow B$ to be a bijection:

- no two arrows point to the same element of B , i.e.

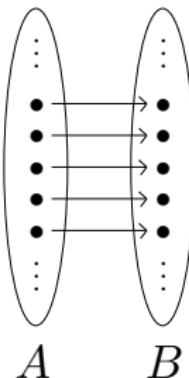
$$\forall x, y \in A \quad x \neq y \implies \phi(x) \neq \phi(y).$$

- each element of B must have an arrow pointing to it: i.e.

$$\forall b \in B \quad \exists a \in A \quad \phi(a) = b.$$

Also remember that $\phi : A \rightarrow B$ is a bijection if and only if it has an inverse $\phi^{-1} : B \rightarrow A$.

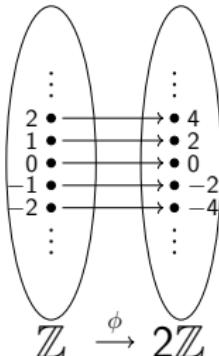
Since the inverse is also a bijection, we say that A and B have the same cardinality if and only if there is a bijection **between** them. (i.e. we don't have to specify the direction of the isomorphism).



Examples of sets with the same cardinality

1 You might expect the set \mathbb{Z} of integers to have a 'bigger' cardinality than the set $2\mathbb{Z} = \{2z : z \in \mathbb{Z}\}$ of even integers, since there are 'twice as many' integers as even integers.

But actually the two sets have the **same** cardinality, because the function $\phi : \mathbb{Z} \rightarrow 2\mathbb{Z} : \phi(z) = 2z$ is (clearly) a bijection.

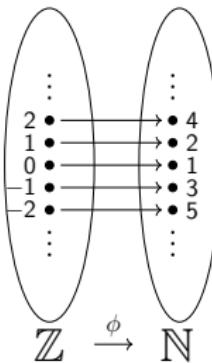


2 By tweaking the function ϕ a little we can establish the perhaps more surprising fact that \mathbb{Z} also has the same cardinality as \mathbb{N} .

The modified version of ϕ is

$$\phi : \mathbb{Z} \rightarrow \mathbb{N} : \phi(z) = \begin{cases} 2z & \text{if } z > 0 \\ 1 - 2z & \text{if } z \leq 0 \end{cases}$$

It's not difficult to see that this is a bijection.



All countably infinite sets have the same cardinality

Recall that we call a set S countably infinite if (and only if) it is infinite and there is a bijection from S to a subset of \mathbb{N} .

In other words, S has the same cardinality as some infinite subset of \mathbb{N} . However

Any infinite subset S of \mathbb{N} has the same cardinality as \mathbb{N} itself.

Proof: Since every bijection has an inverse, it is enough to establish a bijection $\phi : \mathbb{N} \rightarrow S$. This can be done recursively as follows:

$$\phi(1) = \text{least member of } S$$

$$\phi(2) = \text{least member of } S \setminus \{\phi(1)\} \quad (2^{\text{nd}} \text{ least member})$$

$$\forall n \in \mathbb{N} \quad \phi(n+1) = \text{least member of } S \setminus \{\phi(1), \dots, \phi(n)\}.$$

Note: It follows from the result above (and its proof) that proving that an arbitrary infinite set S (not necessarily a subset of \mathbb{N}) is countably infinite amounts to showing that it can be '**well-ordered**'. This means that it is possible to order the elements of S in some (perhaps ingenious) way so that S and every subset of S has a 'least' member.

Not all infinite sets have the same cardinality

Earlier in the course, we saw that \mathbb{Q} is countable. This was shown by describing a bijection from \mathbb{N} to \mathbb{Q} . Since every bijection has an inverse that is a bijection, this suffices to show that there exists a bijection from \mathbb{Q} to \mathbb{N} .

In lectures I stated that \mathbb{R} , the set of real numbers, is uncountable. I uploaded to ECHO360 a video in which I gave an argument showing that there is no surjection (and hence no bijection) from \mathbb{N} to \mathbb{R} . It follows that there is no bijection from \mathbb{R} to \mathbb{N} .

So we have

$$|\mathbb{N}| = |\mathbb{N}^*| = |\mathbb{Z}| = |\mathbb{Q}|$$

but

$$|\mathbb{N}| \neq |\mathbb{R}|.$$

These observations, first made by Georg Cantor (1845-1918), were a breakthrough in mathematical thinking about infinite sets.

How do we count?

How do we count? We have a collection of counting principles. When we need to count some objects, we analyse those objects until we carefully match the situation to one of the situations in which a counting principle applies.

What makes counting hard? Matching your scenario to one of the scenarios described in a counting principle takes care, as the scenarios described in the counting principles sound similar unless you read carefully.

What is your best strategy? Understand the scenarios described in the counting principles. In particular, carefully notice how the scenarios are different. Then, when you have to count something and you think that a counting principle applies, **state which counting principle you are using and explain carefully (write it out) how you know that the situation you have is like the scenario described in the counting principle.**

The structures we count

We have already discussed sets, n -tuples, lists and sequences. In a set, order is unimportant and for elements in a set there is no notion of “how many times” the element is in the set. Lists, n -tuples and sequences are different ways to talk about the same thing—order matters and elements can appear more than once unless explicitly excluded by the language. We need one more structure, one that recognises multiplicity (how many times an element appears) but not order (there is no first element, second element etc)....

Multisets

A multiset is a ‘set’ with multiple copies of elements allowed and acknowledged. An example is $\{c, b, a, c, a\}$, which has 2 a’s, 1 b and 2 c’s.

As for ordinary sets, order is irrelevant: $\{c, b, a, c, a\} = \{a, a, b, c, c\}$.

But the multiplicities **do** matter.

Formally, a **size- r multiset** is a set S together with a ‘multiplicity function’ $m : S \rightarrow \mathbb{N}$, where,

$$\forall s \in S \quad m(s) = \text{number of copies of } s \quad \text{and} \quad r = \sum_{s \in S} m(s).$$

So, for example, $\{c, b, a, c, a\}$ has size $r = 2 + 1 + 2 = 5$.

Principles of counting

Bijections preserve cardinality If A and B are finite sets and there exists a bijection $f : A \rightarrow B$, then $|A| = |B|$.

TO USE THIS PRINCIPLE: Count something easier, and exhibit a bijection between the set you which to count and the set you have counted.

The Pigeonhole Principle If $k + 1$ or more pigeons occupy k pigeonholes, then at least one pigeonhole must contain two or more pigeons.

The Generalised Pigeonhole Principle If N objects are classified in k disjoint categories, then at least one category must contain $\lceil \frac{N}{k} \rceil$ objects. ($\lceil \frac{N}{k} \rceil$ means the least integer that is greater than or equal to $\frac{N}{k}$)

Permutations There are $n!$ ways to arrange n distinct objects in a list.

Principles of counting

r-Permutations There are

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

ways to select and order r out of n distinct objects.

Combinations There are

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

ways to choose a set of r objects from a set of n objects (that is, to select r out of n distinct objects when the order in which objects are selected is not important). The notation $\binom{n}{r}$ is read “ n choose r .”

Principles of counting

Multisets (Stars and Bars) There are $\binom{r+n-1}{r}$ size- r multisets with members from a set of size n . That is, there are $\binom{r+n-1}{r}$ ways to arrange a list of r stars and $n-1$ bars.

Inclusion-Exclusion If A and B are finite sets, then

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

The Sum Rule If A is a finite set and $\{A_1, A_2, \dots, A_m\}$ is a partition of A , then $|A| = |A_1| + |A_2| + \dots + |A_m|$.

The Product Rule If A_1, A_2, \dots, A_m are finite sets, then

$$|A_1 \times A_2 \times \dots \times A_m| = |A_1| \times |A_2| \times \dots \times |A_m|.$$

Example 1

Prove the following: If A is a non-empty finite set, then $|\mathcal{P}(A)| = 2^{|A|}$.

IDEA: A subset of A corresponds to making one of two choices for each element of A : include it in the subset or not. These choices could be recorded as a bit-string. So elements of $\mathcal{P}(A)$ can be counted by counting bit-strings...

Proof: Let A be a non-empty finite set. Let $n = |A|$. Let B be the set of n -bit strings. When representing integers, the smallest element of B represents 0 and the largest represents $2^n - 1$; it follows that $|B| = 2^n$. Since **bijections preserve cardinality**, to prove the result, it is enough to exhibit a bijection from B to $\mathcal{P}(A)$. Let $f : A \rightarrow \{1, 2, \dots, n\}$ be a bijection. Let $g : B \rightarrow \mathcal{P}(A)$ be the function defined by the rule $b_1 b_2 \dots b_n \mapsto \{a \in A \mid b_{f(a)} = 1\}$. We leave the reader to verify that g is a bijection. \square

An example illustrating the functions in Example 1

Let $A = \{\text{cat, dog, chicken}\}$. Then $n = 3$ and

$$B = \{000, 001, 010, 011, 100, 101, 110, 111\}.$$

Let $f : A \rightarrow \{1, 2, 3\}$ be the function defined by

$$f(\text{cat}) = 1, f(\text{dog}) = 2, f(\text{chicken}) = 3.$$

Then

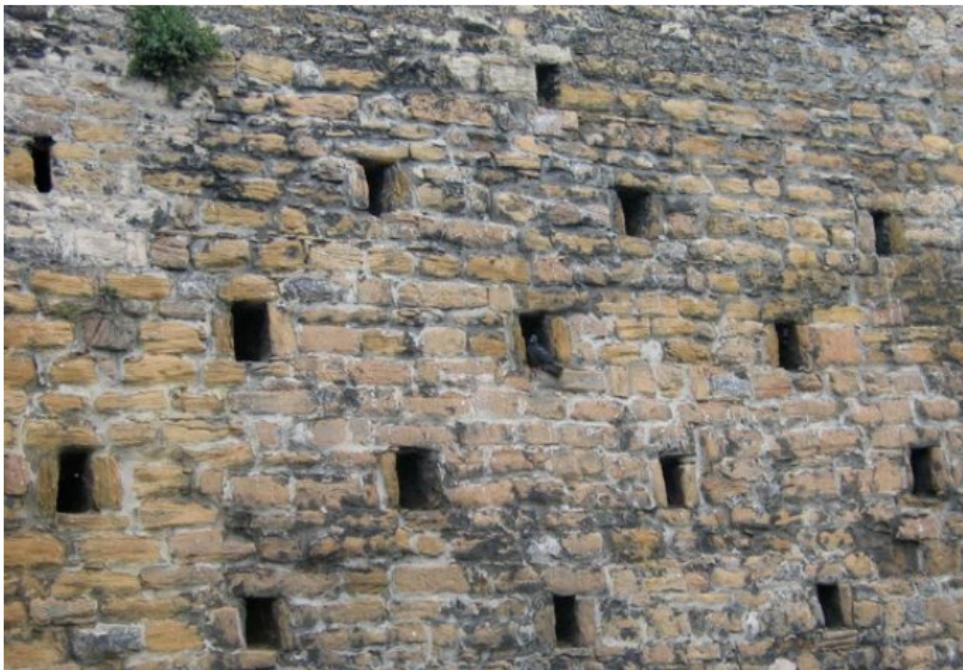
$$g(011) = \{\text{dog, chicken}\}$$

$$g(101) = \{\text{cat, chicken}\}$$

$$g(000) = \emptyset$$

$$g(100) = \{\text{cat}\}$$

The Pigeonhole Principle



Example 2

Prove the following: If there are 11 players in a soccer team that wins 12 – 0, there must be at least one player in the team who scored more than once (assuming no own-goals).

Proof: We shall apply the pigeonhole principle. Imagine a score sheet in which each player's name is written on a different row. When a goal is scored, a stroke in the appropriate row records who scored it. The rows on the scoresheet are like pigeonholes, and the goals are like pigeons. Since there are 12 goals (12 pigeons) to be recorded, and only 11 rows (11 pigeonholes) on the scoresheet, at least one row must contain the record of two goals. So at least one player scored more than once. □

Example 3

Prove the following: If a molecule can exist in 2 different configurations, and you have 10^9 such molecules, at least half of them must be in the same configuration.

Proof: We shall apply the generalised pigeonhole principle. We have 10^9 objects (the molecules) to be classified into 2 disjoint categories (the configurations). By the generalised pigeonhole principle, at least one category contains $\lceil 10^9/2 \rceil$ objects (molecules). Note that $\lceil 10^9/2 \rceil \geq 10^9/2$; that is, $\lceil 10^9/2 \rceil$ is at least half of 10^9 . Hence at least one configuration is taken by at least $10^9/2$ molecules. □

Example 4

Show that, in any set of a thousand words, there must be at least 39 words that start with the same letter.

Proof: We shall apply the generalised pigeonhole principle. Each word can be classified by its first letter, and such categories are disjoint. Since there are 1000 words and 26 categories, at least one category (configuration) contains $\lceil 1000/26 \rceil = \lceil 38.46 \rceil = 39$ words. □

Example 5 (Epp(4ed) Q9.4.33)

Let A be a set of six distinct positive integers each of which is less than 15. Show that there must be two distinct nonempty proper subsets of A whose elements when added up give the same sum.

IDEAS: The phrase '*there must be two distinct subsets*' in the second sentence of the question suggests that the subsets should play the role of the pigeons.

Then the phrase '*give the same sum*' suggests that the possible sums should play the role of the pigeon holes. So

$$A = \{a, b, c, d, e, f \in \mathbb{N} : a < b < c < d < e < f < 15\}$$

Pigeons: subsets of A

Pigeon holes: possible element sums of subsets of A

Now we have to count the pigeons and pigeon holes.

Example 5 (cont)

Proof: First we show that there are 62 distinct proper nonempty subsets of A (pigeons). Since $|A| = 6$, we have that $|\mathcal{P}(A)| = 2^6 = 64$. Since \emptyset is empty, and A is not proper, there are 62 distinct proper nonempty subsets of A .

Now we show that there are only 60 different possible element sums among these nonempty proper subsets. A proper subset of A contains at most 5 elements. The elements of A are selected from $1, 2, \dots, 14$. The most a sum of at most 5 elements from these integers can be is $10 + 11 + 12 + 13 + 14 = 60$. The least the sum can be is 1. Hence the element sum of a proper nonempty subset of A is between 1 and 60. (Thus there are 60 pigeonholes).

By the pigeonhole principle, at least one element sum is taken by two distinct nonempty proper subsets of A . \square

Example 6 (cont.)

Given a set of 52 distinct integers, show that there must be two whose sum or difference is divisible by 100.

Proof: We label the 52 distinct integers a_1, a_2, \dots, a_{52} . For each i such that $1 \leq i \leq 52$, let $A_i = a_i \bmod 100$. We consider two cases.

Case: There exist i, j such that $i < j$ and $A_i = A_j$.

In this case,

$$A_i = A_j \Leftrightarrow a_i \equiv a_j \pmod{100} \Leftrightarrow (a_i - a_j) \text{ is divisible by } 100.$$

In this case, there are two integers whose difference is divisible by 100 and we are done.

Example 6 (cont.)

Case: There does not exist i, j such that $i < j$ and $A_i = A_j$.

In this case, the integers A_1, A_2, \dots, A_{52} are distinct. We shall apply the pigeonhole principle, with the integers A_1, A_2, \dots, A_{52} playing the role of pigeons. We label 51 'pigeonholes' as shown, so that each is labelled by two not necessarily distinct two-digit numbers:

00	01	02	03	49	50
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		<input type="text"/>	<input type="text"/>
00	99	98	97		51	50

Observe that, for each box, the pair of integers labelling the box have a sum that is divisible by 100.

There are now exactly 51 'pigeonholes', and 52 numbers in the list A_1, \dots, A_{52} . So at least two of the numbers in the list A_1, \dots, A_{52} label the same pigeonhole. This means that at least two of the numbers in the list A_1, \dots, A_{52} have a sum that is divisible by 100.

□

Example 7

Andy, Beth and Cai are standing in line.

In how many different orders could this queue be arranged?

Proof: Arranging Andy, Beth and Cai in a line is like arranging 3 distinct objects in a list. Thus there are $3! = 3 \times 2 \times 1 = 6$ ways to do this. □

Checking our answer (unnecessary, but instructive): We can represent an “outcome” by listing three letters in order: the first letter represents the first person in the line; the second letter represents the second person in the line; and the third letter represents the third person in the line. The entire set of outcomes, containing 6 elements, is represented below:

$$\{ABC, ACB, BAC, BCA, CAB, CBA\}$$

Example 8

A pet show awards 1st, 2nd and 3rd prizes. There are 5 entrants:

Rachel the Rabbit Charles the Chicken Tilly the Terrier
Bob the Bilby Karen the Kangaroo

In how many ways can the prizes be handed out?

Proof: Handing out the prizes is like selecting and ordering 3 out of 5 distinct objects. Thus there are

$$P(5, 3) = \frac{5!}{(5 - 3)!} = \frac{5!}{2!} = \frac{5 \times 4 \times 3 \times 2 \times 1}{2 \times 1} = 5 \times 4 \times 3 = 60$$

ways to hand out the prizes. □

Checking our answer to Example 8

We can represent an “outcome” by listing three letters in order: the first letter represents the pet awarded first prize; the second letter represents the pet awarded second prize; and the third letter represents the pet awarded third prize. The entire set of outcomes, containing 60 elements, is represented below:

```
{ BCK, BCR, BCT, BKC, BKR, BKT,  
  BRC, BRK, BRT, BTB, BTC, BTR,  
  CBK, CBR, CBT, CKB, CKR, CKT,  
  CRB, CRK, CRT, CTB, CTK, CTR,  
  KBC, KBR, KBT, KCB, KCR, KCT,  
  KRB, KRC, KRT, KTB, KTC, KTR,  
  RBC, RBK, RBT, RCB, RCK, RCT,  
  RKB, RKC, RKT, RTB, RTC, RTK,  
  TBC, TBK, TBR, TCB, TCK, TCR,  
  TKB, TKC, TKR, TRB, TRC, TRK, }
```

Example 9

A different pet show does not give 1st, 2nd and 3rd prizes.

Instead, three “ribbon-winning” pets are chosen to travel to the district show. There are 5 entrants:

Rachel the Rabbit Charles the Chicken Tilly the Terrier
Bob the Bilby Karen the Kangaroo

In how many ways can the ribbon-winners be chosen?

Proof: Selecting the ribbon-winning pets is like choosing a set of 3 objects (ribbon-winning pets) from a set of 5 objects (the pets in the show). Thus there are

$$C(5, 3) = \binom{5}{3} = \frac{P(5, 3)}{3!} = \frac{5!}{3!(5 - 3)!} = \frac{120}{6 \times 2} = 10$$

ways to choose the ribbon-winners. □

Checking our answer to Example 9

Each “outcome” may be represented by a set containing the ribbon-winning pets. The ten sets representing the ten different outcomes are listed below:

- | | |
|---------------------------|--------------------------|
| {Rachel, Charles, Tilly}, | {Rachel, Charles, Bob}, |
| {Rachel, Charles, Karen}, | {Rachel, Tilly, Bob}, |
| {Rachel, Tilly, Karen}, | {Rachel, Bob, Karen}, |
| {Charles, Tilly, Bob}, | {Charles, Tilly, Karen}, |
| {Charles, Bob, Karen}, | {Tilly, Bob, Karen} |

C2. Probability

Notes originally prepared by Judy-anne Osborn and Pierre Portal.
Editing, expansion and additions by Malcolm Brooks.

Text Reference (Epp) 3ed: Sections 6.7-9

4ed: Sections 9.7-9

5ed: Sections 9.7-9

(Only the last part of §9 on ‘independence’ is relevant for this course)

A thought experiment:



- Toss a coin.
- What are the possible outcomes?
- ‘Heads’ or ‘Tails’
- What is the probability of ‘Heads’?
- We say it is

$$\mathbb{P}(\text{Heads}) = \frac{1}{2}. \quad \textit{Why?}$$

Methods of assigning probabilities

Method 1: Use relative frequencies (empirical experiment)

Method 2: Use a model (combination of prior knowledge, guessing, deduction)

- Eg. assume equally likely outcomes

Relative Frequencies

- For example, one may carry out the experiment of tossing a coin ten times and noting the results.
- My experiment gave

T,T,H,H,T,T,T,H,T,H

- Slightly fewer than half the coin-tosses resulted in 'H' (for 'Heads').
- A 'longer run' may give different (better?) results.
- There is much more to be said on 'relative frequencies', but for this course we will focus on making 'models'.

A model for coin tossing

- Observe a real coin: it has two sides – ‘Heads’ and ‘Tails’.
- Actually there is a third ‘side’: the rim.
- Because the rim is small we **decide to ignore** this possibility.
- We **simplify** on purpose, to make the model *tractable*.
- The ‘Heads’ and ‘Tails’ sides are so similar physically that we make an assumption:

equal likelihood

for

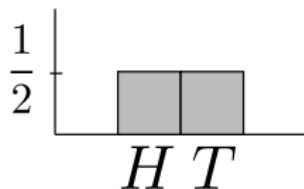
Heads or Tails.

A model for coin tossing: equal likelihood

The two possibilities are just as likely as each other.

$$\mathbb{P}(\text{Heads}) = \frac{1}{2} \quad \mathbb{P}(\text{Tails}) = \frac{1}{2}$$

We can represent this situation graphically as



Fix some terminology

An **Experiment** observes a phenomenon that has one or more possible **outcomes**.

The **Sample space** of an experiment is the set of possible outcomes of the experiment.

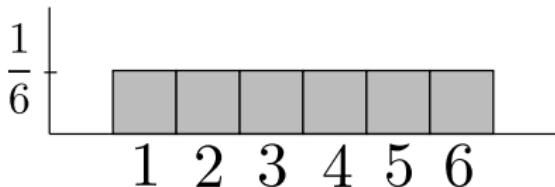
An **Event** is any *subset* of the sample space.

The probability of an event E is denoted by $\mathbb{P}(E)$.

Example:

- **Experiment:** single toss of a standard die, noting upper face's number.
- One possible **outcome** would be '3'
- **Sample space:** $\{1, 2, 3, 4, 5, 6\}$

An equal likelihood model for die-tossing



What's an event? Any subset of the sample space.

Eg. an event is

$$\{3, 6\}$$

= the set of numbers divisible by 3 in sample space {1, 2, 3, 4, 5, 6}.

$$\mathbb{P}(\{3, 6\}) = \frac{|\{3, 6\}|}{|\{1, 2, 3, 4, 5, 6\}|} = \frac{2}{6} = \frac{1}{3}$$

Equal Likelihood for finite Sample Spaces

Generalising from the previous example we have:

- Let S be a finite sample space in which all outcomes are equally likely.
- Let E be an event in S .
- Then the probability of the event E is

$$\mathbb{P}(E) = \frac{|E|}{|S|}$$

where $|E|$ is the number of outcomes in E , and
 $|S|$ is the number of outcomes in S .

Properties of Probability

For any finite sample space:

- $\mathbb{P}(E)$ is a real number between 0 and 1, i.e.

$$0 \leq \mathbb{P}(E) \leq 1.$$

- Probability of the complement is one minus the probability of the event, i.e.

$$\mathbb{P}(\text{not } E) = 1 - \mathbb{P}(E).$$

- The sum of the probabilities of all outcomes in the sample space is 1.
- ' $\mathbb{P}(E) = 1$ ' implies E is certain to occur.*
- ' $\mathbb{P}(E) = 0$ ' implies E is impossible.*

*For infinite sets, this isn't necessarily true. 'Measure theory' explains why.

Previous example of tossing a die:

Probability of event of ‘getting a number exactly divisible by 3’ is one third, which satisfies

$$0 \leq \frac{1}{3} \leq 1.$$

Probability of ‘not E ’ is

$$\mathbb{P}(\text{number not divisible by 3}) = 1 - \frac{1}{3} = \frac{2}{3}.$$

The sum of the probabilities of all outcomes is

$$\mathbb{P}(\{1\}) + \dots + \mathbb{P}(\{6\}) = \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} + \frac{1}{6} = 1$$

The Sum and Product Rules for Probability

The Sum Rule

Sum Rule: If events E_1, \dots, E_n are mutually disjoint, i.e.
 $E_i \cap E_j = \emptyset$ for all $i \neq j$, then

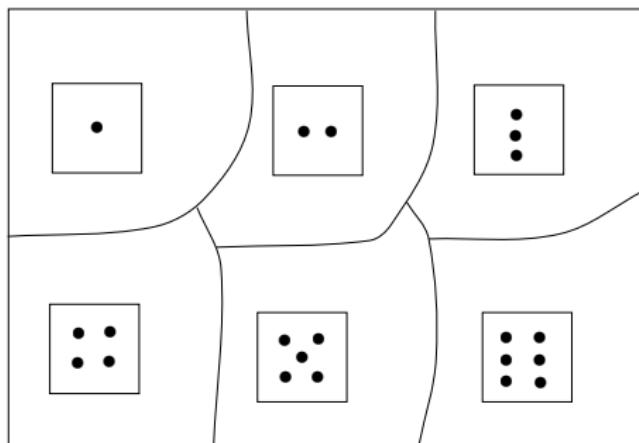
$$\mathbb{P}(E_1 \cup \dots \cup E_n) = \mathbb{P}(E_1) + \dots + \mathbb{P}(E_n).$$

Disjoint events exclude one another in the sense that they cannot happen at the same time.

Sum Rule for probability: another die-tossing example

What is the probability that the outcome from a single toss of a die is an odd number?

The six possible outcomes are all disjoint (cannot occur simultaneously).



Thus the sum rule applies.

- We assign equal probabilities to each of these disjoint events (Why?)
- Six possible outcomes in total \rightarrow each has probability $\frac{1}{6}$ of occurring.
- The probability that the die lands with an odd number up is

$$\Pr(\boxed{\cdot}) + \Pr(\boxed{:}) + \Pr(\boxed{::})$$

$$= \frac{1}{6} + \frac{1}{6} + \frac{1}{6}$$

$$= \frac{1}{2}$$

by the sum rule.

Sum Rule for probability. Example: Non-zero numbers

Let $R_n = \{-n, \dots, -2, -1, 0, 1, 2, \dots, n\}$.

What is the probability that a number chosen at random from R_n is non-zero?

We assume that we are equally likely to choose any element of R_n .

- The probability that the number is negative is $\frac{|R_n^-|}{|R_n|} = \frac{n}{2n+1}$.
- The probability that the number is positive is $\frac{|R_n^+|}{|R_n|} = \frac{n}{2n+1}$.

Therefore the probability of a number chosen at random from the set $\{-n, \dots, -2, -1, 0, 1, 2, \dots, n\}$ being non-zero is:

$$\begin{aligned} & \mathbb{P}(\text{the number is negative}) + \mathbb{P}(\text{the number is positive}) \\ &= \frac{n}{2n+1} + \frac{n}{2n+1} = \frac{2n}{2n+1}. \end{aligned}$$

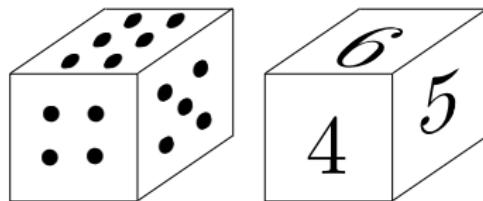
The Product Rule

- **Product Rule:** If events E_1, \dots, E_n are ‘independent’ of each other; then the probability of composite event ‘ E_1 and E_2 and ... and E_n ’ is

$$\mathbb{P}(E_1 \times E_2 \times \dots \times E_n) = \mathbb{P}(E_1) \times \mathbb{P}(E_2) \times \dots \mathbb{P}(E_n).$$

- To see what we mean by ‘independent’, consider a procedure that can be broken down into successive tasks, each of which could be done in a number of ways. If the choice of the way to do any one task had no influence on the choice of ways to do any other of the tasks, then the tasks would be independent.
- A formal definition of independence will be given later.

Product Rule probability example: Tossing two dice



- What is the probability that the outcome from tossing a pair of dice is '4' for the first die and '5' for the second die i.e.



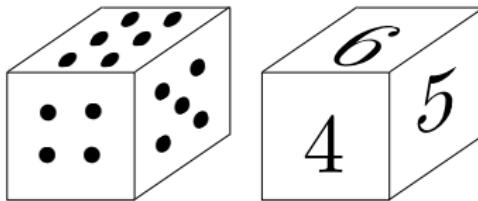
- We assume that the outcomes for each die are **independent**, i.e that they don't influence one another at all.
- Hence the product rule applies.

$$\begin{aligned} & \Pr \left(\begin{array}{cc} \boxed{\vdots \vdots} & \boxed{5} \end{array} \right) \\ = & \Pr \left(\boxed{\vdots \vdots} \right) \times \Pr \left(\boxed{5} \right) \\ = & \frac{1}{6} \times \frac{1}{6} \\ = & \frac{1}{36} \end{aligned}$$

by the Product Rule.

An example of the Sum and Product Rules used together

- Often we combine use of the Sum and Product rules in one problem.
- For example, *what is the probability of getting an odd total when tossing a pair of dice?*



- To obtain an odd total, either
 - the first die must give odd and the second die even; or
 - the first die must give even and the second die odd.
- These two possibilities are **disjoint**, so the sum rule applies:
 $\mathbb{P}(\text{odd total}) = \mathbb{P}(\text{1st odd, 2nd even}) + \mathbb{P}(\text{1st even, 2nd odd})$

- But now consider $\mathbb{P}(\text{1st odd, 2nd even})$. The events “1st odd” and “2nd even” are **independent** of each other; they don’t affect each other.
- Hence the **product rule** applies to this part of the problem:

$$\begin{aligned}\mathbb{P}(\text{1st odd, 2nd even}) &= \mathbb{P}(\text{1st odd}) \times \mathbb{P}(\text{2nd even}) \\ &= \frac{3}{6} \times \frac{3}{6} = \frac{1}{2} \times \frac{1}{2} = \frac{1}{4}\end{aligned}$$

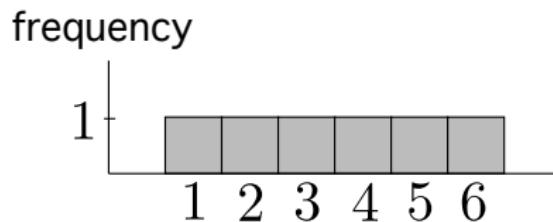
- Similarly, $\mathbb{P}(\text{1st even, 2nd odd}) = \frac{1}{4}$
- Putting it all together,

$$\begin{aligned}\mathbb{P}(\text{odd total}) &= \mathbb{P}(\text{1st odd, 2nd even}) + \mathbb{P}(\text{1st even, 2nd odd}) \\ &= \frac{1}{4} + \frac{1}{4} = \frac{1}{2}.\end{aligned}$$

Density and Distribution

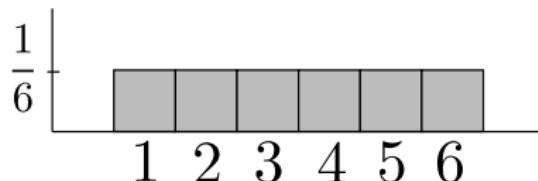
Frequency Histograms

- One way to visualize all possible outcomes of an experiment together is to draw a frequency histogram.
- We have already seen some simple examples, like tossing a die with equally likely possible outcomes: 1, 2, 3, 4, 5, 6:



Probability Density Functions

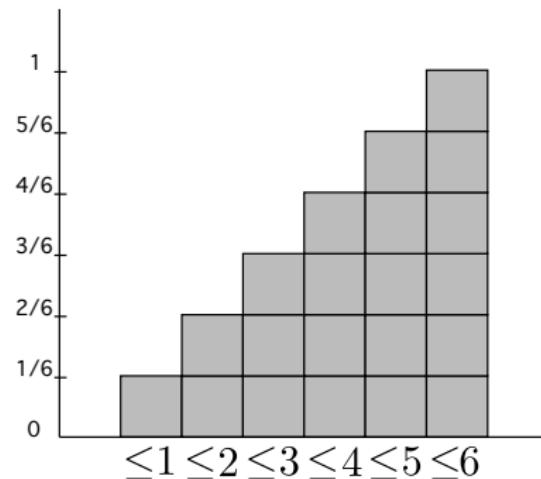
- The **Probability Density Function** (or just **Density**) is obtained from a Frequency Histogram by **normalizing**. We divide the vertical axis by the total number of outcomes.
- Continuing the die-tossing example, we have



What is the area under the curve? Why?

Cumulative Probability Distribution Functions

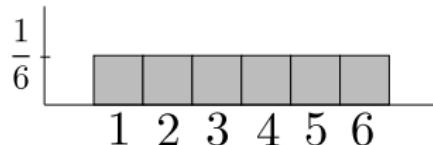
- The Cumulative Probability Distribution Function (or Distribution) is obtained from the Density Function by graphing cumulative totals.
- Continuing the die-tossing example, we have



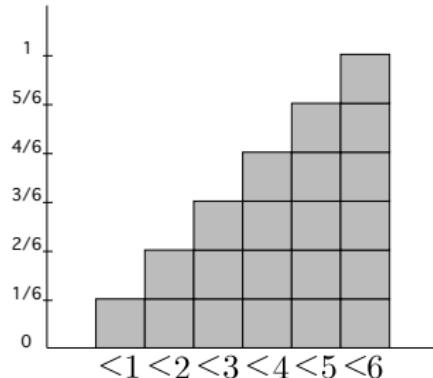
- We will only use of cumulative distributions when looking up probability values in tables or online.

Uniform Distribution

- When every event has the same probability the resulting densities and distributions are called '**uniform**'. Examples:
- Uniform density:**



- Uniform distribution:**



Tossing two coins:

- Some more interesting densities and distributions are obtained by considering events which combine several outcomes.
- For example, tossing two coins. A neat way to list all possible outcomes is to expand

$$\begin{aligned} & (T + H)(T + H) \\ &= TT + TH + HT + HH \end{aligned}$$

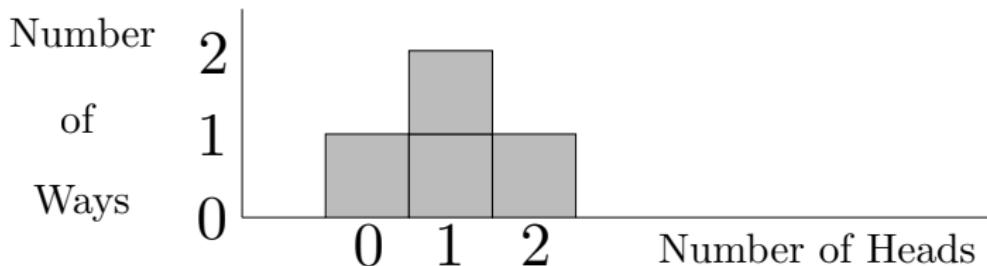
- *What is the sample space?*

$$\boxed{\{TT, TH, HT, HH\}}$$

Tossing two coins:

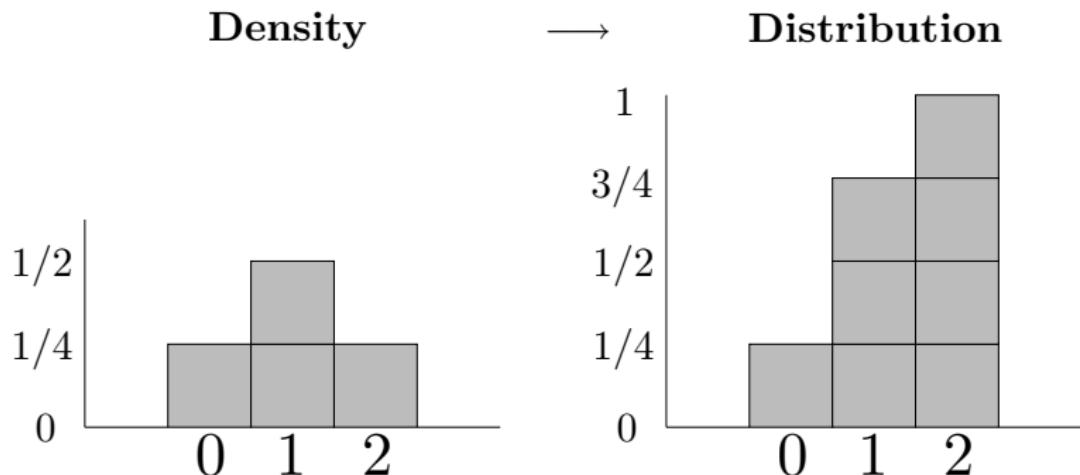
- Now consider events:
 - E_0 : 'No heads'
 - E_1 : 'exactly 1 Head'
 - E_2 : 'exactly 2 Heads'
- Which subsets of the sample space correspond to these events?*
$$E_0 = \{TT\}, \quad E_1 = \{TH, HT\}, \quad E_2 = \{HH\}$$

Frequency Histogram:



Two fair coins: Density and Distribution Functions

- Assuming a fair coin (equally likely outcomes), divide out by the size of the sample space to get density function. Then take the cumulative sum of the density to get the distribution:



Class Example - Three Fair Coins

Q: For tossing three fair coins, what is the sample space?

*A: All possible outcomes are given by $(T + H)(T + H)(T + H)$
 $= TTT + TTH + THT + THH + HTT + HTH + HHT + HHH.$
 S.Space = {TTT, TTH, THT, THH, HTT, HTH, HHT, HHH}.*

*Q: What subsets of the sample space correspond to the events
 E_0 : '0 heads', E_1 : 'exactly 1 Head', E_2 : 'exactly 2 Heads' and
 E_3 : 'exactly 3 Heads' ?*

*A: $E_0 = \{TTT\}$; $E_1 = \{TTH, THT, HTT\}$;
 $E_2 = \{THH, HTH, HHT\}$; $E_3 = \{HHH\}$.*

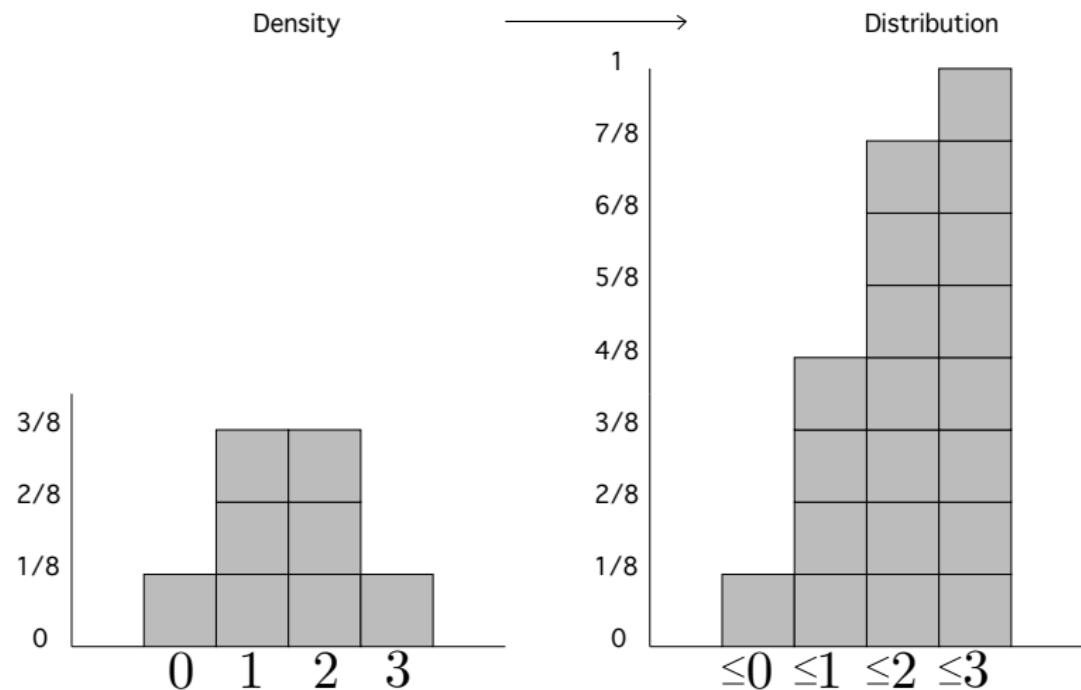
Q: What is the size of the sample space? A: 8

Q: What do we divide frequencies by (on the Frequency Histogram) to get the Density Function? A: 8

Q: What are the probabilities of the four events E_0, E_1, E_2, E_3 ?

A: $\mathbb{P}(E_0) = \frac{1}{8}$; $\mathbb{P}(E_1) = \frac{3}{8}$; $\mathbb{P}(E_2) = \frac{3}{8}$; $\mathbb{P}(E_3) = \frac{1}{8}$.

Three Fair Coins: Density and Distribution Functions



Binomial Probability Distributions

The family of functions that come from coin-tossing are all examples of **binomial** densities/distributions:

Density



Distribution

$n = 2 :$



$n = 3 :$

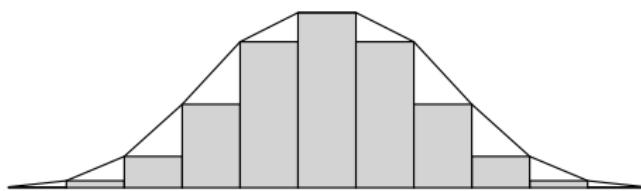


$n = 4 :$



Bell-like curves for large n

As n gets larger and larger these **binomial probability density functions** get closer and closer to the famous Bell Curve:



which is the so-called '**Normal**' Probability Density Function.

Pascal's Triangle
and
Coin Tossing

Pascal's Triangle

$$\begin{array}{ccccccc} & & & & 1 & & \\ & & & & 1 & 1 & \\ & & & & 1 & 2 & 1 \\ & & & & \swarrow + \searrow & & \\ & & & 1 & 3 & 3 & 1 \\ & & & \swarrow + \searrow & & & \\ 1 & 4 & 6 & 4 & 1 & & \end{array}$$

Pascal's Triangle

- Frequencies in Coin-Tossing are numbers in Pascal's Triangle

$$\begin{array}{ccccccc} & & & & 1 & & \\ & & & & 1 & 1 & \\ & & & & 1 & \swarrow + \searrow & 2 & 1 \\ & & & & 1 & 3 & \swarrow + \searrow & 3 & 1 \\ & & & & 1 & 4 & 6 & 4 & 1 \end{array}$$

- Each row is generated by expanding a binomial, eg:

$$(y + x)^4 = y^4 + 4y^3x + 6y^2x^2 + 4yx^3 + x^4.$$

Pascal's Triangle

- We've seen these numbers before in 'combinations': $\binom{n}{k}$:

$$\begin{array}{ccccccc} & & \binom{0}{0} & & & & \\ & & \binom{1}{0} & & \binom{1}{1} & & \\ & & \binom{2}{0} & \textcolor{blue}{+} & \binom{2}{1} & & \binom{2}{2} \\ & & \binom{3}{0} & \binom{3}{1} & \textcolor{blue}{+} & \binom{3}{2} & \binom{3}{3} \\ & & \binom{4}{0} & \binom{4}{1} & \binom{4}{2} & \binom{4}{3} & \binom{4}{4} \end{array}$$

The Binomial Theorem

- The Binomial Theorem states that

$$(y + x)^n = \binom{n}{0} y^n x^0 + \binom{n}{1} y^{n-1} x^1 + \cdots + \binom{n}{n} y^0 x^n$$

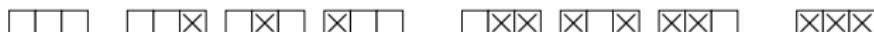
and gives the rows of Pascal's Triangle in its coefficients.

Idea of Proof of Binomial Theorem:

$$(y + x)(y + x)(y + x)$$

$$= yyy + yyx + yxy + yxx + xyy + xyx + xxy + xxx$$

$$= \underbrace{yyy}_{\binom{3}{0} \text{ } x's} + \underbrace{yyx + yxy + xyy}_{\binom{3}{1} \text{ } x's} + \underbrace{yxx + xyx + xxy}_{\binom{3}{2} \text{ } x's} + \underbrace{xxx}_{\binom{3}{3} \text{ } x's}$$



Tossing n Coins

- Let's toss a (fair) coin n times ($n \in \mathbb{N}$.)
- As earlier, let E_k denote the event of obtaining k heads.
- Then
 - E_0 can occur in $\binom{n}{0} = 1$ way, i.e $|E_0| = \binom{n}{0}$
 - E_1 can occur in $\binom{n}{1} = n$ ways, i.e $|E_1| = \binom{n}{1}$
 - E_2 can occur in $\binom{n}{2}$ ways, i.e $|E_2| = \binom{n}{2}$
 - \vdots
 - E_n can occur in $\binom{n}{n}$ ways, i.e $|E_n| = \binom{n}{n}$

What is the total size of the sample space?

- I.e. what is

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \dots + \binom{n}{n}?$$

- The binomial theorem gives a neat way to find the sum.
- Set $x = y = 1$ then

$$\binom{n}{0} 1^n 1^0 + \binom{n}{1} 1^{n-1} 1^1 + \cdots + \binom{n}{n} 1^0 1^n = (1+1)^n$$

so that

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n} = 2^n.$$

- We can also get 2^n by observing that there are two possible outcomes for each toss, and so $2 \times 2 \times \cdots \times 2 = 2^n$ possible outcomes for n tosses.
- So, for example, the probability of obtaining exactly three heads from six tosses of a fair coin is

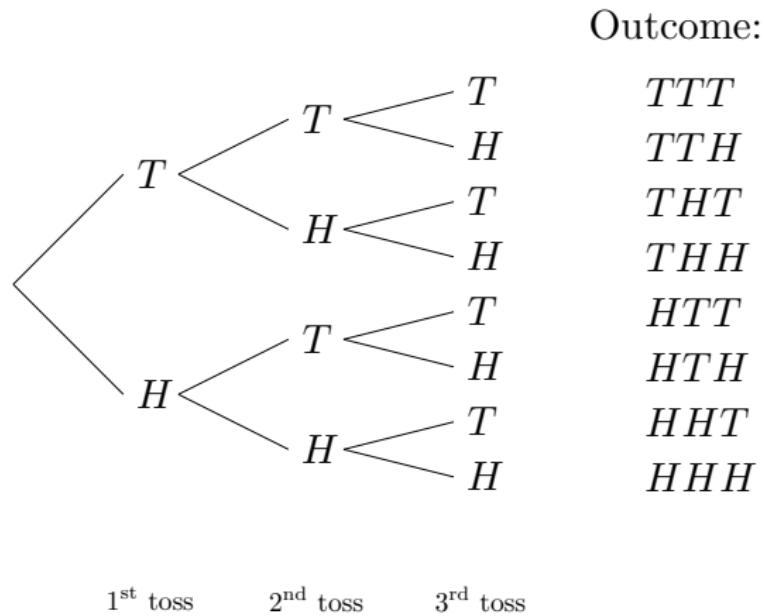
$$\frac{\binom{6}{3}}{2^6} = \frac{\frac{6 \cdot 5 \cdot 4}{3 \cdot 2 \cdot 1}}{64} = \frac{20}{64} = \frac{5}{16}.$$

- Probabilities like these can be looked up in tables rather than calculated. Examples will be found in worksheet and assignment questions.

Tree Diagrams, Fair and Unfair Coins, and the General Binomial Distribution

A tree representation of Coin-tossing

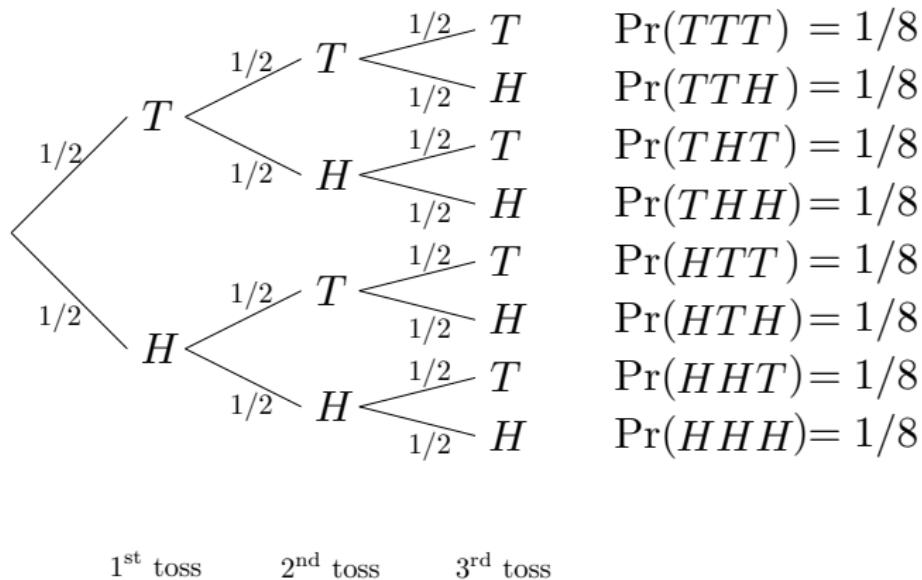
- Another way to list all the outcomes of an event is to draw a Tree Diagram of the Possibilities



Three tosses of a fair coin

- This allows us to deal with fair coins, as before:

Outcome:

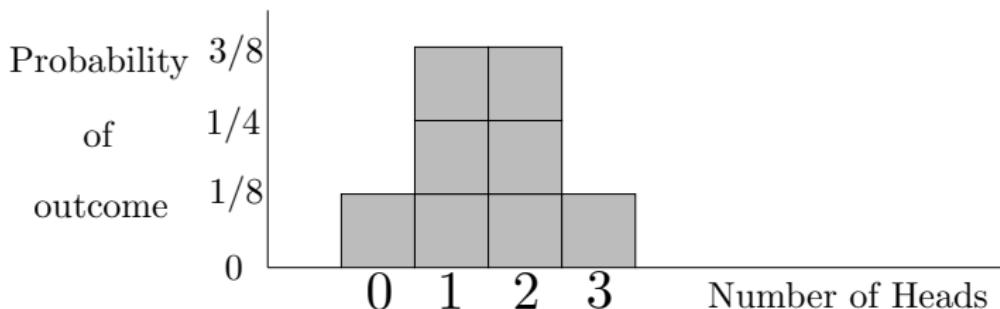


Three tosses of a fair coin

Collecting possibilities from the tree and using the sum rule gives

$$\boxed{\mathbb{P}(0\text{heads}) = \frac{1}{8}, \quad \mathbb{P}(1\text{head}) = \frac{3}{8}, \quad \mathbb{P}(2\text{heads}) = \frac{3}{8}, \quad \mathbb{P}(3\text{heads}) = \frac{1}{8}}$$

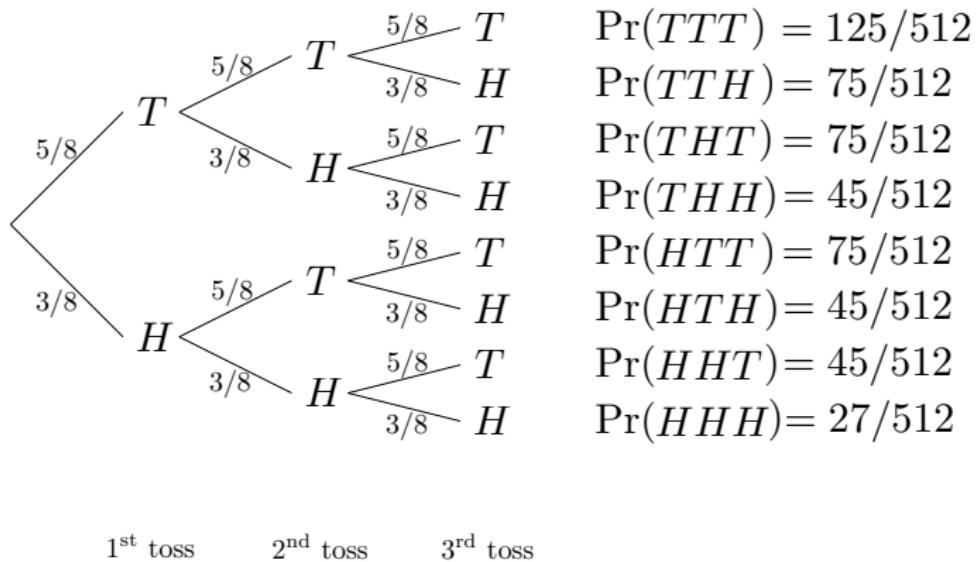
– the same density function as before; $n = 3$ and $p = \frac{1}{2}$ (fair coin):



Three tosses of an unfair coin

- But we can also deal with an **unfair coin** – not equal likelihood:

Outcome:

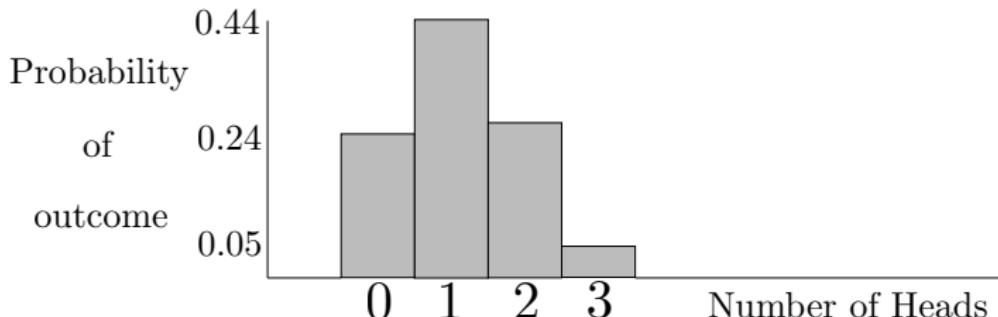


Three tosses of an unfair coin

Collecting possibilities from the tree and using the sum rule gives

$$\mathbb{P}(0\text{heads}) = \frac{125}{512}, \mathbb{P}(1\text{head}) = \frac{225}{512}, \mathbb{P}(2\text{heads}) = \frac{135}{512}, \mathbb{P}(3\text{heads}) = \frac{27}{512}$$

The unfair coin with $n = 3$ tosses and probability $p = 3/8$ of heads on a single toss, gives a non-symmetric binomial density function:



The general binomial density function for n trials (e.g. tosses) with probability p of a success (e.g. head) on each trial is given by

$$\mathbb{P}(k \text{ successes}) = \binom{n}{k} p^k (1-p)^{n-k}.$$

Review of
Probability Density Functions
with
More Challenging Examples

Probability for equally likely outcomes (Review)

For a finite non-empty set S and $E \subseteq S$, the **probability of E for equally likely outcomes** is the number

$$\boxed{\mathbb{P}(E) = \frac{|E|}{|S|}}$$

Vocabulary: S is called the **sample space**. E is called an **event**.
 An element $s \in S$ is called an **outcome**.

Examples:

- (1) Model for a (fair) coin toss. $S = \{H, T\}$.

$\{H\}$ is the event 'coin shows a Head'. $\mathbb{P}(\{H\}) = \frac{|\{H\}|}{|\{H, T\}|} = \frac{1}{2}$.

The probability for equally likely outcomes is such that:

$$\boxed{\mathbb{P}(\emptyset) = 0, \mathbb{P}(\{H\}) = \frac{1}{2}, \mathbb{P}(\{T\}) = \frac{1}{2}, \mathbb{P}(\{H, T\}) = 1.}$$

- (2) Model for a (balanced) die roll. $S = \{1, 2, 3, 4, 5, 6\}$.

5 is an outcome. $\{3, 5\}$ is an event. $\mathbb{P}(\{3, 5\}) = \frac{|\{3, 5\}|}{|S|} = \frac{2}{6} = \frac{1}{3}$.

General probabilities on finite sets

Let $\mathbb{Q}_+ = \{q \in \mathbb{Q} ; q \geq 0\}$, (the set of non-negative rational numbers).

A (general) **probability density function** on a finite set (sample space) S is any function

$$\mathbb{P} : S \rightarrow \mathbb{Q}_+ \text{ with } \sum_{s \in S} \mathbb{P}(s) = 1.$$

We call $\mathbb{P}(s)$ the **probability of s** , so the probabilities sum to 1.

For any subset (event) $E \subseteq S$ the **probability of E** , $\mathbb{P}(E)$ is

$$\mathbb{P}(E) = \sum_{s \in E} \mathbb{P}(s).$$

Example:

For $S = \{s_1, \dots, s_n\}$ define $\mathbb{P} : S \rightarrow \mathbb{Q}_+$ by $\mathbb{P}(s_j) = \frac{1}{n}, j = 1, \dots, n$.

Then $\mathbb{P}(\{\omega_1, \dots, \omega_m\}) = \sum_{j=1}^m \mathbb{P}(\omega_j) = \sum_{j=1}^m \frac{1}{n} = \frac{m}{n} = \frac{|\{\omega_1, \dots, \omega_n\}|}{|S|}$.

So \mathbb{P} is the probability of equally likely outcomes.

Another general probability example

In a group of 10 students, 5 are studying computer science, 2 are studying art history, and 3 are studying mathematics. We pick a student from this group and ask what her/his major is.

The sample space is $S = \{M, A, C\}$, where M stand for maths, A for art history, and C for computer science.

The probability density function $\mathbb{P} : S \rightarrow \mathbb{Q}_+$ is given by

$$\mathbb{P}(M) = \frac{3}{10}, \quad \mathbb{P}(A) = \frac{2}{10}, \quad \mathbb{P}(C) = \frac{5}{10}.$$

The associated event probabilities are

$$\begin{aligned}\mathbb{P}(\emptyset) &= 0, & \mathbb{P}(\{M\}) &= \frac{3}{10}, & \mathbb{P}(\{A\}) &= \frac{2}{10}, & \mathbb{P}(\{C\}) &= \frac{5}{10}, \\ \mathbb{P}(\{M, A\}) &= \frac{5}{10}, & \mathbb{P}(\{A, C\}) &= \frac{7}{10}, & \mathbb{P}(\{M, C\}) &= \frac{8}{10}, & \mathbb{P}(S) &= 1.\end{aligned}$$

The Monty Hall problem

A game: Three doors, with a prize behind one of them.

The contestant chooses one door.

The host, who knows where the prize is, opens one of the other two doors, revealing that the prize is not there.

The contestant can then change her/his choice, or not.

Should she/he change doors?

Before the host opens the door:

Sample space is $S = \{d_1, d_2, d_3\}$ with $\mathbb{P}(d_1) = \mathbb{P}(d_2) = \mathbb{P}(d_3) = \frac{1}{3}$.
The contestant chooses d_1 and has a $\frac{1}{3}$ chance of winning.

The host then opens door 2:

The sample space is now $S' = \{d_1, d_3\}$, but $\mathbb{P}(d_1)$ remains at $\frac{1}{3}$ (since the prize hasn't moved) so $\mathbb{P}(d_3) = \frac{2}{3}$ since the sum of the probabilities must be 1.

Thus the prize is twice as likely to be behind door 3 as behind door 1.

So the contestant should change doors.

Properties of probabilities

Theorem: Let S be a finite set and $\mathbb{P} : \mathcal{P}(S) \rightarrow \mathbb{Q}_+$ a probability density function. Then:

- (i) $\forall E \in \mathcal{P}(S) \quad 0 \leq \mathbb{P}(E) \leq 1.$
- (ii) For $E, F \in \mathcal{P}(S)$ with $E \cap F = \emptyset$, $\mathbb{P}(E \cup F) = \mathbb{P}(E) + \mathbb{P}(F).$
- (iii) For any $E, F \in \mathcal{P}(S)$, $\mathbb{P}(E \cup F) = \mathbb{P}(E) + \mathbb{P}(F) - \mathbb{P}(E \cap F).$
- (iv) $\mathbb{P}(E^c) = 1 - \mathbb{P}(E) \quad \forall E \in \mathcal{P}(S).$

The birthday problem

In a group of 50 people, what is the probability that two people have the same birthday (assuming all birthdays are equally likely)?

Let $D = \{1, \dots, 365\}$ represent the days of the year.

Then the sample space is $S = D^{50} = \{(a_n)_{n=1,\dots,50} \subseteq D\}$, representing all sequences of 50 birthdays.

We have $|S| = |D^{50}| = |D|^{50} = 365^{50}$.

The event E of ‘two persons have the same birthday’ is $E = \{(a_n)_{n=1,\dots,50} \subseteq D ; \exists j, k \in \{1, \dots, 50\} a_j = a_k\}$.

The complementary event ‘everybody has a different birthday’ is $E^c = \{(a_n)_{n=1,\dots,50} \subseteq D ; \forall j, k \in \{1, \dots, 50\} j \neq k \implies a_j \neq a_k\}$.

$$\mathbb{P}(E^c) = \frac{|E^c|}{|S|} = \frac{P(365, 50)}{365^{50}} = \frac{365!}{365^{50} \times 315!} \sim 0.03.$$

Thus

$$\mathbb{P}(E) = 1 - \mathbb{P}(E^c) \sim 0.97.$$

There is a 97% chance that two people will have the same birthday.

Random Variables, Expected Values and Independence

Random variables and Expected values

A (simple) **random variable** on a sample space S is any function $X : S \rightarrow \mathbb{Q}$. (More generally, $S \rightarrow \mathbb{Q}^m$ but we will stick to $m = 1$.)

Note: We will denote the event ‘the random variable X is equal to a ’ by just $\{X = a\}$ instead of the more formal $\{s \in S ; X(s) = a\}$.

Example:

$S = \{\text{H,T}\}^3$ = set of outcomes of tossing three coins.

$X((a, b, c))$ = number of H's amongst a, b, c .

$\{X = 2\} = \{\text{HHT,HTH,THH}\}$.

Relative to a probability density function $\mathbb{P} : S \rightarrow \mathbb{Q}_+$ the **expected value** $\mathbb{E}(X)$ of a random variable X is defined by

$$\mathbb{E}(X) = \sum_{s \in S} \mathbb{P}(s)X(s) = \sum_{a \in \text{Range}(X)} \mathbb{P}(\{X = a\})a$$

Example(cont.):

$$\mathbb{E}(X) = \left(\frac{1}{8}\right)0 + \left(\frac{3}{8}\right)1 + \left(\frac{3}{8}\right)2 + \left(\frac{1}{8}\right)3 = \frac{12}{8} = 1.5.$$

Thus the expected value of X is just the mean (average) number of heads obtained when three coins are tossed.

Die roll example of expected value

Game: \$2 to play. Roll a die. Win \$10 if you get a 6. Play many games. Should you expect to make or lose money? How much?

$$S = \{1, 2, 3, 4, 5, 6\}.$$

$$\mathbb{P} : S \rightarrow \mathbb{Q}_+ \text{ given by } p(j) = \frac{1}{6} \quad \forall j \in \{1, \dots, 6\}.$$

$$\mathbb{P} : \mathcal{P}(S) \rightarrow \mathbb{Q}_+ \text{ given by } \mathbb{P}(E) = \frac{|E|}{6} \text{ (equally likely outcomes).}$$

$$X : S \rightarrow \mathbb{Q} \text{ defined by } X(j) = \begin{cases} 10 - 2 = 8 & \text{if } j = 6, \\ -2 & \text{otherwise.} \end{cases}$$

X is your gain (or loss), which is a random variable.

$$\mathbb{E}(X) = \sum_{j=1}^6 \frac{1}{6} \times X(j) = 5 \left(\frac{1}{6} \times -2 \right) + \left(\frac{1}{6} \times 8 \right) = \frac{-2}{6} = -\frac{1}{3}.$$

If you play this game 30 times, you should expect to *lose* $30(\frac{1}{3}) = 10$ dollars.

Independent Events

For a sample space S with probability density function $\mathbb{P} : S \rightarrow \mathbb{Q}_+$, $E, F \in \mathcal{P}(S)$ are called **independent events** when

$$\mathbb{P}(E \cap F) = \mathbb{P}(E) \times \mathbb{P}(F)$$

Illustration:

Toss two coins:

$S = \{\text{H,T}\}^2 = \{\text{HH,HT,TH,TT}\}$ with equally likely outcomes.

- $E = \{\text{HH,HT}\}$ (1st coin gives Head), $\mathbb{P}(E) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$,
 $F = \{\text{HT,TT}\}$ (2nd coin gives Tail), $\mathbb{P}(F) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$.
 E, F are independent (as we would expect) since

$$\mathbb{P}(E \cap F) = \mathbb{P}(\{\text{HT}\}) = \frac{1}{4} = \frac{1}{2} \times \frac{1}{2} = \mathbb{P}(E) \times \mathbb{P}(F).$$

- $G = \{\text{HT,TH,HH}\}$ (at least one Head), $\mathbb{P}(G) = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = \frac{3}{4}$,
 $K = \{\text{TH,HT,TT}\}$ (at least one Tail), $\mathbb{P}(K) = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} = \frac{3}{4}$.
 G, K are **not** independent (again as we would expect) since
 $\mathbb{P}(G \cap K) = \mathbb{P}(\{\text{HT,TH}\}) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2} \neq \frac{9}{16} = \frac{3}{4} \times \frac{3}{4} = \mathbb{P}(G) \times \mathbb{P}(K)$.

Independent random variables

For a sample space S with probability density function $\mathbb{P} : S \rightarrow \mathbb{Q}_+$, $X, Y : S \rightarrow \mathbb{Q}$ are called **independent random variables** when

$$\begin{aligned}\forall a \in \text{Range}(X) \quad \forall b \in \text{Range}(Y) \\ \{X = a\}, \{Y = b\} \text{ are independent.}\end{aligned}$$

Illustration:

The last illustration showed that, when tossing two coins, getting T on the second toss is independent of getting H on the first.

Using random variables we get the expected result that *any* results of the two tosses are independent:

Let X, Y be number of heads (0 or 1) on the 1st, 2nd toss respectively.

Then for *any* $a, b \in \{0, 1\}$:

$$\mathbb{P}(\{X=a\}) = \mathbb{P}(\{Y=b\}) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2} \text{ and } \mathbb{P}(\{X=a\} \cap \{Y=b\}) = \frac{1}{4},$$

and hence the events $\{X=a\}, \{Y=b\}$ are independent because

$$\mathbb{P}(\{X=a\} \cap \{Y=b\}) = \frac{1}{4} = \frac{1}{2} \times \frac{1}{2} = \mathbb{P}(\{X=a\}) \times \mathbb{P}(\{Y=b\}).$$

Thus, by the above definition, X, Y are independent.

Independent random variables — Example

Toss a regular fair die. $S = \{1, \dots, 6\}$, $\mathbb{P}(i) = \frac{1}{6}$, $i = 1, \dots, 6$.

Let $X, Y : S \rightarrow \mathbb{Q}$ be random variables as follows:

Table 1: Definition of X and Y

s	1	2	3	4	5	6
$s \bmod 2 = X(s)$	1	0	1	0	1	0
$s \bmod 3 = Y(s)$	1	2	0	1	2	0

Table 2: Probabilities

a	0	1	2
$\mathbb{P}(\{X=a\})$	$\frac{1}{2}$	$\frac{1}{2}$	0
$\mathbb{P}(\{Y=a\})$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

The columns in Table 1 are all different and cover all possible combinations of values of X, Y . This ensures that each pair of values $(X, Y) = (a, b)$ relates to a unique s (Table 3), and hence has probability $\mathbb{P}(s)$ ($= \frac{1}{6}$).

Using Table 2 it now follows that, for any $a \in \{0, 1\}$ $b \in \{0, 1, 2\}$ the events $\{X=a\}, \{Y=b\}$ are independent because

$$\mathbb{P}(\{X=a\} \cap \{Y=b\}) = \frac{1}{6} = \frac{1}{2} \times \frac{1}{3} = \mathbb{P}(\{X=a\}) \times \mathbb{P}(\{Y=b\}).$$

Table 3: s

$b =$	0	1	2
$a=0$	6	4	2
$a=1$	3	1	5

Thus, by definition, the random variables X, Y are independent.

Non-independent random variables — Example

Let's modify the previous example just a little:

Toss a regular fair die. $S = \{1, \dots, 6\}$, $\mathbb{P}(i) = \frac{1}{6}$, $i = 1, \dots, 6$.

Let $Y, Z : S \rightarrow \mathbb{Q}$ be random variables as follows:

Table 1: Definition of Y and Z

s	1	2	3	4	5	6
$s \bmod 3 = Y(s)$	1	2	0	1	2	0
$s \bmod 4 = Z(s)$	1	2	3	0	1	2

Table 2: Probabilities

a	0	1	2	3
$\mathbb{P}(\{Y=a\})$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0
$\mathbb{P}(\{Z=a\})$	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

Notice that, in the Table 1, many potential columns are not present. For example there is no value of s for which $Y(s)=0$ and $Z(s)=0$. Using this, and also Table 2, we now have

$$\mathbb{P}(\{Y=0\} \cap \{Z=0\}) = 0 \neq \frac{1}{3} \times \frac{1}{6} = \mathbb{P}(\{Y=0\}) \times \mathbb{P}(\{Z=0\}),$$

and so the events $\{Y=0\}$, $\{Z=0\}$ are not independent.

It follows that the random variables Y, Z are not independent.

Challenge: Are the random variables X, Z independent?

END OF SECTION C2

C3. Markov Processes

Notes originally prepared by Judy-anne Osborn.
Editing, expansion and additions by Malcolm Brooks.

This material is not covered in the textbook by Epp. Check books on Finite Mathematics or Discrete Mathematics in the Library,
e.g. *Finite Mathematics* By Maki & Thompson Chapter 8

Markov processes

Markov processes are about probabilities. We consider

- the **state** of a system, amongst a finite number of possibilities
- the **probability** of moving between states in one time-step,
- and the probable state after **many** time-steps.
- We often don't make a sharp distinction between **proportions** and **probabilities** as you will see in the examples.

This works well for large samples but you may need to be careful with small samples.

Introductory example

adapted from 'Finite Mathematics', Maki & Thompson

A freelance computer network consultant, let's call her Cathy, is employed in weekly contracts. Each week she is either:

- employed (E) or
- unemployed (U).

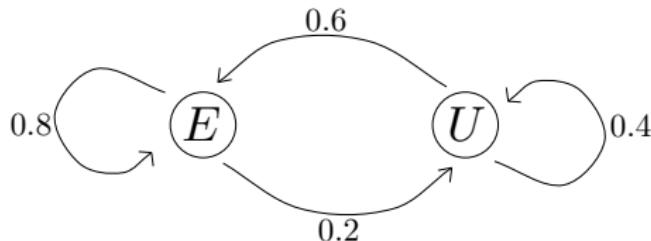
Her records support the following assumptions:

- (a) If she's employed this week, then next week she'll be employed with probability 0.8 and unemployed with probability 0.2.
- (b) If she's unemployed this week, then next week she'll be employed with probability 0.6 and unemployed with probability 0.4.

System, States and Transitions

We can model Cathy's situation by a Markov process:

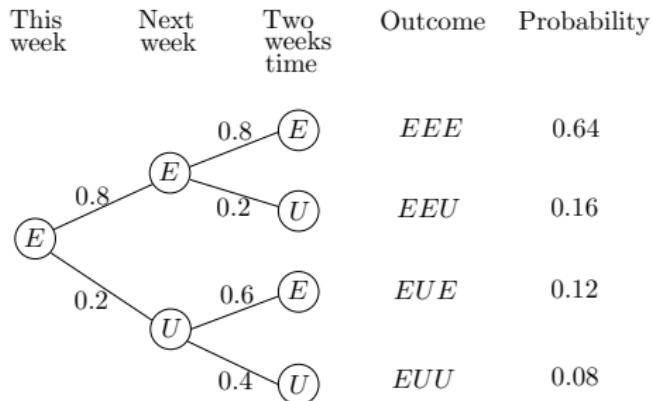
- The **system** is Cathy herself.
- The system can be in one of two **states**:
 - E: Employed
 - U: Unemployed
- A **Transition Diagram** encodes the transition probabilities:



It is a property of a Markov Process that the probability of stepping from one state to another *only depends on the current state*.

Two time-steps

If Cathy is employed this week, what is the probability that she will be employed two weeks from now? We can use a tree:



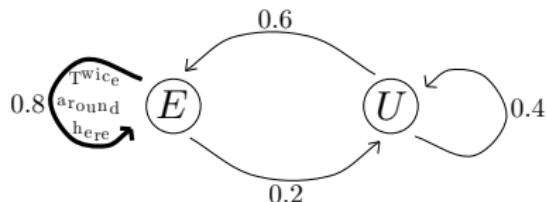
From the tree diagram, the probability that Cathy will be employed two weeks from now is

$$\Pr(\text{EEE or EUE}) = \Pr(\text{EEE}) + \Pr(\text{EUE}) = 0.64 + 0.12 = 0.76.$$

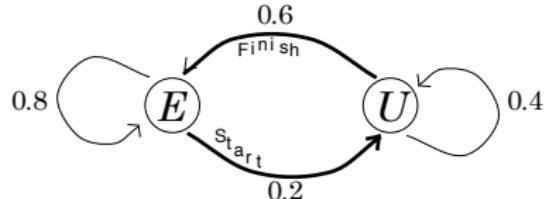
Two time-steps on the transition diagram

Starting employed, then employment after two weeks can be shown on the transition diagram as

either

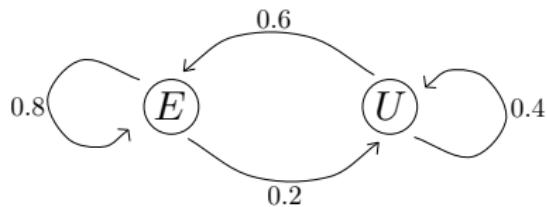


or



Transition Matrix

The information in Cathy's transition diagram



can be encoded in the **transition matrix**

$$T = \begin{matrix} & \nearrow & \searrow \\ \begin{matrix} E \\ U \end{matrix} & \left[\begin{matrix} 0.8 & 0.2 \\ 0.6 & 0.4 \end{matrix} \right] \end{matrix}$$

State Vectors

The **state vector** x_n shows probabilities of being in each state after n time-steps.

In Cathy's case, the state vector has two entries.

- The first entry records probability of employment.
- The second entry records probability of unemployment.

Initially ($n = 0$) the only possible values for the state vector are

- $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ if Cathy is employed, and
- $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ if Cathy is unemployed,

since we assumed that, in any week, Cathy was either:
100% employed and 0% unemployed, or
0% employed and 100% unemployed.

Transpose of the Transition Matrix

Recall that the transition (transfer) matrix is

$$T = \begin{bmatrix} & \nearrow E & \nearrow U \\ \nearrow E & 0.8 & 0.2 \\ \nearrow U & 0.6 & 0.4 \end{bmatrix}$$

We will need

$$T' = \begin{bmatrix} & \swarrow E & \swarrow U \\ \swarrow E & 0.8 & 0.6 \\ \swarrow U & 0.2 & 0.4 \end{bmatrix}$$

This is the **transpose** of the transition matrix.

It is very important to remember that it is always the *transpose* of the transition matrix that is used in calculations.

Using Matrices and State Vectors

Suppose Cathy is employed in Week 0.

The initial state vector is $\mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

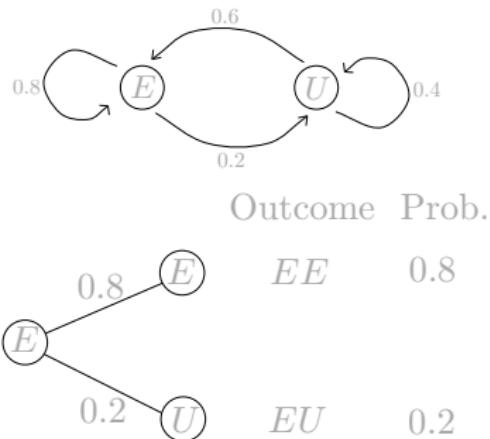
After one time-step, i.e. next week (Week 1), her probabilities of being employed or not are given by the state vector \mathbf{x}_1 .

This can be expressed as:

$$\mathbf{x}_1 = \mathbf{T}' \mathbf{x}_0$$

$$= \begin{bmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$$



Two time-steps

In Week 2, i.e. after two time-steps, Cathy's chances of work are given by the state vector \mathbf{x}_2 .

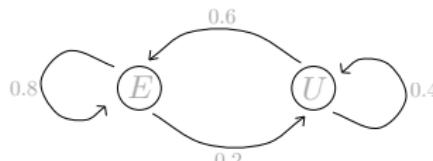
This can be calculated by:

$$\mathbf{x}_2 = \mathbf{T}' \mathbf{x}_1$$

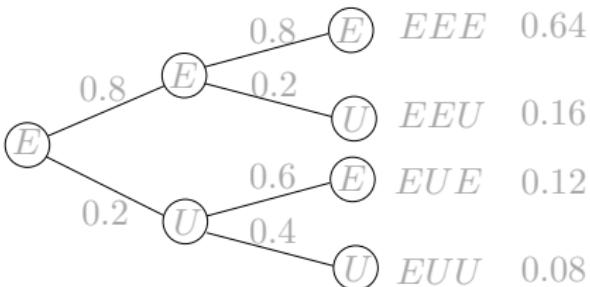
$$= \begin{bmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.64 + 0.12 \\ 0.16 + 0.08 \end{bmatrix}$$

$$= \begin{bmatrix} 0.76 \\ 0.24 \end{bmatrix}$$



Outcome Prob.



n time-steps

$$\text{Continuing: } \mathbf{x}_3 = T' \mathbf{x}_2 = \begin{bmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 0.76 \\ 0.24 \end{bmatrix} = \begin{bmatrix} 0.752 \\ 0.248 \end{bmatrix}$$

$$\mathbf{x}_4 = T' \mathbf{x}_3 = \begin{bmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 0.752 \\ 0.248 \end{bmatrix} = \begin{bmatrix} 0.7504 \\ 0.2496 \end{bmatrix}$$

$$\mathbf{x}_5 = T' \mathbf{x}_4 = \begin{bmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 0.7504 \\ 0.2496 \end{bmatrix} = \begin{bmatrix} 0.75008 \\ 0.24992 \end{bmatrix}$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$$

$$\text{Thus: } \mathbf{x}_1 = T' \mathbf{x}_0$$

$$\mathbf{x}_2 = T' \mathbf{x}_1 = T' T' \mathbf{x}_0 = (T')^2 \mathbf{x}_0$$

$$\mathbf{x}_3 = T' \mathbf{x}_2 = T' T' T' \mathbf{x}_0 = (T')^3 \mathbf{x}_0$$

$$\vdots \quad \vdots \quad \vdots$$

$$\mathbf{x}_n = (T')^n \mathbf{x}_0$$

The n^{th} power of T'

Successive powers of $T' = \begin{bmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{bmatrix}$ are:

$$(T')^2 = T'T' = \begin{bmatrix} 0.76 & 0.72 \\ 0.24 & 0.28 \end{bmatrix}$$

$$(T')^3 = T'(T')^2 = \begin{bmatrix} 0.752 & 0.744 \\ 0.248 & 0.256 \end{bmatrix}$$

$$(T')^4 = T'(T')^3 = \begin{bmatrix} 0.7504 & 0.7488 \\ 0.2496 & 0.2512 \end{bmatrix}$$

$$(T')^5 = T'(T')^4 = \begin{bmatrix} 0.75008 & 0.74976 \\ 0.24992 & 0.25024 \end{bmatrix}$$

⋮ ⋮ ⋮

So: $(T')^n \simeq \begin{bmatrix} 0.75 & 0.75 \\ 0.25 & 0.25 \end{bmatrix}$ for large values of n .

The significance of $(T')^n$ for large n

We have seen that, for Cathy, $(T')^n \simeq \begin{bmatrix} 0.75 & 0.75 \\ 0.25 & 0.25 \end{bmatrix}$ for large values of n .

Notice that the columns of this matrix are equal, and that

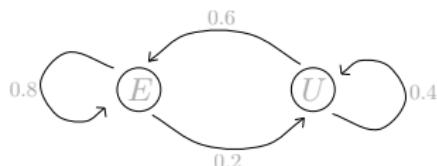
$$\begin{bmatrix} 0.75 & 0.75 \\ 0.25 & 0.25 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0.75 & 0.75 \\ 0.25 & 0.25 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}.$$

So, irrespective of the initial state, in the long term the state vector becomes approximately $\begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}$. This means

No matter what, eventually Cathy will be employed 75% of the time.

The Steady State Vector

Cathy's employment situation can now be summed up by:



$$(T')^n \simeq \begin{bmatrix} 0.75 & 0.75 \\ 0.25 & 0.25 \end{bmatrix}$$

When, as here, the columns of $(T')^n$ tend to become all the same for large values of n , this column \mathbf{v} (in this case $\begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}$) is called a **steady state vector** because then

$$(T')^n \mathbf{u} \simeq \mathbf{v}$$

for any initial state vector \mathbf{u} .

The steady state vector is an eigenvector

The steady state vector has the property that multiplication by the transposed transition matrix does not change it, e.g. for Cathy:

$$T' \mathbf{v} = \begin{bmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{bmatrix} \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 0.60 + 0.15 \\ 0.15 + 0.10 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix} = \mathbf{v}.$$

More generally, for *any* transition matrix T we call *any* vector \mathbf{v} for which

$$T' \mathbf{v} = \mathbf{v}$$

a *steady state vector* for T .

Caution: A Markov process may not always reach a steady state!

Note: The definition of \mathbf{v} makes it a special case of an **eigenvector**.

Courses in linear algebra cover more about eigenvectors and also numbers called **eigenvalues**.

A steady state vector has an associated eigenvalue of 1.

Some Definitions

It is time to collect and formalise the definitions made informally while discussing the example of Cathy's employment.

A **probability vector** is a vector with non-negative entries that sum to 1.

A **stochastic matrix** is a square matrix all whose rows are probability vectors.

A (discrete) **Markov process** is a system that, at each of a sequence of time steps, can be in exactly one of a finite number k of states, with the probability of the system being in any particular state at time step $n \geq 1$ being dependent only on

- (i) its state at the $(n - 1)$ -th time step, and
- (ii) a fixed stochastic matrix $T \in M_k(Q_+)$ called the **transition matrix** of the process.

Some Definitions (continued)

The (i,j) -entry T_{ij} of the transition matrix T specifies the probability that the system will be in the j -th state at any time step $n \geq 1$, given that it was in the i -th state at time step $n - 1$.

A **transition diagram** for a Markov process is a complete weighted directed graph with k vertices representing the states of the system and the edge from the i -th vertex to the j -th vertex labelled with the probability T_{ij} .

The **n -th state vector x_n** is a column probability vector with k entries. The i -th entry records a probability of the system being in the i -th state at time step n .

A **steady state vector v** is any probability vector for which $T'v = v$.

Using the transition matrix

The following theorem generalises to any number k of states what we saw in Cathy's example for just two states:

Theorem: Let $T = (T_{ij})_{1 \leq i,j \leq k}$ be the transition matrix for a k state Markov process with state vectors \mathbf{x}_n , $n \in \mathbb{N}$. Then $\forall n \geq 1$:

- (i) $\mathbf{x}_n = T' \mathbf{x}_{n-1}$
- (ii) $(T^n)_{ij}$ is the n -step i -to- j transition probability.
- (iii) $\mathbf{x}_n = (T')^n \mathbf{x}_0$

Proof of (i): Let $\mathbf{x}_n = (p_i)_{1 \leq i \leq k}$ and $\mathbf{x}_{n-1} = (q_i)_{1 \leq i \leq k}$. Then

$$\begin{aligned} p_i &= \text{probability of being in state } i \text{ at step } n \\ &= \sum_{j=1}^k (T_{ij})(\text{prob. of being in state } j \text{ at step } n-1) = \sum_{j=1}^k T_{ij} q_j. \end{aligned}$$

The result follows by the definition of matrix multiplication.

Proofs of (ii) and (iii): These are simple corollaries to (i).

Markov Processes Have No Memory

- The state of a Markov Process at time n only depends on fixed transition probabilities and its state at time $n - 1$.
- It does not depend on the state at any earlier time. In other words, it is a *first-order* (matrix) recurrence.
- Because of this, Markov processes are said to “have no memory”.

Finding steady state vectors

- One way to find a steady state vector of a Markov process is to do as we did in the example - namely multiply together enough copies of T' - or equivalently T - to see the higher powers tending to a limit.
- It does not matter whether we transpose first and then exponentiate or the other way around, since

$$(T^n)' = (T')^n$$

- There are more direct methods of finding steady state vectors, and we demonstrate these in the next example.

Another example: weather in ‘Oz’

We investigate the weather on the island of Oz.

(This, it will be seen, is a rather simplified example, to illustrate the principles without too much heavy calculation.)

Weather in Oz is not great, e.g. it never has two fine days in a row.

If the weather on a particular day is known, we cannot predict exactly what the weather will be the next day, but we can give probabilities of various kinds of weather as follows:

There are only three kinds: fine (F), cloudy (C) and rain (R).

Here is the behaviour:

- After F, the weather is equally likely to be C or R.
- After C, the probabilities are $\frac{1}{4}$ for F, $\frac{1}{4}$ for C and $\frac{1}{2}$ for R.
- After R, the probabilities are $\frac{1}{4}$ for F, $\frac{1}{2}$ for C and $\frac{1}{4}$ for R.

Displaying the Oz weather system

Here are three ways to show the probabilities:

As a table:

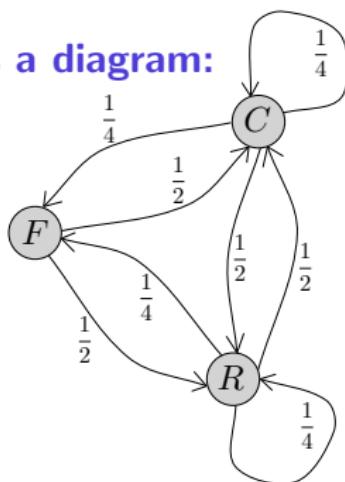
Probabilities of weather tomorrow are:

		fine	cloudy	rain
		fine	$\frac{1}{2}$	$\frac{1}{2}$
Given that the weather today is:	fine	0	$\frac{1}{2}$	$\frac{1}{2}$
	cloudy	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$
	rain	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$

As a matrix:

$$T = \begin{bmatrix} & F & C & R \\ F & 0 & 1/2 & 1/2 \\ C & 1/4 & 1/4 & 1/2 \\ R & 1/4 & 1/2 & 1/4 \end{bmatrix}$$

As a diagram:



Next day in Oz, via transition matrix

- Given probabilities on Day n , we can find probabilities on Day $n + 1$.
- We need a state vector. Let us call the probabilities on day n
 - Prob x that it will be fine,
 - Prob y that it will be cloudy, and
 - Prob z that it will rain.

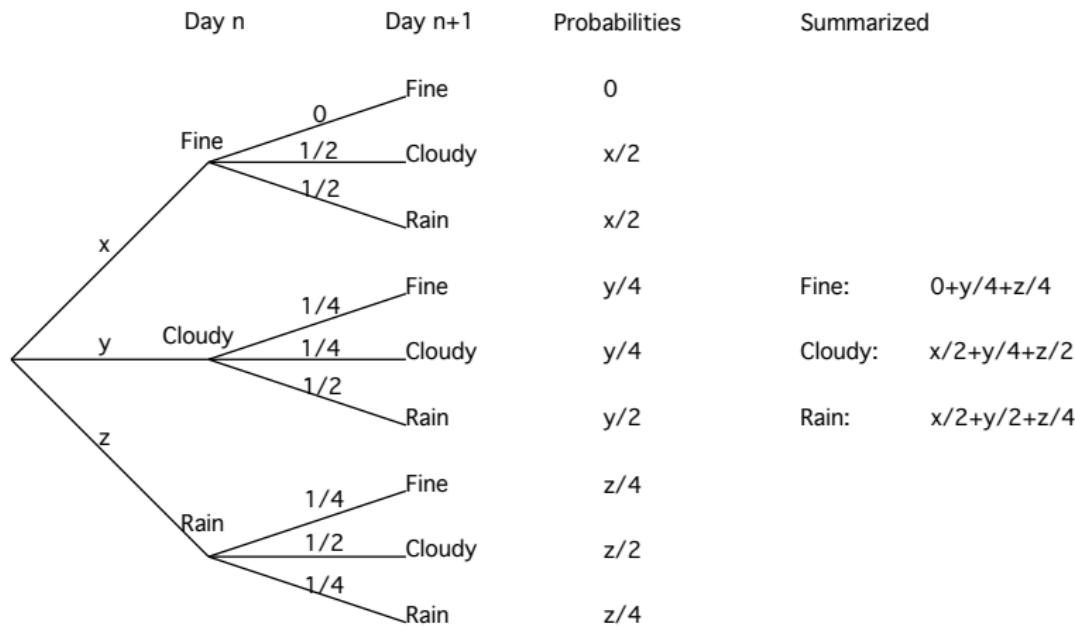
- Define $\mathbf{x}_n = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

- Then, according to the Markov process theorem:

$$\begin{aligned}\mathbf{x}_{n+1} &= T' \mathbf{x}_n \\ &= \begin{bmatrix} 0 & 1/4 & 1/4 \\ 1/2 & 1/4 & 1/2 \\ 1/2 & 1/2 & 1/4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (1/4)y + (1/4)z \\ (1/2)x + (1/4)y + (1/2)z \\ (1/2)x + (1/2)y + (1/4)z \end{bmatrix}\end{aligned}$$

Next day in Oz, via probability tree

Let's check that the probabilities obtained using the transition matrix agree with those obtained using a probability tree:



Yes, the state vector \mathbf{x}_{n+1} and probability tree agree.



Next week in Oz

Suppose we look outside
and see that today is fine.
Then we can say that

$$\mathbf{x}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

That is, the probability of fine is 1, the probability of cloudy is 0 and the probability of rain is 0 today.

Then the probabilities
one week from today
(Day 7) are given by

Using a computer to calculate the 7th power of the matrix, we get

$$\mathbf{x}_7 = \begin{bmatrix} 0 & 1/4 & 1/4 \\ 1/2 & 1/4 & 1/2 \\ 1/2 & 1/2 & 1/4 \end{bmatrix}^7 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 819/4096 \\ 3277/8192 \\ 3277/8192 \end{bmatrix}.$$

Perhaps decimals would be more illuminating?

Days 1 through 10 in Oz

Computer calculations give:

$$\mathbf{x}_1 = \begin{bmatrix} 0 \\ .5 \\ .5 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} .250 \\ .375 \\ .375 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} .18750 \\ .40625 \\ .40625 \end{bmatrix}, \quad \mathbf{x}_4 = \begin{bmatrix} .2031250 \\ .3984375 \\ .3984375 \end{bmatrix},$$

$$\mathbf{x}_5 = \begin{bmatrix} .199218750 \\ .400390625 \\ .400390625 \end{bmatrix}, \quad \mathbf{x}_6 = \begin{bmatrix} .1999511719 \\ .4000244141 \\ .4000244141 \end{bmatrix}, \quad \mathbf{x}_7 = \begin{bmatrix} .19995511719 \\ .4000244141 \\ .4000244141 \end{bmatrix},$$

$$\mathbf{x}_8 = \begin{bmatrix} .2000122070 \\ .3999938965 \\ .3999938965 \end{bmatrix}, \quad \mathbf{x}_9 = \begin{bmatrix} .1999969438 \\ .4000015260 \\ .4000015260 \end{bmatrix}, \quad \mathbf{x}_{10} = \begin{bmatrix} .2000007629 \\ .3999996185 \\ .3999996185 \end{bmatrix}.$$

A steady state for the weather in Oz

These values seem to be converging to a long-term steady state of

$$S = \begin{bmatrix} .2 \\ .4 \\ .4 \end{bmatrix},$$

i.e. a probability of 0.2 of fine weather, a probability of 0.4 of cloudy weather and a probability of 0.4 of rainy weather.
As they must, these probabilities sum to 1.

To check that this S really is a steady state vector, we calculate

$$T'S = \begin{bmatrix} 0 & 0.25 & 0.25 \\ 0.50 & 0.25 & 0.50 \\ 0.50 & 0.50 & 0.25 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.1 + 0.1 \\ 0.1 + 0.1 + 0.2 \\ 0.1 + 0.2 + 0.1 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}.$$

Therefore

$$T'S = S. \quad \checkmark$$

Finding steady state vectors

We derive another way to find steady state vectors, illustrating with weather from Oz.

Assume that $S = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$ is a steady state vector.

Then, as we know, S is an eigenvector of T' . i.e.

$$T'S = S$$

(In other words we have reached a stage where the probabilities don't change from day to day any more.)

Notice that we can rearrange this equation in the form

$$T'S - S = 0.$$

Remember to *think about what kinds of objects* are in this equation

$$T'S - S = 0$$

- The right-hand-side is a column vector of zeros.
- S is a column vector of unknowns (x , y and z).
- $T'S$ is the product of a square matrix and a column vector - making a column vector.

We can rewrite the equation yet again in the form

$$T'S - IS = 0$$

where I is the 3×3 identity matrix.

Finally using a distributive law, we can re-write it as:

$$(T' - I)S = 0.$$

Methods for solving the matrix equation

There are several ways to solve the equation

$$(T' - I)S = 0.$$

1. Gaussian (or Gauss-Jordan) elimination. However,
Gaussian elimination is not taught nor assumed in this course.
Use it if you know it, but there are other ways:
2. For 2×2 systems we can use the matrix inverse formula.
3. For larger systems we can use computer applications such as:
Matrix Reshish: <https://matrix.reshish.com>
MatrixCalc: <https://matrixcalc.org/en/>
or, less conveniently but more robustly,
WolframAlpha: <https://www.wolframalpha.com/>

Using Gauss-Jordan elimination

First re-write in augmented form as:

$$[T' - I | 0]$$

and then row-reduce to solve for the unknowns in S .

Since

$$T' - I = \begin{bmatrix} 0 & 1/4 & 1/4 \\ 1/2 & 1/4 & 1/2 \\ 1/2 & 1/2 & 1/4 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1/4 & 1/4 \\ 1/2 & -3/4 & 1/2 \\ 1/2 & 1/2 & -3/4 \end{bmatrix}$$

our augmented matrix is

$$\left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 1/2 & -3/4 & 1/2 & 0 \\ 1/2 & 1/2 & -3/4 & 0 \end{array} \right]$$

Row reducing,

$$\left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 1/2 & -3/4 & 1/2 & 0 \\ 1/2 & 1/2 & -3/4 & 0 \end{array} \right] \sim \left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] \quad R'_2 = (-4/5)R_2$$

$$\sim \left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 1 & -3/2 & 1 & 0 \\ 1 & 1 & -3/2 & 0 \end{array} \right] \quad R'_2 = 2R_2 \quad R'_3 = 2R_3 \quad \sim \left[\begin{array}{ccc|c} 1 & -1/4 & -1/4 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] \quad R'_1 = -R_1$$

$$\sim \left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 0 & -5/4 & 5/4 & 0 \\ 0 & 5/4 & -5/4 & 0 \end{array} \right] \quad R'_2 = R_2 + R_1 \quad R'_3 = R_3 + R_1 \quad \sim \left[\begin{array}{ccc|c} 1 & 0 & -1/2 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] \quad R'_1 = R_1 + (1/4)R_2$$

$$\sim \left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 0 & -5/4 & 5/4 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] \quad R'_3 = R_3 + R_2 \quad \begin{matrix} \uparrow \\ \text{This column tells us we need a parameter} \end{matrix}$$

Let $z = t$, $t \in \mathbb{R}$

So our original matrix equation is equivalent to

$$\begin{bmatrix} 1 & 0 & -1/2 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix};$$

that is, to just the two equations

$$x - (1/2)z = 0$$

$$y - z = 0.$$

Substituting the parameter $z = t$ gives

$$x - (1/2)t = 0$$

$$y - t = 0,$$

leading to the solution $x = (1/2)t$

$$y = t$$

$$z = t.$$

Recall we were solving $(T' - I)S = 0$.

We have found that this equation has an infinite family of solutions for S in the form

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (1/2)t \\ t \\ t \end{bmatrix}, \quad t \in \mathbb{R}$$

But a steady-state vector has an extra property: it must be a **probability vector**, i.e. its entries must sum to 1.

Hence we fix a unique solution by requiring that $x + y + z = 1$, i.e.

$$(1/2)t + t + t = 1 \implies (5/2)t = 1 \implies t = 2/5$$

so that S is given by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1/5 \\ 2/5 \\ 2/5 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}$$

– the same as we found before by exponentiating and guessing.

A short cut

A short cut to this process is to take the augmented matrix $[T' - I|0]$ as below,

$$\left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 1/2 & -3/4 & 1/2 & 0 \\ 1/2 & 1/2 & -3/4 & 0 \end{array} \right]$$

throw away the last row and replace it with $[1\dots 1|1]$, as in

$$\left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 1/2 & -3/4 & 1/2 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right]$$

and solve this new system to directly obtain the unique solution for S .

After row-reducing the new system we find that

$$\left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 1/2 & -3/4 & 1/2 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right] \sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & 1/5 \\ 0 & 1 & 0 & 2/5 \\ 0 & 0 & 1 & 2/5 \end{array} \right]$$

from which we read off the steady state vector

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1/5 \\ 2/5 \\ 2/5 \end{bmatrix}.$$

Can you figure out *why* this short cut works?

Solving by Computer (using Reshish)

The system of equations

$$\left[\begin{array}{ccc|c} -1 & 1/4 & 1/4 & 0 \\ 1/2 & -3/4 & 1/2 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right]$$

is entered into the Reshish Matrix Calculator, using the “Gauss-Jordan Elimination” Tool :

Note that I have chosen to use “fractional” coefficients, to ensure an exact solution.

Matrix input X

Complex numbers (more) i

Fractional ▼

	X ₁	X ₂	X ₃	b
1	-1	1/4	1/4	0
2	1/2	-3/4	1/2	0
3	1	1	1	1

Very detailed solution

Here is how Reshish responds:

Show solution

Solution set:

$$x_1 = 1/5$$

$$x_2 = 2/5$$

$$x_3 = 2/5$$

Back to the first example

We have seen that to find the steady state vector S for a Markov process with transition matrix T we need to solve the linear system that results from replacing the last equation in

$$(T' - I)S = 0$$

by the equation that says that S is a probability vector.

For Cathy's employment process we had

$$T = \begin{bmatrix} 0.8 & 0.2 \\ 0.6 & 0.4 \end{bmatrix}$$

and, by a 'guess and check' method, we discovered that

$$S = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}.$$

Solution by matrix inverse

Because T is 2×2 , and we have a formula for the inverse of a 2×2 matrix, we can find Cathy's steady state vector directly, without Gaussian elimination or computer. There are three steps:

1. Write out the matrix equation $(T' - I)S = 0$:

$$\left(\begin{bmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\text{i.e. } \begin{bmatrix} -0.2 & 0.6 \\ 0.2 & -0.6 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

2. Replace the second equation by $x + y = 1$:

$$\begin{bmatrix} -0.2 & 0.6 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Solution by matrix inverse (conclusion)

3. Solve this system using matrix inverse:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -0.2 & 0.6 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \frac{1}{-0.2 - 0.6} \begin{bmatrix} 1 & -0.6 \\ -1 & -0.2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \frac{1}{-0.8} \begin{bmatrix} -0.6 \\ -0.2 \end{bmatrix} = \begin{bmatrix} 6/8 \\ 2/8 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.25 \end{bmatrix}$$

New Example — Colours of flowers

A species of flower (carnations say) has three colour varieties. The relevant genetics are as shown in the table:

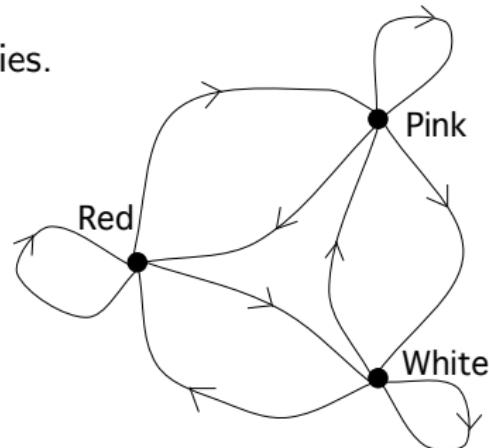
Colour	Genotype
Red	RR
Pink	RW
White	WW

At the nursery they are always crossed with the pink variety.
What will be the long term proportions of the three varieties?

First, we need the transition probabilities.
Can you work them out?

The transition matrix is

$$T = \begin{matrix} & \text{Red} & \text{Pink} & \text{White} \\ \text{Red} & \begin{bmatrix} 0.5 & 0.5 & 0 \end{bmatrix} \\ \text{Pink} & \begin{bmatrix} 0.25 & 0.5 & 0.25 \end{bmatrix} \\ \text{White} & \begin{bmatrix} 0 & 0.5 & 0.5 \end{bmatrix} \end{matrix} .$$



Finding the steady state

(a) $[T' - I|0]$ is

$$\left[\begin{array}{ccc|c} -0.5 & 0.25 & 0 & 0 \\ 0.5 & -0.5 & 0.5 & 0 \\ 0 & 0.25 & -0.5 & 0 \end{array} \right]$$

(b) Replacing the bottom row with all 1's gives

$$\left[\begin{array}{ccc|c} -0.5 & 0.25 & 0 & 0 \\ 0.5 & -0.5 & 0.5 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right]$$

Finding the steady state (cont.)

(c) Row reduction gives

$$\left[\begin{array}{ccc|c} -0.5 & 0.25 & 0 & 0 \\ 0.5 & -0.5 & 0.5 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right]$$

$$\sim \left[\begin{array}{ccc|c} 1 & -0.5 & 0 & 0 \\ 0 & 1 & -2 & 0 \\ 0 & 0 & 1 & 1/4 \end{array} \right] \quad R'_3 = (1/4)R_3$$

$$\sim \left[\begin{array}{ccc|c} -1 & 0.5 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array} \right] \quad R'_1 = 2R_1 \quad R'_2 = 2R_2$$

$$\sim \left[\begin{array}{ccc|c} 1 & -0.5 & 0 & 0 \\ 0 & 1 & 0 & 1/2 \\ 0 & 0 & 1 & 1/4 \end{array} \right] \quad R'_2 = R_2 + 2R_3$$

$$\sim \left[\begin{array}{ccc|c} -1 & 0.5 & 0 & 0 \\ 0 & -0.5 & 1 & 0 \\ 0 & 1.5 & 1 & 1 \end{array} \right] \quad R'_2 = R_2 + R_1 \quad R'_3 = R_3 + R_1$$

$$\sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & 1/4 \\ 0 & 1 & 0 & 1/2 \\ 0 & 0 & 1 & 1/4 \end{array} \right] \quad R'_1 = R_1 + (1/2)R_2$$

$$\sim \left[\begin{array}{ccc|c} -1 & 0.5 & 0 & 0 \\ 0 & -0.5 & 1 & 0 \\ 0 & 0 & 4 & 1 \end{array} \right] \quad R'_3 = R_3 + 3R_2$$

yielding $S = \begin{bmatrix} 1/4 \\ 1/2 \\ 1/4 \end{bmatrix}$.

Finding the steady state by computer

Alternatively, we can solve the system using the computer. For example, using Reshish:

Hence there is a unique steady state vector of

$$S = \begin{bmatrix} 1/4 \\ 1/2 \\ 1/4 \end{bmatrix}$$

So the species has a steady state in which 25% of the flowers are coloured red, 50% pink, and 25% white.

	X_1	X_2	X_3	b
1	-1/2	1/4	0	0
2	1/2	-1/2	1/2	0
3	1	1	1	1

Solution set:

$$\begin{aligned} x_1 &= 1/4 \\ x_2 &= 1/2 \\ x_3 &= 1/4 \end{aligned}$$

Checking the answer

The steady state vector S must be an eigenvector of T' .

Let's check:

$$\begin{aligned}
 T'S &= \begin{bmatrix} 1/2 & 1/4 & 0 \\ 1/2 & 1/2 & 1/2 \\ 0 & 1/4 & 1/2 \end{bmatrix} \begin{bmatrix} 1/4 \\ 1/2 \\ 1/4 \end{bmatrix} \\
 &= \begin{bmatrix} 1/8 + 1/8 \\ 1/8 + 1/4 + 1/8 \\ 1/8 + 1/8 \end{bmatrix} \\
 &= \begin{bmatrix} 1/4 \\ 1/2 \\ 1/4 \end{bmatrix}
 \end{aligned}$$

So yes, $T'S = S$. ✓

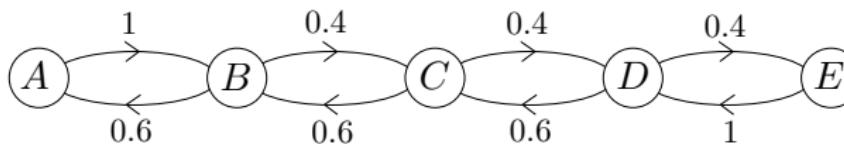
Will a Markov process always get to a steady state?

Not necessarily!

Example: chemical compounds in transition

Consider a chemical compound whose molecule can exist in any one of five states, termed A, B, C, D and E .

Each molecule frequently undergoes transitions from one state to another, always to an ‘adjacent’ state, according to the probabilities shown in the transition diagram.



The transition matrix for
this Markov Process is

$$T = \begin{bmatrix} & \curvearrowleft & A & B & C & D & E \\ A & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \\ B & \begin{bmatrix} 0.6 & 0 & 0.4 & 0 & 0 \end{bmatrix} \\ C & \begin{bmatrix} 0 & 0.6 & 0 & 0.4 & 0 \end{bmatrix} \\ D & \begin{bmatrix} 0 & 0 & 0.6 & 0 & 0.4 \end{bmatrix} \\ E & \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{bmatrix}$$

A beaker full of chemical

- Now suppose we have a beaker full of this chemical. We expect the proportions of A , B , C , D , E to relate to transition probabilities.
- What proportions of the compound will be in the various states?
- To do a thorough analysis of all possible behaviours of this Markov Process, you need to study '*eigenvalues and eigenvectors*' – a reason to take a course or read a book on [Linear Algebra](#).
- But let's see what we can figure out without those tools.

Chemical example — investigating with a computer

Suppose the beaker only contains form 'A' to start with, i.e.

$\mathbf{x}_0 = [1, 0, 0, 0, 0]'$. Then by computer to 6dp we find:

$$\mathbf{x}_{100} = (T')^{100} \mathbf{x}_0$$

$$= [0.415383, 0.000000, 0.461538, 0.000000, 0.123077]'$$

$$\mathbf{x}_{101} = T' \mathbf{x}_{100}$$

$$= [0.000000, 0.692308, 0.000000, 0.307692, 0.000000]'$$

and continuing in the same manner

$$\mathbf{x}_{102} = [0.415383, 0.000000, 0.461538, 0.000000, 0.123077]'$$

$$\mathbf{x}_{103} = [0.000000, 0.692308, 0.000000, 0.307692, 0.000000]'$$

$$\mathbf{x}_{104} = [0.415383, 0.000000, 0.461538, 0.000000, 0.123077]'$$

$$\mathbf{x}_{105} = [0.000000, 0.692308, 0.000000, 0.307692, 0.000000]'$$

:

:

It appears to alternate!

However starting with a beaker half full of A and half of B, i.e.
 $\mathbf{x}_0 = [0.5, 0.5, 0, 0, 0]'$, and again using formulae

$$\mathbf{x}_n = (T')^n \mathbf{x}_0 \quad \text{and} \quad \mathbf{x}_{n+1} = T' \mathbf{x}_n$$

repeatedly we get

$$\mathbf{x}_{100} = [0.207692, 0.346154, 0.230769, 0.153846, 0.061539]'$$

$$\mathbf{x}_{101} = [0.207692, 0.346154, 0.230769, 0.153846, 0.061539]'$$

$$\mathbf{x}_{102} = [0.207692, 0.346154, 0.230769, 0.153846, 0.061539]'$$

 \vdots
 \vdots

This looks like a steady state!

So this Markov Process is different to those we used to model employment, weather in Oz, and flower-colours because

eventual behaviour depends on where you start!

Steady state(s) for a beaker of chemical?

We can solve for the steady state to find out if it is unique.

We need to solve

$$T'S = S$$

for $S = [x_1, x_2, x_3, x_4, x_5]'$ subject to additional constraint

$$x_1 + x_2 + x_3 + x_4 + x_5 = 1.$$

We use the 'short cut' method:

- (a) First construct $[T' - I|0]$.
- (b) Then replace the last row with all 1's.
- (c) Then solve by Gaussian elimination or computer.

Steady state(s) for a beaker of chemical?

(a) $[T' - I | 0]$ is

$$\left[\begin{array}{ccccc|c} -1 & 0.6 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0.6 & 0 & 0 & 0 \\ 0 & 0.4 & -1 & 0.6 & 0 & 0 \\ 0 & 0 & 0.4 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0.4 & -1 & 0 \end{array} \right]$$

(b) Replace the last row with all 1's

$$\left[\begin{array}{ccccc|c} -1 & 0.6 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0.6 & 0 & 0 & 0 \\ 0 & 0.4 & -1 & 0.6 & 0 & 0 \\ 0 & 0 & 0.4 & -1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right]$$

$$\sim \left[\begin{array}{cccc|c} -1 & 0.6 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0.6 & 0 & 0 & 0 \\ 0 & 0.4 & -1 & 0.6 & 0 & 0 \\ 0 & 0 & 0.4 & -1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{array} \right]$$

$$\sim \left[\begin{array}{ccccc|c} 1 & -0.6 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1.5 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2.5 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4/65 \end{array} \right] \begin{matrix} R'_1 = -R_1 \\ R'_2 = (-5/2)R_2 \\ R'_3 = (-5/2)R_3 \\ R'_4 = (-5/2)R_4 \end{matrix}$$

$$\sim \left[\begin{array}{ccccc|c} -1 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & -0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0.4 & -1 & 0.6 & 0 & 0 \\ 0 & 0 & 0.4 & -1 & 1 & 0 \\ 0 & 1.6 & 1 & 1 & 1 & 1 \end{array} \right] \begin{matrix} R'_2 = R_2 + R_1 \\ R'_5 = R_5 + R_1 \end{matrix}$$

$$\sim \left[\begin{array}{ccccc|c} 1 & -0.6 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1.5 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 10/65 \\ 0 & 0 & 0 & 0 & 1 & 4/65 \end{array} \right] \begin{matrix} R'_4 = R_4 + (5/2)R_5 \end{matrix}$$

$$\sim \left[\begin{array}{ccccc|c} -1 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & -0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & -0.4 & 0.6 & 0 & 0 \\ 0 & 0 & 0.4 & -1 & 1 & 0 \\ 0 & 0 & 3.4 & 1 & 1 & 1 \end{array} \right] \begin{matrix} R'_3 = R_3 + R_2 \end{matrix}$$

$$\sim \left[\begin{array}{ccccc|c} 1 & -0.6 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1.5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 15/65 \\ 0 & 0 & 0 & 1 & 0 & 10/65 \\ 0 & 0 & 0 & 0 & 1 & 4/65 \end{array} \right] \begin{matrix} R'_3 = R_3 + (3/2)R_4 \end{matrix}$$

$$\sim \left[\begin{array}{ccccc|c} -1 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & -0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & -0.4 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & -0.4 & 1 & 0 \\ 0 & 0 & 0 & 6.1 & 1 & 1 \end{array} \right] \begin{matrix} R'_4 = R_4 + R_3 \\ R'_5 = R_5 + (8.5)R_3 \end{matrix}$$

$$\sim \left[\begin{array}{ccccc|c} 1 & -0.6 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 45/130 \\ 0 & 0 & 1 & 0 & 0 & 15/65 \\ 0 & 0 & 0 & 1 & 0 & 10/65 \\ 0 & 0 & 0 & 0 & 1 & 4/65 \end{array} \right] \begin{matrix} R'_2 = R_2 + (3/2)R_3 \end{matrix}$$

$$\sim \left[\begin{array}{ccccc|c} -1 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & -0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & -0.4 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & -0.4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 16.25 & 1 \end{array} \right] \begin{matrix} R'_5 = R_5 + (16.25)R_3 \end{matrix}$$

$$\sim \left[\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & 27/130 \\ 0 & 1 & 0 & 0 & 0 & 45/130 \\ 0 & 0 & 1 & 0 & 0 & 15/65 \\ 0 & 0 & 0 & 1 & 0 & 10/65 \\ 0 & 0 & 0 & 0 & 1 & 4/65 \end{array} \right] \begin{matrix} R'_1 = R_1 + (5/3)R_2 \end{matrix}$$

$$\sim \left[\begin{array}{ccccc|c} -1 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & -0.4 & 0.6 & 0 & 0 & 0 \\ 0 & 0 & -0.4 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & -0.4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4/65 \end{array} \right] \begin{matrix} R'_5 = (4/65)R_5 \end{matrix}$$

Steady state for a beaker of chemical - by computer

Decimal

	x_1	x_2	x_3	x_4	x_5	b
1	-1	0.6	0	0	0	0
2	1	-1	0.6	0	0	0
3	0	0.4	-1	0.4	0	0
4	0	0	0.4	-1	1	0
5	1	1	1	1	1	1

Solution set:

$$\begin{aligned}x_1 &= 27/130 \\x_2 &= 9/26 \\x_3 &= 3/13 \\x_4 &= 2/13 \\x_5 &= 4/65\end{aligned}$$

This confirms the **unique** steady-state solution found by row reduction on the previous slide:

$$\begin{bmatrix}x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5\end{bmatrix} = \begin{bmatrix}27/130 \\ 45/130 \\ 15/65 \\ 10/65 \\ 4/65\end{bmatrix} = \begin{bmatrix}0.2077 \\ 0.3462 \\ 0.2308 \\ 0.1538 \\ 0.0615\end{bmatrix}.$$

So the steady-state proportions of the five forms of the chemical are:

- A: 20.77%, B: 34.62%,
- C: 23.08%, D: 15.38%,
- E: 6.15%.

A steady state for a beaker of chemical - conclusion

We found that **provided** the beaker reaches a **steady-state**, then proportions of the various forms of the chemical remain stable at

$A : 20.77\%$, $B : 34.62\%$, $C : 23.08\%$, $D : 15.38\%$, $E : 6.15\%$.

Individual molecules DO change their form, but at a rate such that overall, proportions remain as above.

END OF SECTION C3

D1. Graph Theory

Notes originally prepared by Judy-anne Osborn.

Editing, expansion and additions by Malcolm Brooks.

Editing by Adam Piggott.

Text Reference (Epp) 3ed: Chapter 11

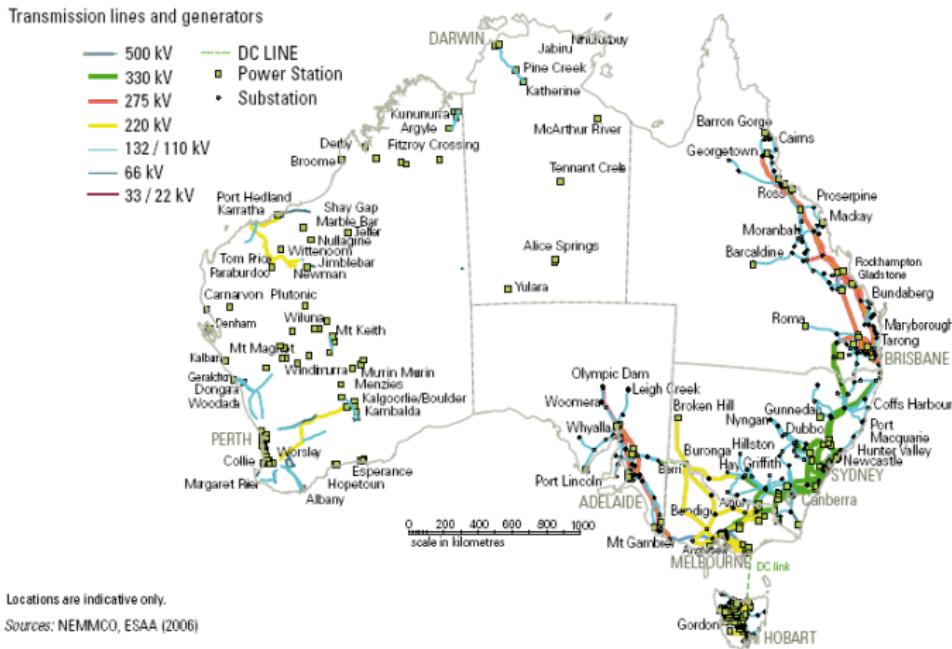
4ed: Chapter 10

5ed: Chapter 10

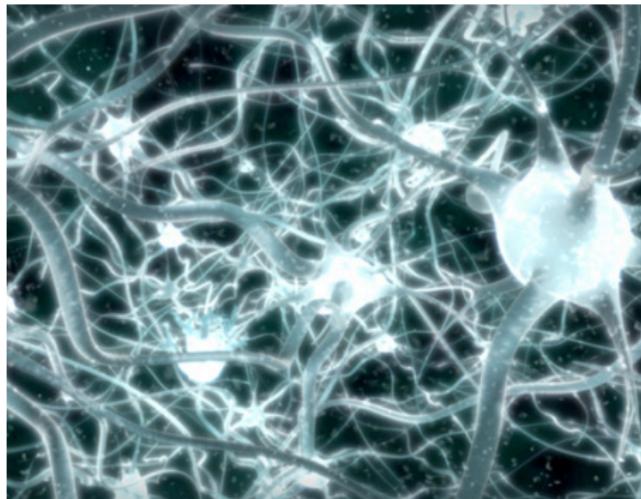
These references may not *completely* cover everything in this section, but they do have most it. They also contain a few items we do not cover.

Real-world phenomena often
modeled with graphs:

Australian Power Transmission Network



Brain Network



from documentary, 'Inside the living body'
<http://abcnews.go.com/2020/popup?id=3560899>

Graphs

A **graph** G is an ordered pair $G = (V(G), E(G))$ comprising:

- a set of **vertices** $V(G)$; and
- a multiset of **edges** $E(G)$, with each edge being a size-2 multiset of vertices.

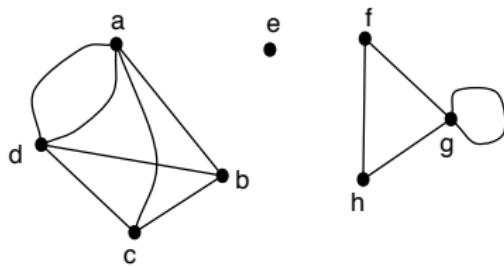
In this course we only consider **finite** graphs;
i.e. $V(G)$ is a finite set and $E(G)$ is a finite multiset.

Diagrams of Graphs

To draw a graph we use:

- dots/circles for vertices
- lines for edges

Example: The graph G



has vertex set $V(G) = \{a, b, c, d, e, f, g, h\}$

and edge multiset $E(G) =$

$\{\{a,b\}, \{a,c\}, \{a,d\}, \{a,d\}, \{b,c\}, \{b,d\}, \{c,d\}, \{f,g\}, \{f,h\}, \{g,g\}, \{g,h\}\}.$

A table of edges

- The same graph as a table of labelled edges:

Edge	Endpoints
e_1	$\{a, b\}$
e_2	$\{a, c\}$
e_3	$\{a, d\}$
e_4	$\{a, d\}$
e_5	$\{b, c\}$
e_6	$\{b, d\}$

Edge	Endpoints
e_7	$\{c, d\}$
e_8	$\{f, g\}$
e_9	$\{f, h\}$
e_{10}	$\{g, g\}$
e_{11}	$\{g, h\}$

- Notice that e_3 is distinct from e_4 even though both edges have the same endpoints.

A vertex adjacency listing

- The same graph as a vertex adjacency listing:

Vertex	Adjacent to:
a	b, c, d, d
b	a, c, d
c	a, b, d
d	a, a, b, c
e	
f	g, h
g	f, g, h
h	f, g

An adjacency matrix

- An adjacency matrix for the same graph

$$\begin{matrix}
 & a & b & c & d & e & f & g & h \\
 a & \begin{bmatrix} 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 b & \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 c & \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 d & \begin{bmatrix} 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 e & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
 f & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \\
 g & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \\
 h & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}
 \end{matrix}$$

- The $(i, j)^{\text{th}}$ entry is the number of edges between vertices i and j .
- Thus $a_{i,j}$ is number of ways that i is adjacent to j .

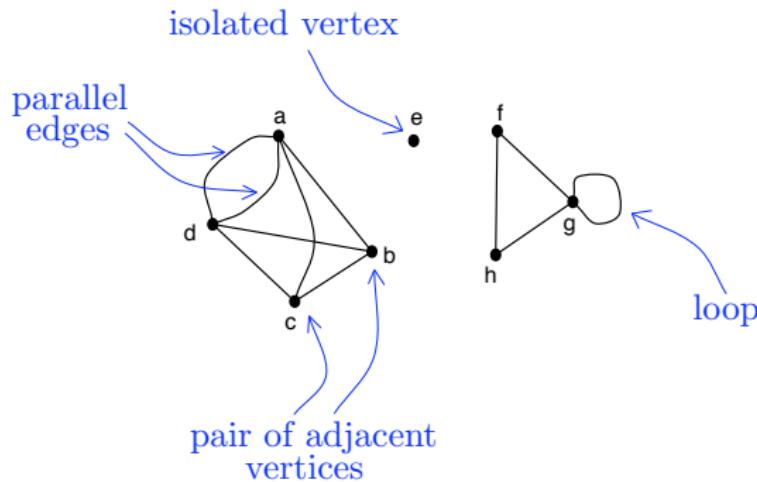
Some Graph Terminology

- An edge connects its **endpoints**.
- An edge with both endpoints the same is called a **loop**.
- Two edges may connect the same pair of endpoints, in which case they are said to be **parallel**.
- Two vertices are **adjacent** if they are connected by an edge; two edges are **adjacent** if they share an endpoint.

Some Graph Terminology

- An edge is **incident on** its endpoints.
- A vertex with no incident edges is **isolated**.
- A graph with no vertices (hence no edges) is **empty**.
- The **order** of a graph, G , is the number of vertices in it, i.e. $|V(G)|$.
(A graph of order '0' is empty.)

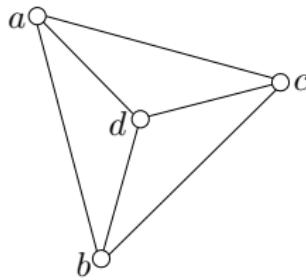
Some graph concepts illustrated



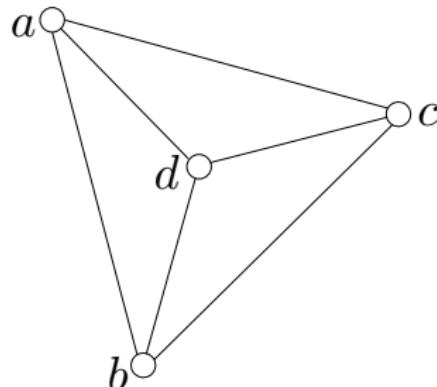
Another example

- Tetrahedron Graph

$G :$



- $V(G) = \{a, b, c, d\}$
- $E(G) = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\}$



- Tetrahedron Graph:

Adjacency listing:

Adjacency matrix:

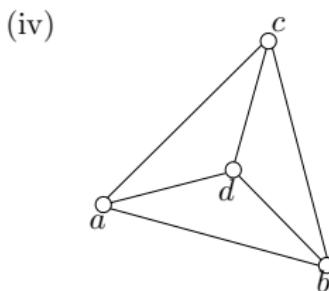
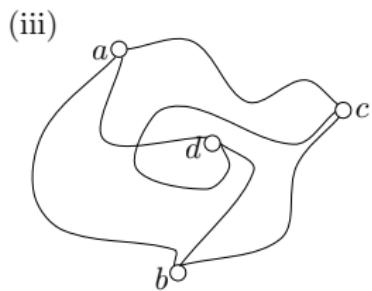
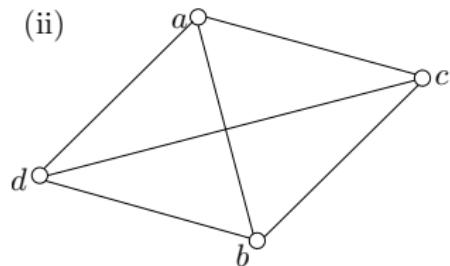
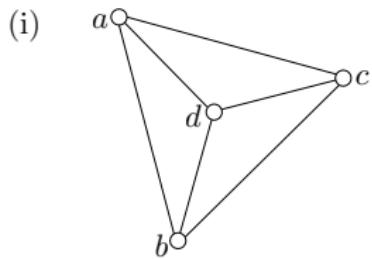
Vertex	Adjacent to:
a	b, c, d
b	a, c, d
c	a, b, d
d	a, b, c

$$\begin{matrix} & \begin{matrix} a & b & c & d \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{matrix}$$

More about graph diagrams

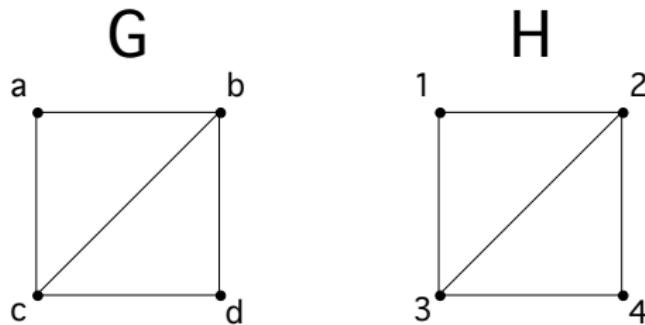
- Position, length, curvedness and orientation in a graph diagram do not matter for the graph represented.
- The only things which matter are that precisely those vertices in $V(G)$ are shown and precisely those edges in $E(G)$ are shown.
- For instance, the following diagrams all represent the same graph.

Four diagrams of the same graph



Isomorphic Graphs

Consider graphs G and H as pictured:

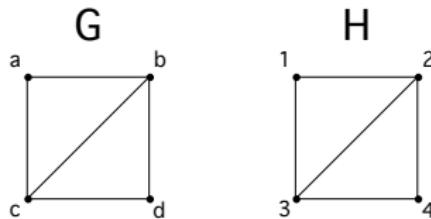


- They are different graphs because their vertex labels are different.
- But they are the same in some sense.
- Formally these graphs are called ‘isomorphic’.

Isomorphisms

An **isomorphism** between two graphs G_1 and G_2 is a bijection $f : V(G_1) \rightarrow V(G_2)$ such that $\{u, v\}$ appears in $E(G_1)$ **exactly as many times as** $\{f(u), f(v)\}$ appears in $E(G_2)$.

Isomorphisms

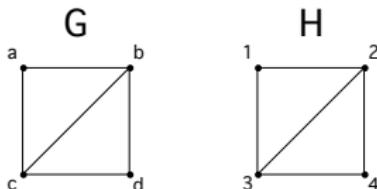


An example of an isomorphism between G and H is the mapping

$$f : V(G) \rightarrow V(H)$$

$$\begin{array}{rcl} a & \mapsto & 1 \\ b & \mapsto & 2 \\ c & \mapsto & 3 \\ d & \mapsto & 4 \end{array}$$

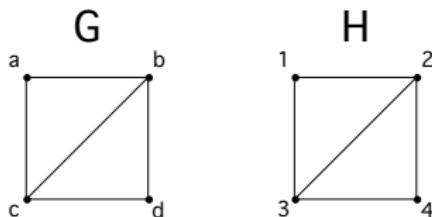
Isomorphisms



The mapping ‘preserves the multiplicity of edges’:

$\{a, a\}$	\mapsto	$\{1, 1\}$	✓ both have multiplicity 0
$\{a, b\}$	\mapsto	$\{1, 2\}$	✓ both have multiplicity 1
$\{a, c\}$	\mapsto	$\{1, 3\}$	✓ both have multiplicity 1
$\{a, d\}$	\mapsto	$\{1, 4\}$	✓ both have multiplicity 0
$\{b, b\}$	\mapsto	$\{2, 2\}$	✓ both have multiplicity 0
$\{b, c\}$	\mapsto	$\{2, 3\}$	✓ both have multiplicity 1
$\{b, d\}$	\mapsto	$\{2, 4\}$	✓ both have multiplicity 1
$\{c, c\}$	\mapsto	$\{3, 3\}$	✓ both have multiplicity 0
$\{c, d\}$	\mapsto	$\{3, 4\}$	✓ both have multiplicity 1
$\{d, d\}$	\mapsto	$\{4, 4\}$	✓ both have multiplicity 0

Isomorphisms



A different example of an isomorphism between G and H is the mapping

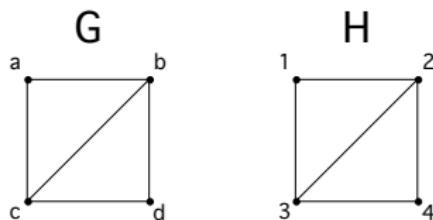
$$g : V(G) \rightarrow V(H)$$

$$\begin{array}{rcl} a & \mapsto & 1 \\ b & \mapsto & 3 \\ c & \mapsto & 2 \\ d & \mapsto & 4 \end{array}$$

Isomorphic Graphs

If there exists an isomorphism between two graphs then the graphs are said to be **isomorphic**.

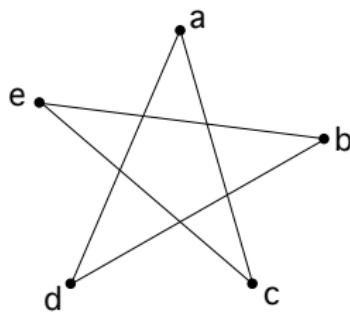
Example: Graphs G and H are isomorphic.



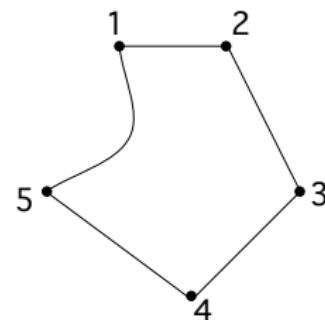
Isomorphic Graphs

- The following graphs pictured are isomorphic.

G_1



G_2



- Can you specify an explicit isomorphism between them?

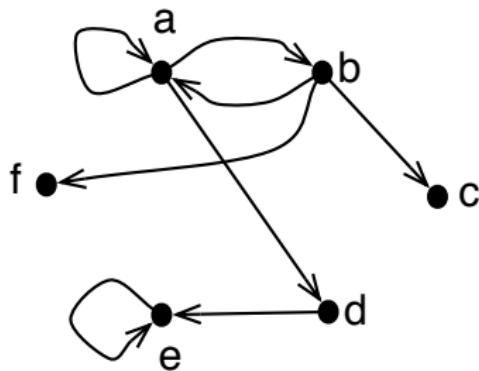
Directed Graphs

Digraphs

Digraphs were introduced in Section A3 in order to represent some relations diagrammatically. Here we look at digraphs in general.

- A **directed graph** (or **digraph**) is the same as a graph except that edges are *ordered* pairs of endpoints.
- Each edge has an **initial vertex** and a **final vertex**.
- Loops are still allowed.
- The edges of a digraph are sometimes called **arcs**.
- In a diagram of a digraph, the direction of an arc is given by an arrow.

Example

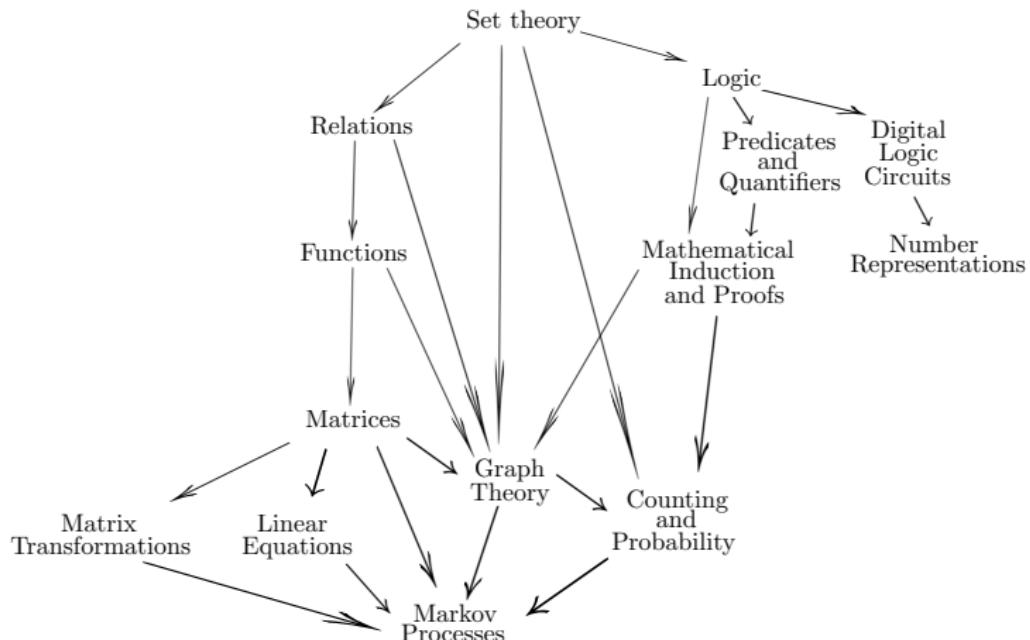


The vertex set and edge set for this graph are:

$$V(G) = \{a, b, c, d, e, f\}$$

$$E(G) = \{(a,a), (a,b), (a,d), (b,a), (b,c), (b,f), (d,e), (e,e)\}$$

An application: Recording Information Dependencies



An arrow from A to B means that B depends upon A in some way.

Foodwebs

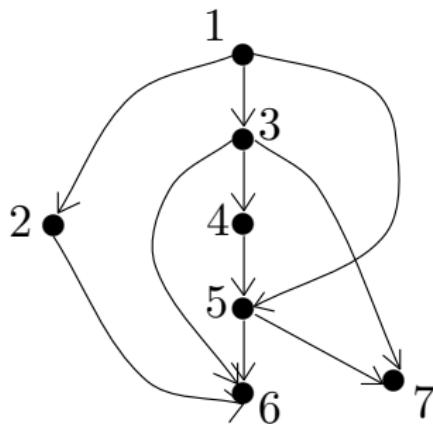
- An application of digraphs in ecology is in describing a *foodweb*.
- The next slide shows a foodweb developed by Parsons and LeBrasseur, as adapted by Cohen, pertaining to the following species in the Strait of Georgia, British Columbia.

KEY SPECIES

1. Juvenile pink salmon
2. P. Minutus
3. Calanus and Euphausiid Burcillia
4. Euphausiid Eggs
5. Euphausiids
6. Chaetoceros Socialis and Debilis
7. Mu-Flagellates

Example

An arrow from i to j means ' i eats j ' :



For example:

- species 1 eats species 2, 3, and 5
- species 4 *only* eats species 5.

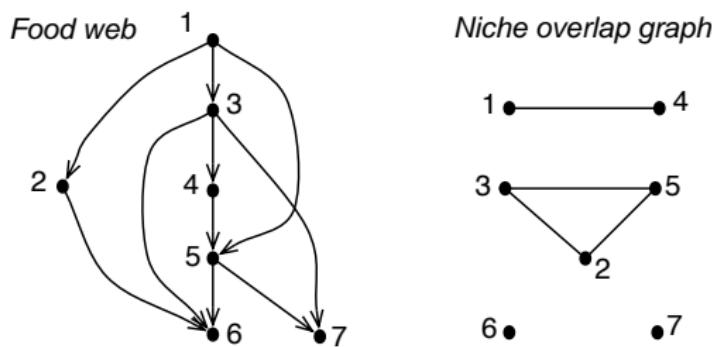
Note: Some foodweb diagrams have their arrows *reversed*:
i.e. an arrow from A to B means ' A is food for B '.

We shall use the first convention unless stated otherwise.

Niche Overlap Graphs

- An application of graphs in ecology is in describing commonalities (or competition) between species in a *Niche Overlap Graph*.
- Each species is represented by a vertex. An undirected edge connects two vertices if and only if the species represented by these vertices compete for food.
- The following niche overlap graph is constructed from the Food Web data of the previous example.

Food Webs and Niche Overlap Graphs



- For example:
species 1 and 4 compete for food (species 5), so are connected by an edge in the niche overlap graph.

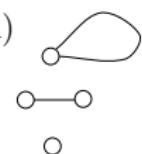
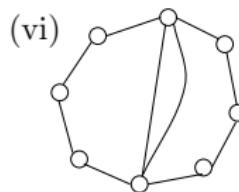
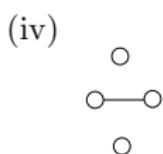
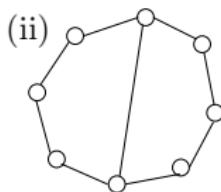
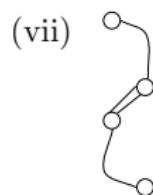
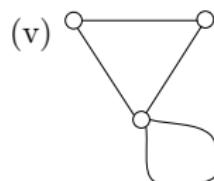
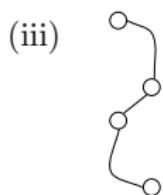
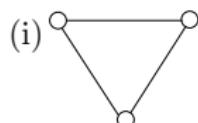
Types of Graphs and Digraphs

Sometimes it is useful to restrict our attention to two types of graphs and digraphs, called:

- **Simple Graphs** and
- **Simple Digraphs**

Simple Graphs

A **simple graph** is a graph that has **no loops** and **no parallel edges**.



Some simple Graphs

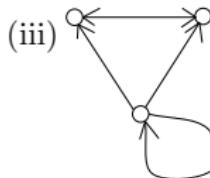
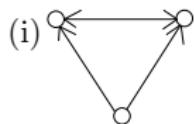
Some non-simple Graphs

Terminology warning

- **Warning:** in graph theory, different authors use the *same words* to mean *different things*.
- For some, what we are calling a **simple graph** is just a graph.
- For some, what we would call a **graph** with parallel edges, is a **multi-graph**.

Simple Digraphs

Similarly, a **simple digraph** is a digraph that has **no loops** and **no parallel edges**.



Some simple Digraphs

Some non-simple Digraphs

Note that:

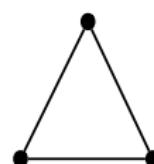
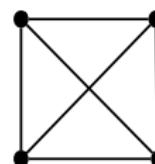
- It is okay to have both (a, b) and (b, a) - these are not parallel;
- It is not okay to have (a, b) twice - those would be parallel.
- We sometimes draw a single edge with an arrow at each end to indicate a pair of edges (a, b) and (b, a) , instead of drawing two distinct lines.

Special simple graphs I: Complete Graphs

A **complete graph** on n vertices is a simple graph in which each pair of distinct vertices are adjacent (i.e. are 'joined' by an edge).

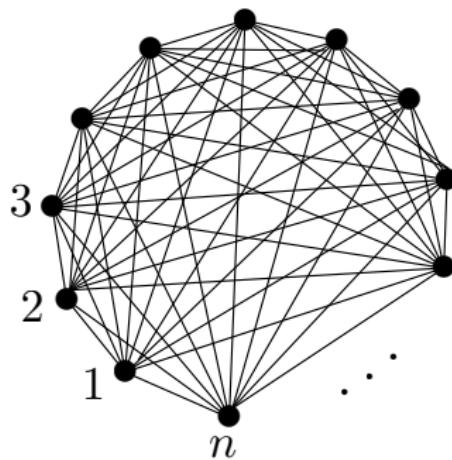
A complete graph on n vertices is denoted by K_n .

Examples:

 K_1  K_2  K_3  K_4

How many edges in K_n ?

How many edges are there in K_n , the complete graph of order n ?

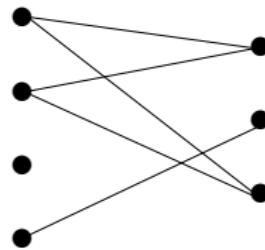


The answer is $\boxed{\binom{n}{2} = \frac{n(n-1)}{2}}$. *Why?*

Special simple graphs II: Bipartite Graphs

A **bipartite** graph is a simple graph whose vertices can be partitioned into two disjoint sets A and B such that **every edge** of the graph connects a vertex in A to a vertex in B .

Example:

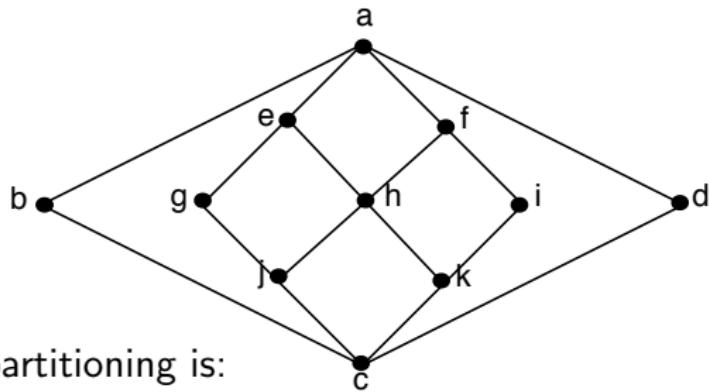


A B

A larger example of a bipartite graph

Sometimes it is not obvious at first glance that a graph is bipartite.

Example: This graph is bipartite:

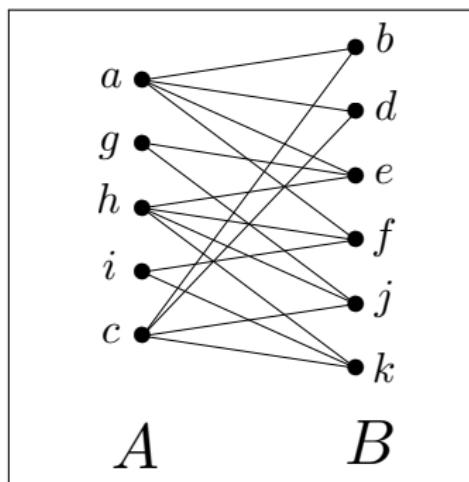


The vertex partitioning is:

$$\begin{aligned} A &= \{a, g, h, i, c\} \\ B &= \{b, d, e, f, j, k\} \end{aligned}$$

Larger example continued

This is the same graph, redrawn.



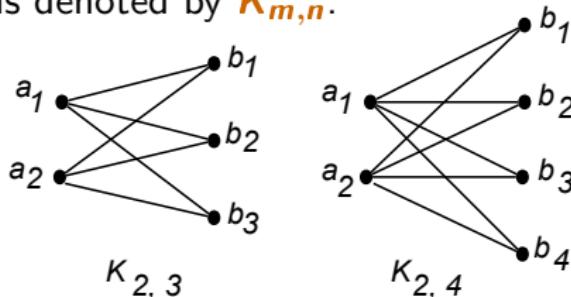
Special simple graphs III: Complete Bipartite Graphs

A **complete bipartite** graph is a simple graph whose vertices may be partitioned into two sets A and B such that:

- $\forall a \in A$ and $\forall b \in B$, the edge $\{a, b\}$ belongs to the graph.
- There are no other edges in the graph.

If A has m vertices and B has n vertices, the complete bipartite graph on A and B is denoted by $K_{m,n}$.

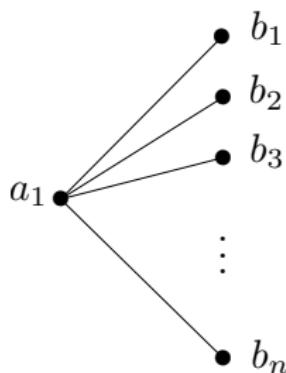
Examples:



Caution: The name “Complete Bipartite Graph” is misleading. Except for $K_{1,1}$, such graphs are **not complete graphs**. The adjective ‘complete’ is qualifies ‘bipartite’, not ‘graph’.

How many edges in $K_{1,n}$?

- How many edges in the complete bipartite graph $K_{1,n}$?



How many edges in $K_{m,n}$?

- *How many edges in $K_{1,n}$?*

- *How many edges in $K_{2,n}$?*

⋮

- *How many edges in $K_{m,n}$?*

Subgraphs

- A **subgraph**, S , of a graph G , is a graph whose vertices are a subset of $V(G)$ and whose edges are a subset of $E(G)$, i.e.

$$V(S) \subseteq V(G)$$

$$E(S) \subseteq E(G)$$

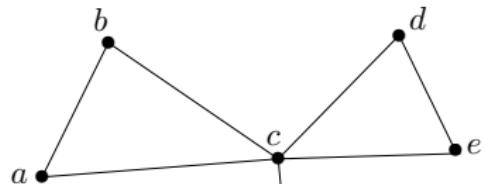
- Since S is a graph, if the edge

$$\{a, b\} \in E(S),$$

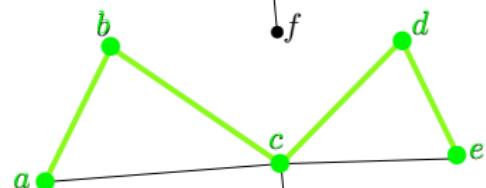
we require its endpoints to be in $V(S)$, i.e. $a, b \in V(S)$.

A subgraph example:

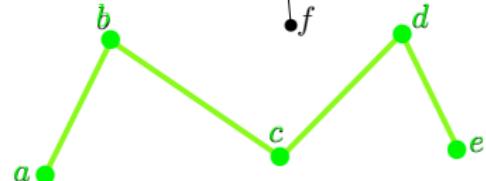
Let G be the graph:



Select some edges and vertices:

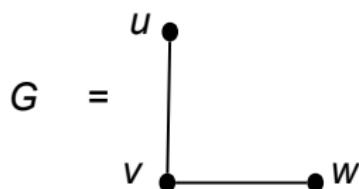


Now S is a subgraph of G :

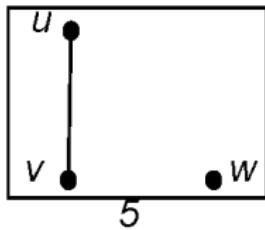
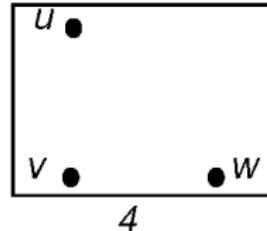
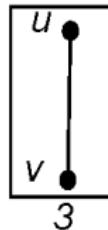
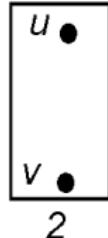


Another subgraphs example:

Let

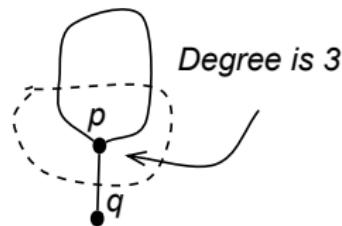
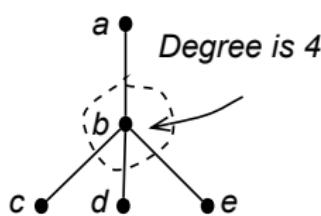


The following five graphs (numbered 1-5) are each subgraphs of G :



Degree of a vertex

- The **degree** of a vertex is the number of edges incident on it (but with each loop counted twice — once for each ‘end’).



- Let v be a vertex and let k be its degree. We write $\deg(v)=k$.
- So in the above examples $\deg(b) = 4$ and $\deg(p) = 3$.

Warning: Some authors count a loop as only adding one to a degree, rather than adding two as we do.

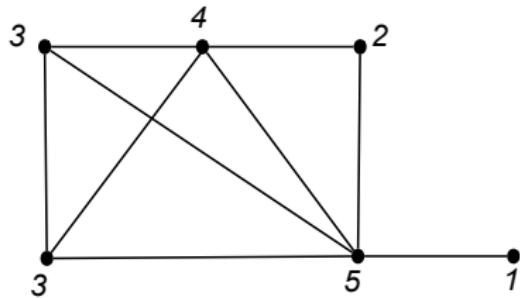
We shall always use the ‘adding two’ version.

Total degree of a graph

The **total degree** of a graph G is the sum of the degrees of all its vertices, $\sum_{v \in V(G)} \deg(v)$. In other words:

Let v_1, v_2, \dots, v_n for $n \in \mathbb{N}$, be the vertices of G . Then
 the total degree of $G = \deg(v_1) + \deg(v_2) + \dots + \deg(v_n)$.

Example: In this graph the degree of each vertex is shown.



The total degree of the graph is $3 + 4 + 2 + 3 + 5 + 1 = 18$.

The Handshake Theorem

The Handshake Theorem: If G is any graph, then the total degree of G equals twice the number of edges of G .

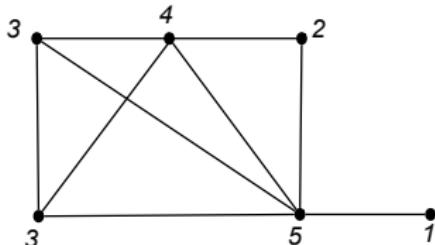
i.e.

$$\sum_{v \in V(G)} \deg(v) = 2|E(G)|$$

Why?

Because when we count degrees we are counting edges, but we count both ends of each edge, hence we count all the edges twice.

Example:



Total degree = 18
Number of edges = 9.

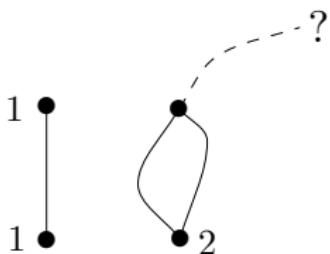
A Corollary

In any graph there is an even number of vertices of odd degree.

How does this corollary follow from the handshake theorem?

Example:

- The set $\{1, 1, 2, 3\}$ cannot possibly be the set of degrees of the vertices of some graph.
- However we try, we always end up with an edge that doesn't have a vertex to connect to:



A useful abbreviation

We often abbreviate

- the graph edge $\{a, b\}$ as ab , and also
- the digraph arc (a, b) as ab .

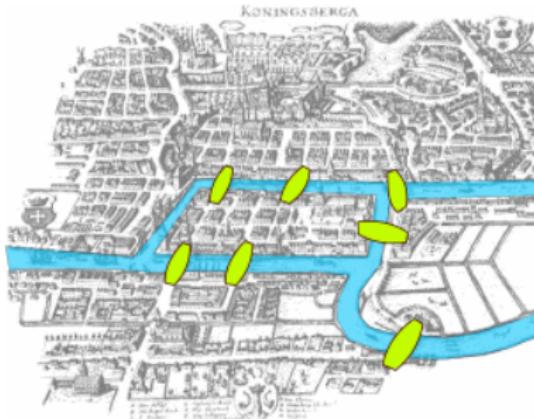
Note that

- for graphs, ab means *the same* as ba , since $\{a, b\} = \{b, a\}$; but
- for digraphs, ab is *different* from ba since $(a, b) \neq (b, a)$.

Walks on Graphs

The bridges of Königsberg

- The city of Königsberg in Prussia was set on both sides of the Pregel River, and included two large islands which were connected to each other and the mainland by seven bridges.
- The problem was to find a walk through the city that would cross each bridge once and only once.

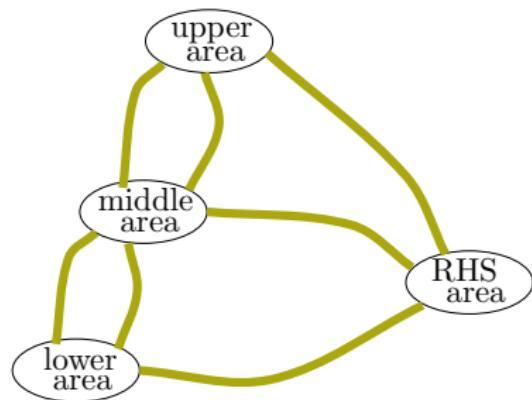
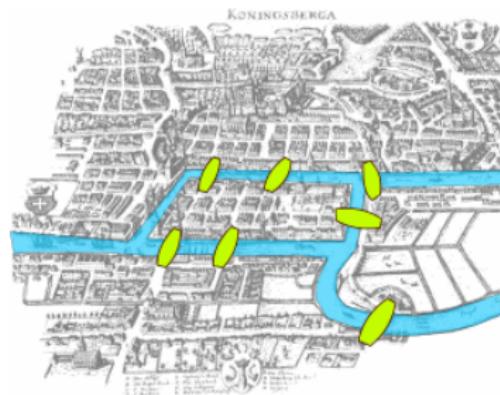


Adapted from:

http://en.wikipedia.org/wiki/Bridges_of_Konigsberg

The bridges of Königsberg

Leonard Euler realized that the task can be modeled as a problem in graph theory, which he invented for the purpose.



Adapted from:

http://en.wikipedia.org/wiki/Bridges_of_Königsberg

Walks

- A **walk** in a graph is a sequence of vertices alternating with edges:

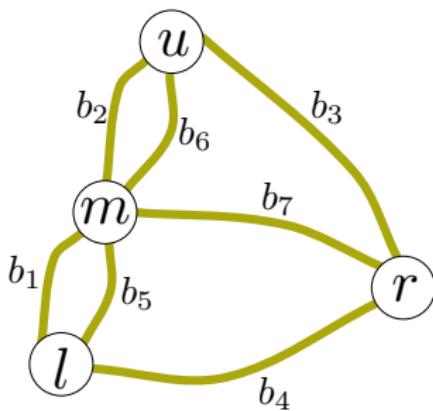
$$v_0, e_1, v_1, e_2, v_2, e_3, \dots, e_n, v_n$$

in which each edge e_k has endpoints v_{k-1} and v_k .

- A walk starting at v_0 and ending at v_n is said to **connect** v_0 to v_n .
- The **length** of the walk is the number of edges listed; length n for the walk above.
- A **trivial walk**, say v_0 , contains no edges; hence has length 0.

Walks on the Königsberg Graph

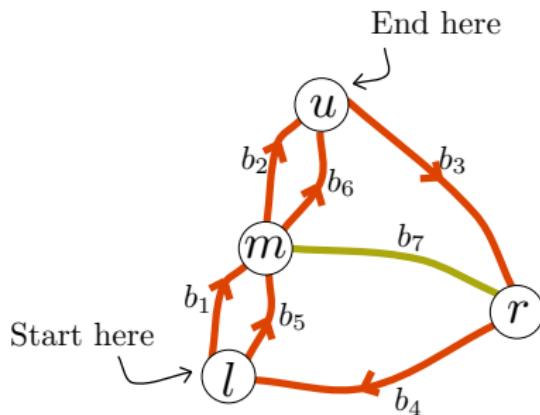
The question becomes, 'Is there a walk on the Königsberg Graph which traverses each edge exactly once?'



Walks on the Königsberg Graph

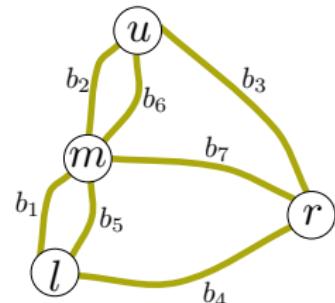
Try starting in the lower part of town, going via bridge b_1 to the middle island, then via bridge b_2 to the upper part, then via bridge b_3 to the right-most part; continuing as in the listed walk:

$$l b_1 m b_2 u b_3 r b_4 l b_5 m b_6 u$$



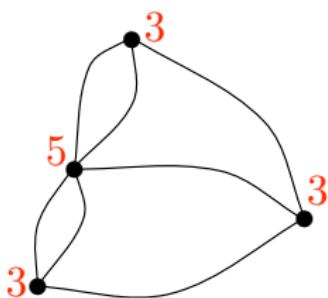
Walks on the Königsberg Graph

- That attempt didn't work, because there is no unused edge to exit u from on the second visit.
- For brevity we'll leave out vertices and just list edges. Try
 - $b_1 b_2 b_3 b_4 b_5 b_6 \dots$ stuck at u
 - $b_1 b_2 b_3 b_4 b_5 b_7 \dots$ stuck at r
 - $b_2 b_6 b_1 b_5 b_7 b_4 \dots$ stuck at l
 - $b_1 b_2 b_6 b_5 b_4 b_7 \dots$ stuck at m



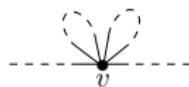
Impossibility of Königsberg Bridge Walk

- The kind of walk needed to meet the criteria for the Königsberg Bridge problem - a walk which traverses all edges and repeats none - came to be called an **Euler Path**.
- Euler showed that no such walk exists on the Königsberg Graph.
- *How? Think about the degrees of the vertices:*



Impossibility of Königsberg Bridge Walk

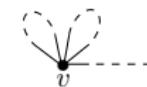
- *There are too many vertices of odd degree in the Königsberg Graph for an Euler path to be possible. Can you prove it?*
- *Hint: can an odd degree vertex occur in the middle of a walk?*



Even degree
vertex
in a walk



Odd degree
vertex
in a walk



Odd degree
vertex
in a walk

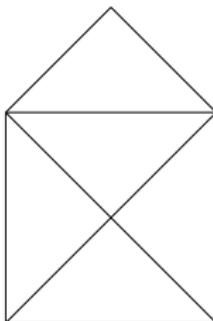
OR

OR

A related puzzle

You may have come across the following puzzle:

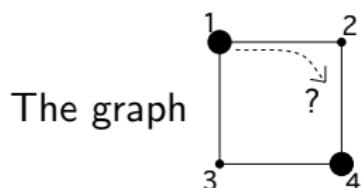
Can you draw the following 'house' diagram without taking your pen off the paper or overwriting edges?



Can you?

Walks, paths and circuits

Counting Walks with Adjacency Matrices



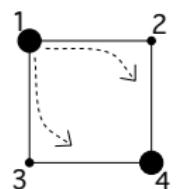
The graph

has adjacency matrix

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}.$$

There are 0 ways to walk from vertex 1 to vertex 4 in a single step.

There are 2 ways to walk from vertex 1 to vertex 4 in a two steps:



$$M^2 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{pmatrix}$$

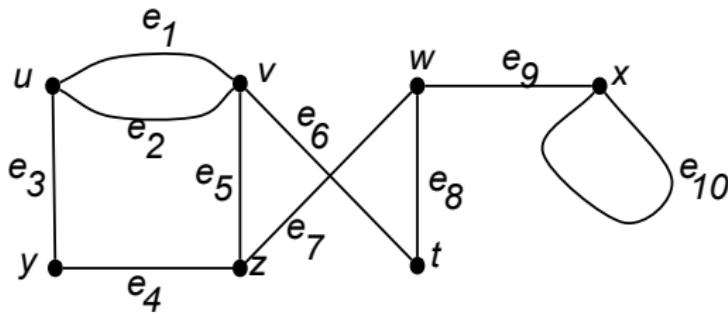
For any graph, the number of ways to walk from vertex i to vertex j in t steps is given in terms of its adjacency matrix M by the $(i,j)^{\text{th}}$ entry of M^t .

Some special kinds of walks

Some properties that a walk on a graph may, or may not, possess are:

- being ‘closed’;
- having no repeated edges;
- having no repeated vertices.

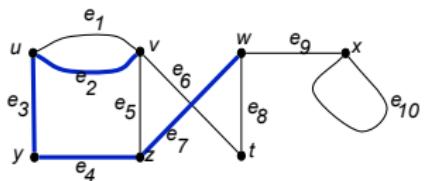
We will now look at each of these potential properties in turn, with examples using the graph G below.



Closed walks

A walk $v_0, e_1, v_1, e_2, \dots, e_n, v_n$ is called a **closed** when $v_0 = v_n$.

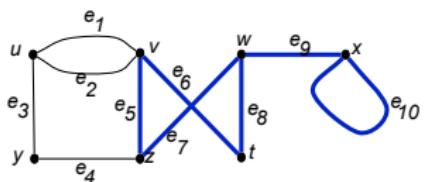
Examples:



The walk

$v \mathbf{e}_2 u \mathbf{e}_3 y \mathbf{e}_4 z \mathbf{e}_7 w$

has length 4 and is **not** closed
because $v = v_0 \neq v_n = v_4 = w$.



The walk

$v \mathbf{e}_6 t \mathbf{e}_8 w \mathbf{e}_9 x \mathbf{e}_{10} x \mathbf{e}_9 w \mathbf{e}_7 z \mathbf{e}_5 v$

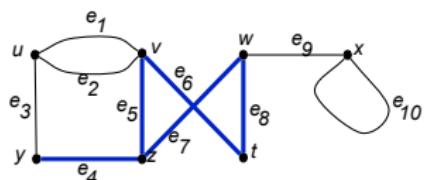
has length 7 and **is** closed
because $v = v_0 = v_n = v_7 = v$.

Paths

A **path** is a walk that does not repeat any edge.

A **simple path** is a path which does not repeat any vertex — except the first and last if the path is closed.

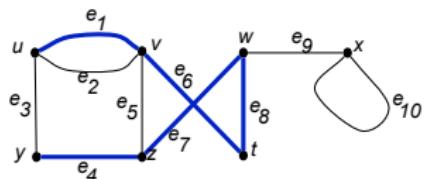
Examples:



The walk

$y \ e_4 \ z \ e_7 \ w \ e_8 \ t \ e_6 \ v \ e_5 \ z$

has length 5 and is a **path** because the five edges are all different, but is **not simple** because $z = v_1 = v_5$



The walk

$y \ e_4 \ z \ e_7 \ w \ e_8 \ t \ e_6 \ v \ e_1 \ u$

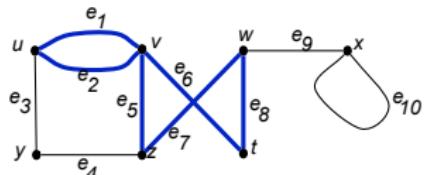
has length 5 and is a **simple path** because the six vertices are all different.

Circuits

A **circuit** is a closed path.

A **simple circuit** is a simple closed path.

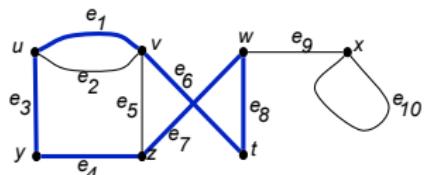
Examples:



The walk

$z \mathbf{e}_7 w \mathbf{e}_8 t \mathbf{e}_6 v \mathbf{e}_1 u \mathbf{e}_2 v \mathbf{e}_5 z$

has length 6 and is a **circuit** as it is closed with all different edges, but is **not simple** because $v = v_3 = v_5$.



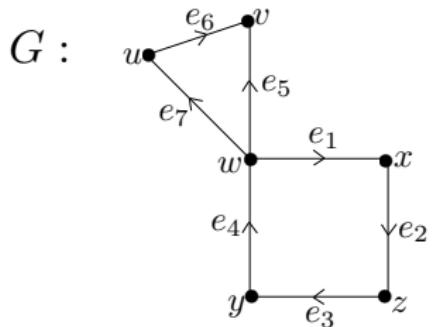
The walk

$z \mathbf{e}_7 w \mathbf{e}_8 t \mathbf{e}_6 v \mathbf{e}_1 u \mathbf{e}_3 y \mathbf{e}_4 z$

has length 6 and is a **simple circuit** as it is closed without repeated vertices except the first and last $z = v_0 = v_6$.

Walks on digraphs

- A walk in a **digraph** is sometimes called a **directed walk**.
- In a directed walk, the sequence of edges alone always uniquely determines the path. Examples:



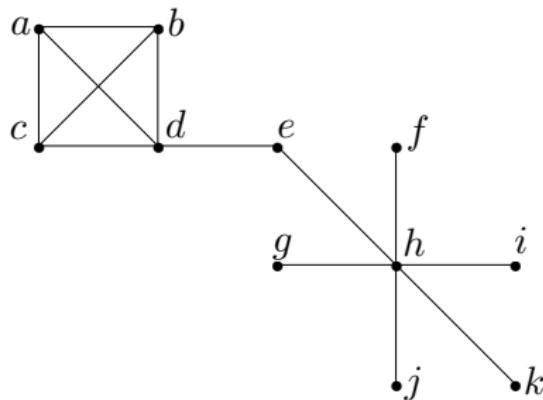
- $e_1e_2e_3e_4e_7e_6$ is a directed walk in G
- $e_1e_2e_3e_4$ is a closed directed walk in G
- $\cancel{e_7e_6e_5}$ is NOT a directed walk in G

- Note that for a simple graph or digraph (with no loops and no parallel edges), **any** walk is uniquely determined by its sequence of vertices.

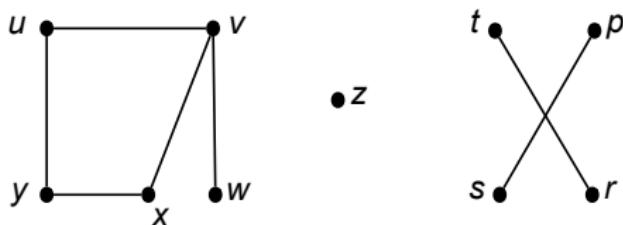
Connected Graphs

- A graph is **connected** if every pair of vertices can be connected by a walk (and therefore by a path).
- A **component** of a graph is a maximal connected subgraph.

Here is an example of a connected graph:



Example

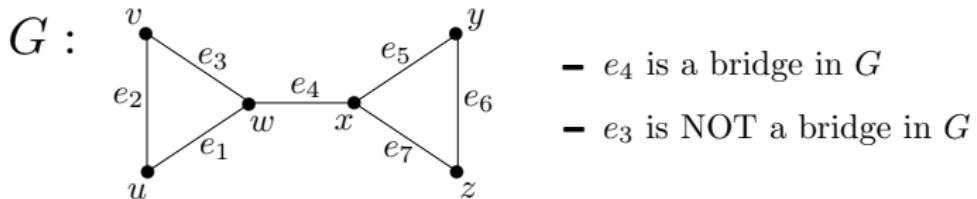


The graph above is **not connected** and has **4 components**:

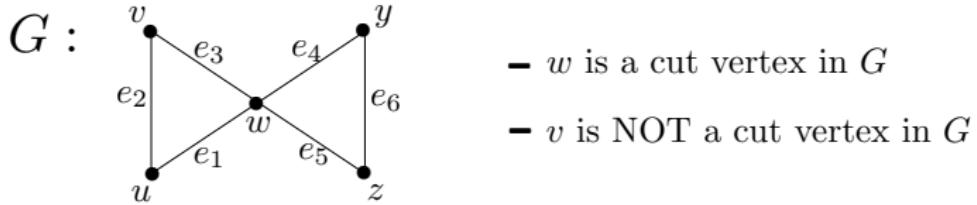
1. the subgraph with vertex set $\{u, v, w, x, y\}$ and edge set $\{\{u, v\}, \{u, y\}, \{v, w\}, \{v, x\}, \{x, y\}\}$,
2. the subgraph with vertex z and no edges,
3. the subgraph with two vertices t, r and one edge $\{t, r\}$,
4. the subgraph with two vertices p, s and one edge $\{p, s\}$.

Bridges and Cut Vertices

- A **bridge** in a connected graph is an edge which on erasure disconnects the graph. Examples:

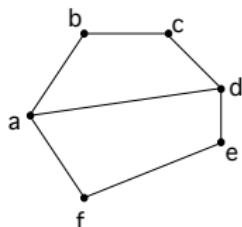


- A **cut vertex** in a connected graph is a vertex which on erasure disconnects the graph. Examples:



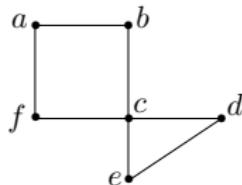
Euler Paths and Circuits

- An **Euler path** for a graph is a path passing through every edge (hence every vertex). (Recall the Königsberg Bridge problem.) Example:



This graph has an Euler path: *abcdafed*.

- An **Euler circuit** for a graph is a circuit passing through every edge (hence every vertex). Example:



This graph has an Euler circuit: *abcedcfa*.

Note: By convention, an Euler path must be open, i.e not a circuit.

When do Euler Paths and Circuits exist?

- **Theorem:** A connected graph has an Euler circuit if and only if each of its vertices has even degree.

In this case we will give an algorithm for finding an Euler circuit.

- **Corollary:** A connected graph has an Euler path if and only if it has exactly two vertices of odd degree.

The algorithm easily adapts to this case.

Issues regarding finding an Euler Circuit in a graph G

- By the theorem, G must be connected and have all its vertices of even degree. Given this, an Euler circuit must exist.
- On a circuit, you can start anywhere and return after completing the circuit. So start the search at any vertex in G .
- As you walk through G mark each edge you use; it cannot be used again. Imagine the edge has been erased, so that graph is 'reduced'.
- After 'erasing' an edge, the start vertex and the 'current' vertex have odd degree in the reduced graph, while all other vertices remain with even degree. You then seek an Euler **path** (in the reduced graph) from the current vertex to the start vertex. By the corollary, such a path exists provided the reduced graph is connected.
- So choose each edge so that the reduced graph is still connected.
- Always leave an edge to return to the start vertex as the last step.

Fleury's Algorithm for finding Euler Circuits

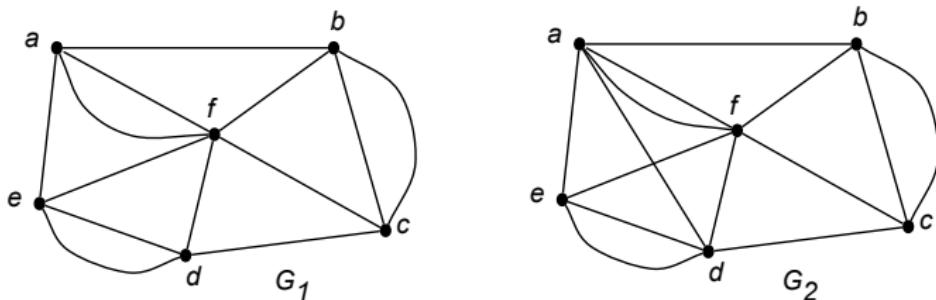
Let G be a connected graph with all vertices of even degree.

The Algorithm

1. Pick any vertex of G as a starting point.
2. From that vertex **choose any edge to traverse that does not cross a bridge of the current reduced graph, unless there is no other choice.** ("No choice" will only happen when the vertex has degree 1 in the current reduced graph. The new reduced graph will then omit that vertex and so remain connected.)
3. Mark the edge (e.g. darken it) as a reminder not to traverse it again. Treat the edge as erased, so reducing the graph.
4. Travel that edge, coming to the next vertex.
5. Repeat steps 2 - 4 until all edges have been traversed, and you are back to the starting vertex.

Candidate Graphs for Fleury's Algorithm

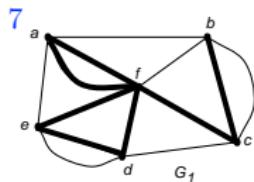
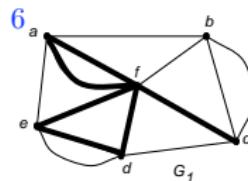
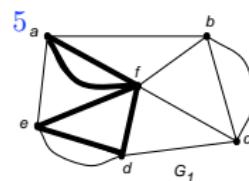
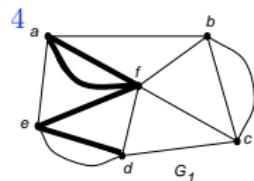
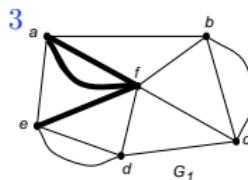
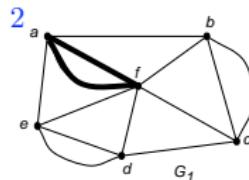
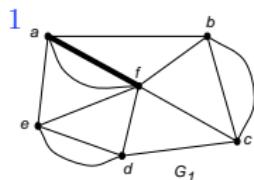
The graph G_1 below satisfies the criterion that all vertices have even degree, so it contains an Euler circuit and Fleury's algorithm can be used to find that circuit.



The graph G_2 has two vertices of odd degree. Fleury's algorithm can be modified to find an Euler path in this graph. The only modification needed is that the first vertex must be one of the vertices of odd degree.

Fleury's Algorithm example in pictures

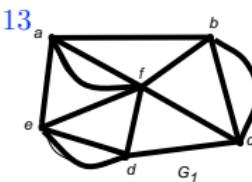
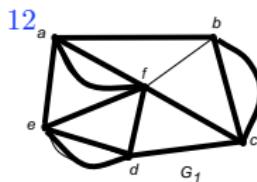
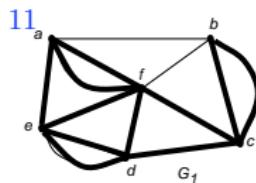
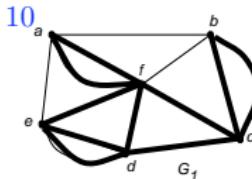
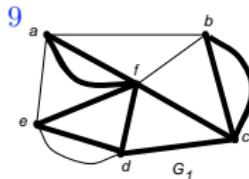
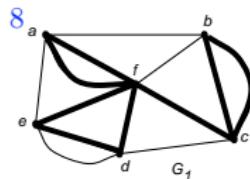
Start at f (just because we feel like it!)



Notice that from here
we MAY NOT step to f
but either a or c is allowed

Fleury's Algorithm example in pictures

At step 9 we're forced to go to d ; and then all steps are forced.



The final path is $fafedfcbcdeabf$.

The path is now closed, providing the Euler circuit we sought.

(As mentioned earlier, we do not call this path an Euler *path*, because, by convention, Euler paths are not closed.)

Fleury's Algorithm example discussion

- Note that if we had started at some other vertex, or made different choices along the way, Fleury's algorithm would have given a different Euler circuit.
- In any implementation, we never have to back-track, so the algorithm is quite fast; as are some other algorithms to solve this problem.
- By contrast, the following problem – that of finding a *Hamilton* path or circuit – has no known ‘fast’ algorithm.

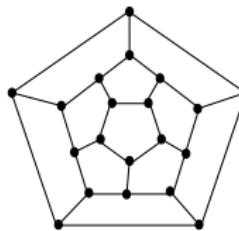
Hamilton Paths and Circuits

- A **Hamilton path** for a graph is a *simple* path which passes through every vertex.
- A **Hamilton circuit** for a graph is a *simple* circuit which passes through every vertex.
- Note that a Hamilton path (respectively circuit)
 - does not have to use every edge,
 - may use an edge at most once by the definition of path (respectively circuit),
 - must use each vertex exactly once by the definition of simple.
- A graph is called **Hamiltonian** if and only if it possesses a Hamilton circuit.

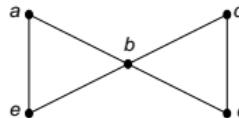
Note: By convention, a Hamilton path must be open, *i.e* not a circuit.

Graphs and Hamilton Paths / Circuits

- This graph has a Hamilton Circuit. *Can you find one?*



- This graph has no Hamilton circuit. *Why not?*



Types of Walks on Graphs – Summary

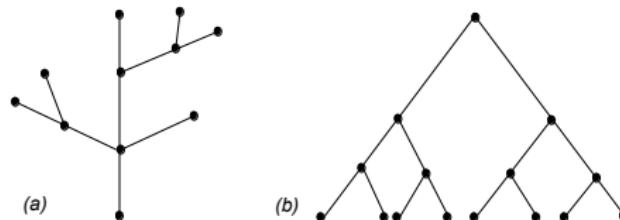
For a walk $v_0, e_1, v_1, e_2, \dots, e_n, v_n$ on a graph G :

Properties					
Name:	closed	—	Euler	simple	Hamilton
Description:	—	no repeated edges	uses all edges	no repeated vertices	uses all vertices
Requirement:	$v_0 = v_n$	$\begin{array}{c} i \neq j \\ \implies \\ e_i \neq e_j \end{array}$	$\begin{array}{c} \forall e \in E(G) \\ \exists i \ e_i = e \end{array}$	$\begin{array}{c} i \neq j \\ \implies \\ v_i \neq v_j \\ (v_0 = v_n \text{ OK}) \end{array}$	$\begin{array}{c} \forall v \in V(G) \\ \exists i \ v_i = v \end{array}$
path		✓			
simple path		✓		✓	
Euler path		✓	✓		
Hamilton path		✓		✓	✓
closed walk	✓				
circuit	✓	✓			
simple circuit	✓	✓		✓	
Euler circuit	✓	✓	✓		
Hamilton circuit	✓	✓		✓	✓

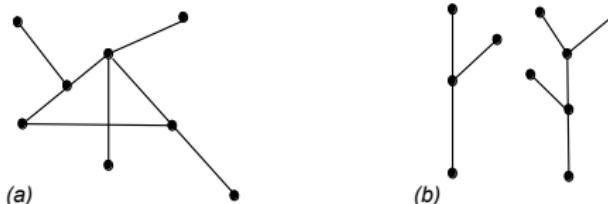
Trees

Trees

- A **tree** is a connected graph with no circuits other than the trivial ones. Examples:



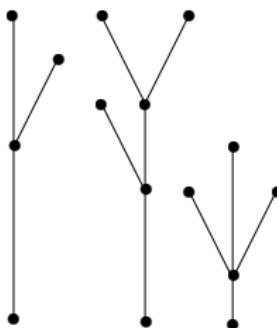
- Examples of **non-trees**



- Example (a) contains a circuit, so is not a tree.
- Example (b) is not a tree. *Why not?*

Forests

- A **forest** is a disjoint collection of trees. Example:



Trees as Models

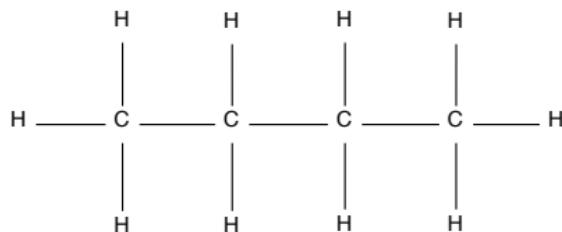
- Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, psychology, management, linguistics and many others. (See Epp, 10.5[4ed], 10.4[5ed]).
- On your computer, the folder and file directory has a tree structure.
- Trees are used in linguistics to describe grammatical relationships.
- In botany and zoology, taxonomy of species is entirely based on a tree structure.
- In evolutionary science we have 'the tree of life'.
- Organic chemistry uses graphs and trees for models of molecules. We look at this next.

Structure of Hydrocarbon Molecules

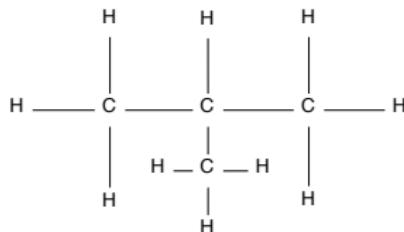
- Graphs can be used to represent molecules, where atoms are represented by vertices and the bonds between them by edges.
- **Hydrocarbon** molecules are composed of carbon and hydrogen only.
- Each carbon atom normally forms 4 bonds with other atoms; we shall only consider this case.
- Each hydrogen atom forms just one bond with another atom.
- In the representation of such molecules we use C to represent a carbon atom and H to represent a hydrogen atom. These will be used instead of dots for the vertices.

Examples of Hydrocarbon Graphs

- Butane:



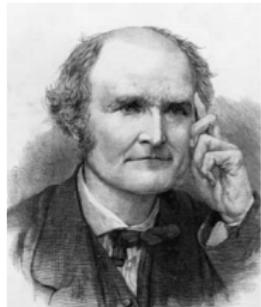
- Isobutane:



Graphs and Isomers

- Butane and Isobutane graphs each have 4 carbon atoms and 10 hydrogen atoms, but the configuration of the atoms is different.
- They have the same chemical formula, C_4H_{10} , but different chemical bonds and are called **isomers**.
- **Saturated hydrocarbon** molecules contain the maximum number of hydrogen atoms for a given number of carbon atoms.

Some History of Trees



Arthur Cayley 1821 - 1895

- The English mathematician Arthur Cayley developed some of the theory of trees when he was trying to enumerate the isomers of the saturated hydrocarbons of the form C_nH_{2n+2} .
- Cayley showed that a saturated hydrocarbon molecule must have this formula.
- You will explore a proof of this formula in Workshops next week.

Characterising Trees

Theorem: Let T be a graph with n vertices. The following statements are logically equivalent:

- (i) T is a tree (it is connected and has no non-trivial circuits).
- (ii) T has no simple circuits and $n - 1$ edges.
- (iii) T is connected and has $n - 1$ edges.
- (iv) T is connected and every edge is a bridge.
- (v) Any two vertices of T are connected by exactly one simple path.
- (vi) T contains no non-trivial circuits, but the addition of any new edge (connecting an existing pair of vertices) creates a simple circuit.

Proving and Using the Theorem

- The previous Theorem collects together several different ways of characterizing a tree. In different contexts, each of the different descriptions are useful.
- To prove the Theorem, we should show that any statement in the list is derivable from any other statement. One way to do this is to show the chain of implications:

$$(i) \Rightarrow (ii) \Rightarrow (iii) \Rightarrow (iv) \Rightarrow (v) \Rightarrow (vi) \Rightarrow (i).$$

Proving the Theorem

- *Can you prove that $(i) \Rightarrow (j)$ for any of the statements in the Tree-Characterising Theorem?*
- Some results that you might find helpful are ...

Lemma A: Any tree that has more than one vertex has at least one vertex of degree 1.

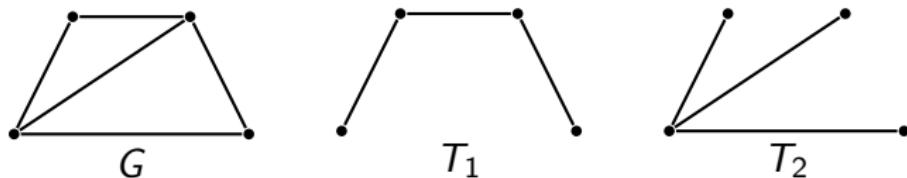
Lemma B: If G is any connected graph, C is a non-trivial circuit in G , and one of the edges of C is removed, then the subgraph that remains is connected.

Applications of graphs, paths and trees

- Many problems can be modelled with paths formed by travelling along the edges of graphs.
- For instance, the problem of determining whether a message can be sent between two computers using intermediate links can be studied with a graph model.
- Problems of efficiently planning routes for mail delivery, garbage pickup, diagnostics in computer networks, and so on, can be solved using models that involve graphs.
- A subgraph of a connected graph that provides a unique path between any two vertices is a *spanning tree*. These trees have applications in many fields, including engineering.

Spanning trees

- **Definition:** A **spanning tree** for a graph G is a subgraph of G which is a tree and contains all the vertices of G .
- Spanning trees need not be unique. That is, a given graph can have a number of different spanning trees.
- Example: Two distinct spanning trees, T_1 and T_2 , for a graph G are shown below:



How many more spanning trees can you find for G ?

Which graphs have spanning trees?

Theorem:

1. Every connected graph has a spanning tree.

Can you prove this?

Hint: Use Lemma B.

2. Any 2 spanning trees for a graph have the same number of edges.

Why?

Hint: See (ii) or (iii) of the tree characterisation theorem.

A method to make a spanning tree

Let G be a connected graph with n vertices.

The Algorithm

1. Initialize T to be the vertices of G but no edges.
2. Pick an edge.
3. Add the chosen edge to T if and only if it does not make a circuit in T .
4. Repeat steps (2) and (3) until T has $n - 1$ edges, then stop.

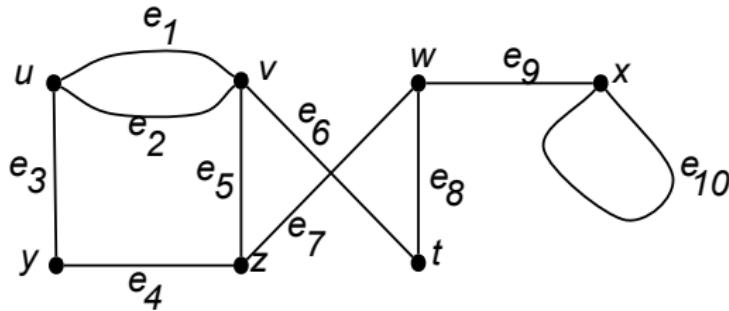
The above description of the algorithm is intended for 'pen-and-paper' work.

For computer implementation it is necessary to **also**:

- at step 1 initialize a 'pool of potential edges' P to $E(G)$,
- at step 2 ensure the picked edge e comes from P and
- after step 3 remove e from P (whether it contributes to T or not).

Building a spanning tree: example

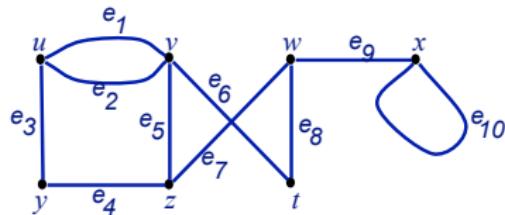
To demonstrate the spanning tree algorithm we will use a graph we have seen before:



Initialize T to

- vertex set $V(T) = \{u, v, w, x, y, z, t\}$,
- edge set $E(T) = \{\}$:

Building a spanning tree for



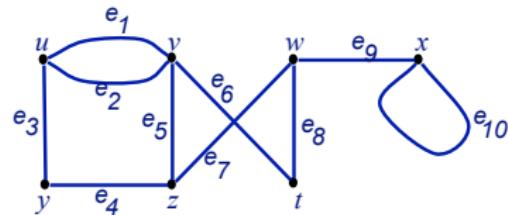
$$E(T) = \{\}: \quad T : \quad \begin{matrix} u \\ \bullet \end{matrix} \quad \begin{matrix} v \\ \bullet \end{matrix} \quad \begin{matrix} w \\ \bullet \end{matrix} \quad \begin{matrix} x \\ \bullet \end{matrix}$$

$$\begin{matrix} y \\ \bullet \end{matrix} \quad \begin{matrix} z \\ \bullet \end{matrix} \quad \begin{matrix} t \\ \bullet \end{matrix}$$

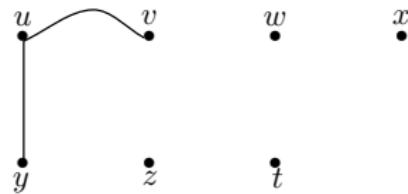
$$E(T) = \{e_1\} \quad T : \quad \begin{matrix} u \\ \bullet \end{matrix} \begin{matrix} v \\ \bullet \end{matrix} \quad \begin{matrix} w \\ \bullet \end{matrix} \quad \begin{matrix} x \\ \bullet \end{matrix}$$

$$\begin{matrix} y \\ \bullet \end{matrix} \quad \begin{matrix} z \\ \bullet \end{matrix} \quad \begin{matrix} t \\ \bullet \end{matrix}$$

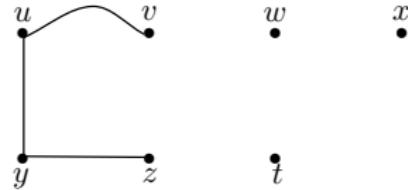
Building a spanning tree for



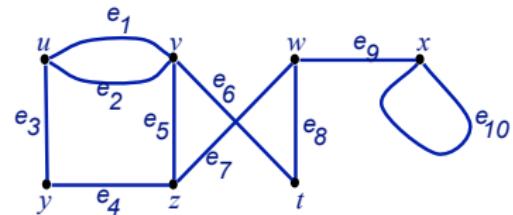
$$E(T) = \{e_1, e_3\} : \quad T :$$



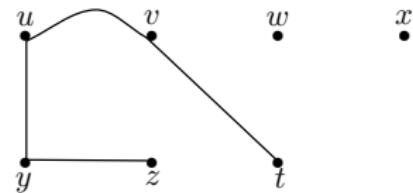
$$E(T) = \{e_1, e_3, e_4\} \quad T :$$



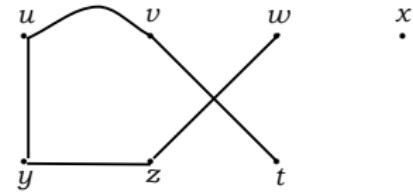
Building a spanning tree for



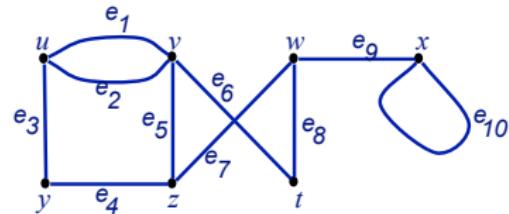
$$E(T) = \{e_1, e_3, e_4, e_6\} : \quad T :$$



$$E(T) = \{e_1, e_3, e_4, e_6, e_7\} \quad T :$$

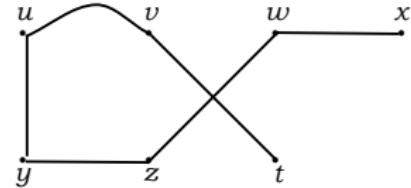


Completed spanning tree for



$$E(T) = \{e_1, e_3, e_4, e_6, e_7, e_9\}:$$

$T :$



Our tree now has $7 - 1 = 6$ edges, so we are done (as you can see).

As it happens, this tree has no branching and so contains a path of length 6. That just results from our order of choosing edges.

Can you make a spanning tree in which the longest path has length 4? Length 3?

END OF SECTION D1

D2. Weighted Graphs

Notes by Malcolm Brooks,
partly inspired by notes of Pierre Portal.

Text Reference (Epp) 3ed: Chapter 11
 4ed: Chapter 10
 5ed: Chapter 10

Some of the work in this section is not covered in our text by Epp.
I have based some examples on ones from:
Kolman, Busby & Ross *Discrete Mathematical Structures*
Johnsonbaugh *Discrete Mathematics*

Weighted graphs

- In some applications of graphs there is a non-negative number associated with each edge, known as the **weight** of the edge.
- So a **weighted graph** is a graph G together with a **weight function** $\text{weight} : E(G) \rightarrow \mathbb{Q}_+$.
- Examples include:
 - **Airline networks:** Vertices are airports used; edges are services or potential services between airports; weights are air miles or cost of flights.
 - **Fibre telecommunications:** Vertices are primary routing stations; there are edges between every pair of vertices (a complete graph); weights are the costs of laying fibre between stations.
 - **The internet:** Vertices are internet nodes; edges are all direct connections between nodes; weights are times (in milliseconds) for a packet to travel across a connection.

Problems on weighted graphs

We define the **(total) weight** of a subgraph S of a weighted graph G to be the sum of the weights of all its edges.

There are several well-studied optimisation problems relating to subgraphs of weighted graphs. We will look at just three:

Minimal spanning tree:

Find a spanning tree of least possible total weight.

Travelling salesman problem:

Find a Hamilton circuit of least possible total weight.

Shortest path:

Find a path between two given vertices that has least possible total weight.

We will also look at a different kind of problem on a weighted **directed** graph: **Maximal Flow**. Details later.

Kruskal's algorithm for minimal spanning tree

Kruskal's algorithm is an obvious modification to the spanning-tree-finding algorithm we discussed previously. We simply choose the 'cheapest' possible edge at each step:

Input: Weighted connected graph G with n vertices.

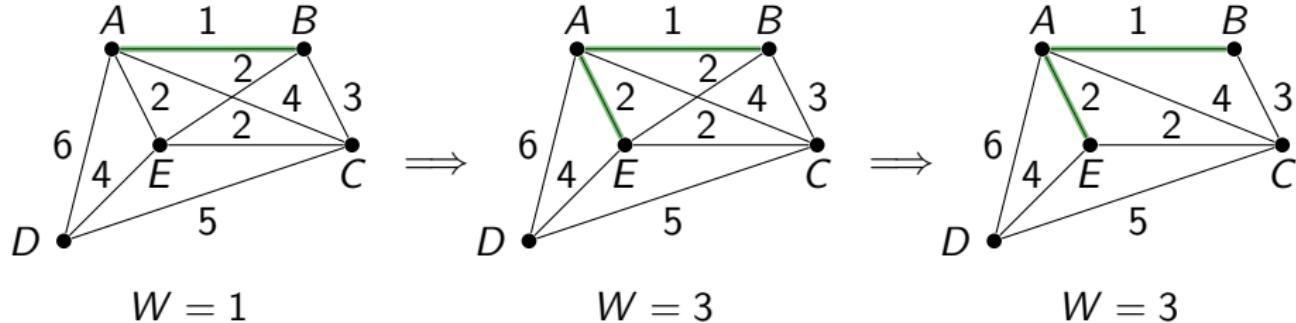
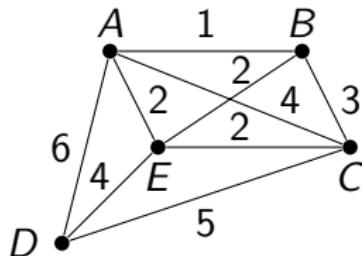
Output: Minimal spanning tree T for G .

Total weight W of this tree.

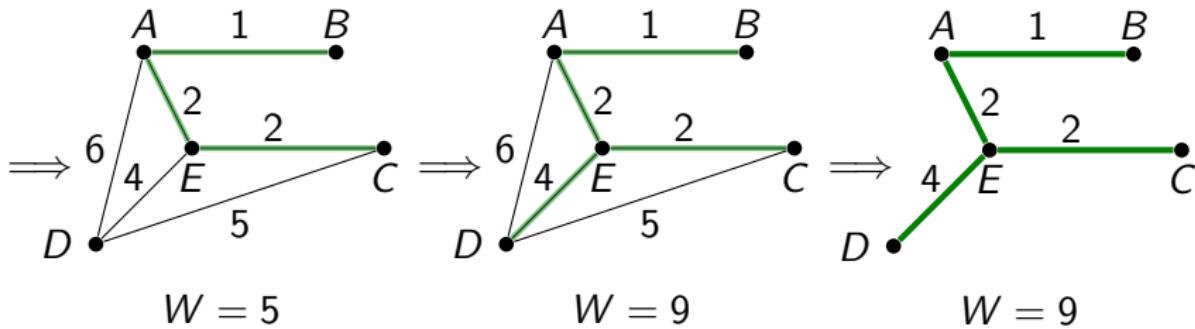
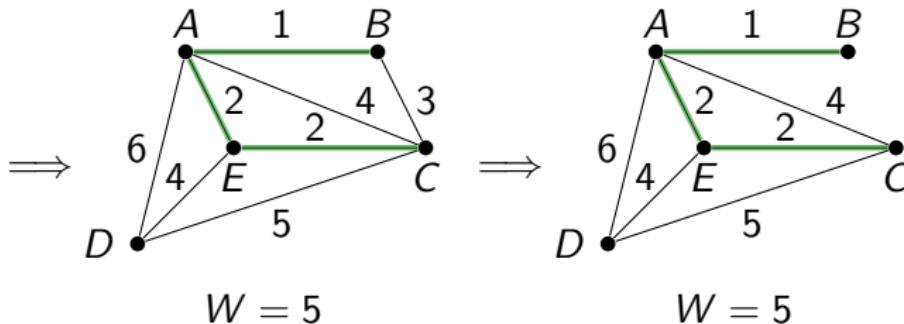
- Method:**
1. Initialise T to have all the vertices of G but no edges. Initialise W to 0.
 2. From the edges of currently in G pick one, e , of least weight and remove it from G .
 3. If adding e to T does not create a circuit in T , add e to T and add $\text{weight}(e)$ to W .
 4. Repeat steps 2 and 3 until T has $n - 1$ edges.

Example: Applying Kruskal's algorithm

Find a minimal spanning tree for this weighted graph:



Example: Applying Kruskal's algorithm (cont.)



(The generated spanning tree)

'Greedy' algorithms

- A 'greedy' algorithm is an algorithm for an optimisation problem that attempts to find a globally optimal solution by finding a locally optimal solution at each step.
- Kruskal's algorithm, that we examined for solving the minimal spanning tree problem, is an example of a greedy algorithm because:
- Kruskal's algorithm attempts to find a spanning tree of **least** possible *total* weight by, at each step, adding an edge of **least** possible (*individual*) weight (from amongst all unused edges that would not create a circuit).
- Kruskal's algorithm always succeeds! (Non-obvious theorem omitted)

That is, it always finds a minimal spanning tree, given any weighted connected (finite) graph.

The ‘Travelling salesman’ problem

- This problem originates from a time when all salespeople were men !!
- The salesman needs to visit n towns on a shortest possible ‘circular tour’.
- Given: a table of distances between every pair of towns.
- **Model:** Graph K_n with towns as vertices and edges weighted by the inter-town distances.

Find a Hamilton circuit of minimum possible total weight.

The ‘Nearest Neighbour’ algorithm (for the travelling salesman problem)

Input: Weighted complete graph G with n vertices.

Output: Hamilton circuit for G as a list L of vertices.
Total weight W of this circuit.

- Method:**
1. Initialise L to the empty list, W to 0 and index i to 1
 2. Choose any vertex and append it to L as $L(1)$.
 3. From all vertices in G but not in L , choose a vertex v such that $\text{weight}(L(i), v)$ is as small as possible. (v is a ‘nearest neighbour’ to $L(i)$).
 4. Add $\text{weight}(L(i), v)$ to W . Increment i by 1. Append v to L as $L(i)$.
 5. Repeat steps 3 and 4 until $i = n$.
 6. Add $\text{weight}(L(n), L(1))$ to W . Append $L(1)$ to L as $L(n + 1)$.

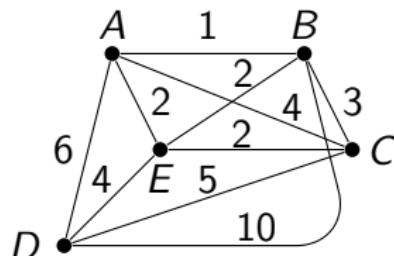
Greedy algorithms don't always succeed

- The Nearest Neighbour algorithm is another example of a greedy algorithm because at each step it looks for the 'best way out', ignoring any possible disadvantages later on.
- But the Nearest Neighbour algorithm often fails to find the shortest Hamilton circuit.
- Greed doesn't always pay !!
- In fact, no efficient successful algorithm for the travelling salesman problem is known at this time. Finding one, or proving that none exists, is a major outstanding problem in mathematics.

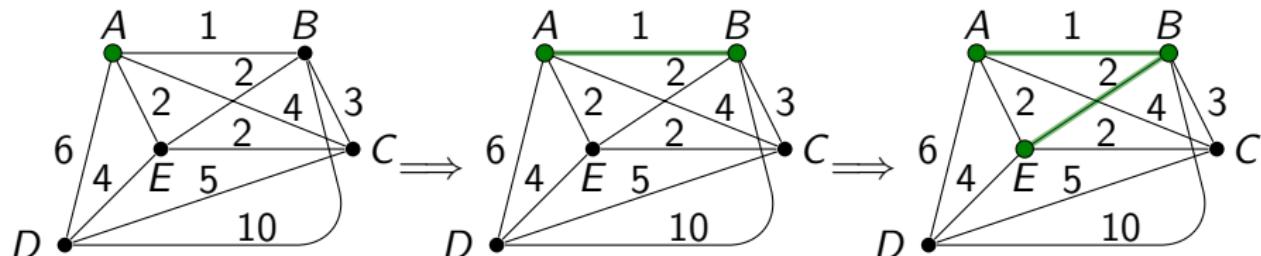
Example: Applying the Nearest Neighbour algorithm

Find a minimal Hamilton circuit (tour) for this weighted graph:

Note: This graph is as for the minimal spanning tree example but with the addition of an edge BD to make it complete.



Let's start at A (we can start anywhere):

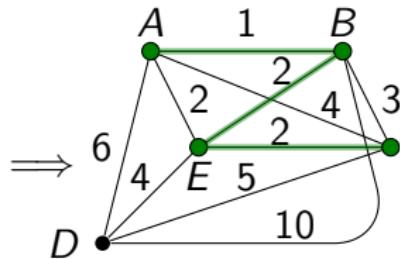


$$L(1) = A, \quad W = 0$$

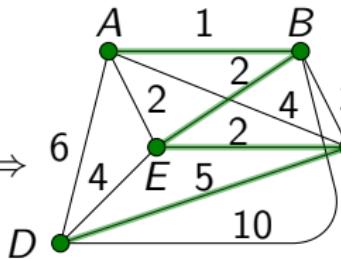
$$L(2) = B, \quad W = 1$$

$$L(3) = E, \quad W = 3$$

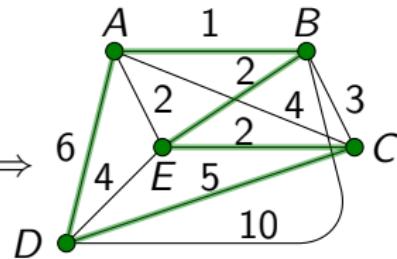
Example: Applying the Nearest Neighbour algorithm (cont.)



$$L(4) = C, \quad W = 5$$



$$L(5) = D, \quad W = 10$$



$$L(6) = A, \quad W = 16$$

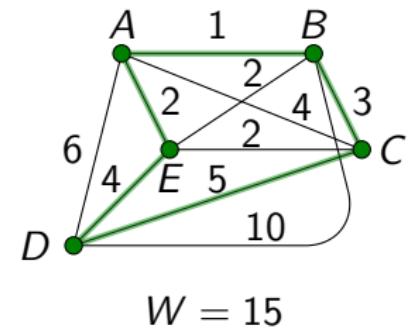
So Nearest Neighbour has generated a tour of weight 16.

However this is NOT minimal.

The weight of a minimal tour is in fact 15.

Here is a tour of minimal weight:

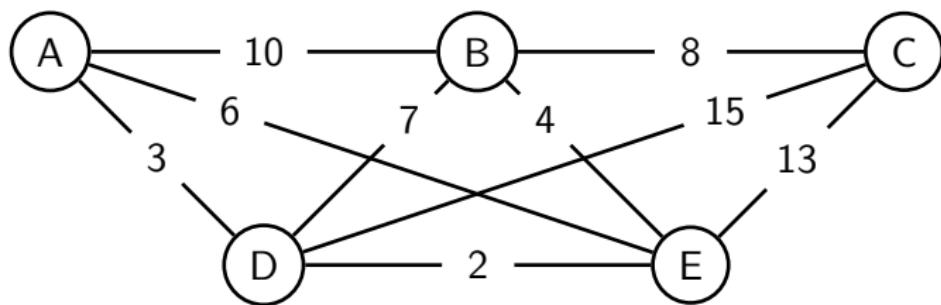
Note that Nearest Neighbour may generate this tour if we start at D instead of A . Then $L(2) = E$ and it just depends on the choice for $L(3)$.



$$W = 15$$

Shortest Path — Introduction

Consider this weighted graph:



Problem: Find a shortest path from A to C.

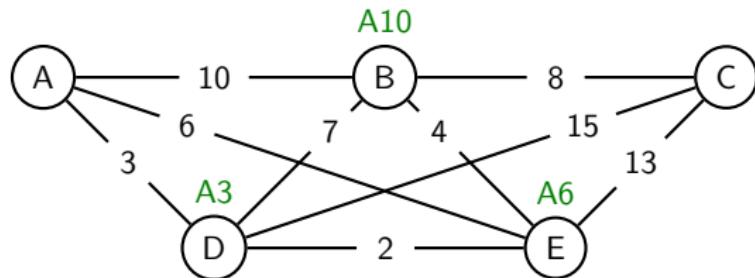
That is, from the many paths from A to C find one whose total weight is as small as possible.

For a small graph like this you can soon find a shortest path just by trying many alternatives (there are 10 or so simple $A \rightarrow C$ paths here).

Can you spot a shortest path?

For large graphs this approach is not practical. We need an *algorithm*.

Dijkstra's Algorithm



Edsger Dijkstra 1930 - 2002

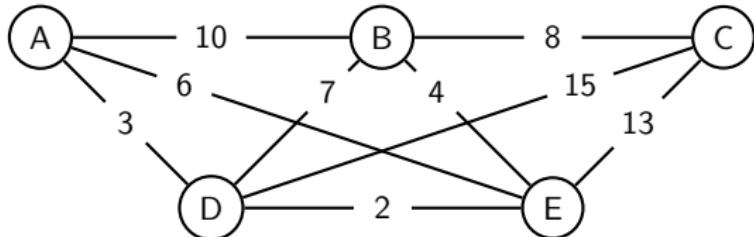
Did you notice the special layout of the graph?

As will be explained, when using Dijkstra's algorithm (for finding a shortest path) markers and labels are inserted above vertices (green text in the above example). The special layout helps to avoid these annotations getting mixed up with the main graph information.

Dijkstra's algorithm has some similarities to Kruskal's algorithm for finding a minimum spanning tree but is a little more complicated. So I will delay setting out the algorithm formally and launch straight in to demonstrating its use on the above example.

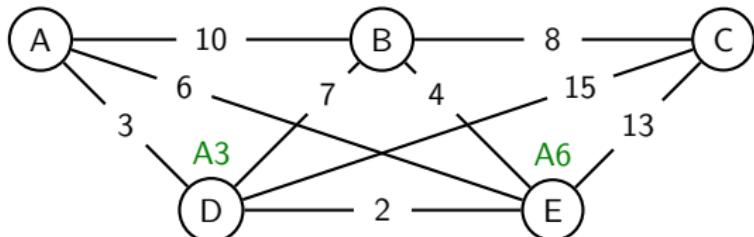
Example 1 — Slide 1

We seek a minimal weight path from A to C in the graph below.



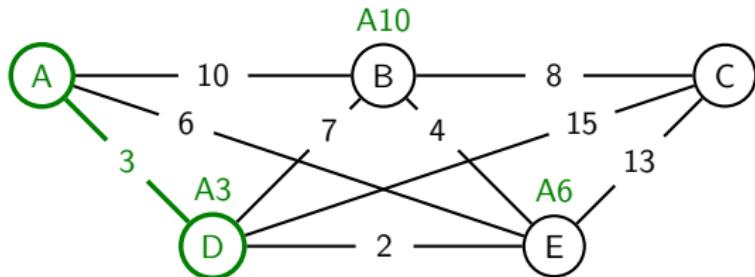
In particular this will yield the minimal 'distance', via graph edges, of C from A. In fact the algorithm proceeds by progressively finding the minimal distance of many, possibly all, vertices from A.

Start by labelling each vertex adjacent to A with its 'direct' distance from A:



Example 1 — Slide 2

Vertex D carries the smallest of these distances. 'Lock it in' and also the edge leading to it (I use heavy lines to indicate locking):



We call vertex D the **current vertex c**.

Next mark and label any un-locked vertex v adjacent to the current vertex c .

There are always three possibilities, and all three occur here:

Possibility 1: v is unmarked. e.g. Vertex C .

Mark v with c and label with its distance from A via c .
So we write **D18** above C.

Possibility 2: v needs updating. e.g. Vertex E .

The distance to v via c is less than the distance currently shown. Remark with c and relabel with the shorter distance.

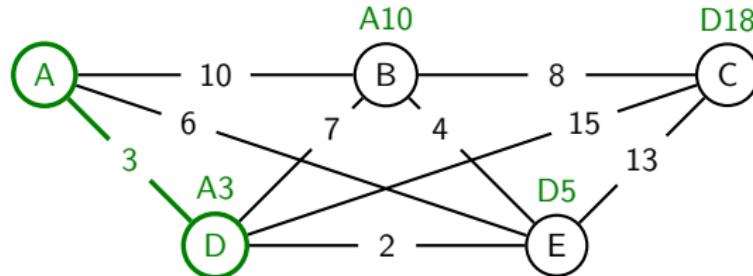
So we replace A6 above E with D5

Possibility 3: v needs no updating. e.g. Vertex B.

The distance to v via c is no less than the distance currently shown.

So we leave the A10 above B as it is.

The annotated graph now looks like this:



Example 1 — Slide 4

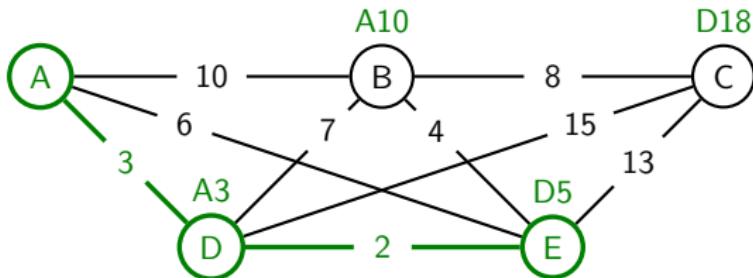
We now have three so-called ‘fringe’ vertices, B, C and E.

Fringe vertices are those that have been annotated (marked and labelled) but not yet locked in.

Now locate and lock in a fringe vertex v with the lowest label value.

That's vertex E in our example since $5 < 10$ and $5 < 18$.

Also lock in its marked lead-in edge. That's edge DE for us.



The latest locked-in vertex E becomes the new current vertex.

We now repeat the process applied to the previous current vertex D.

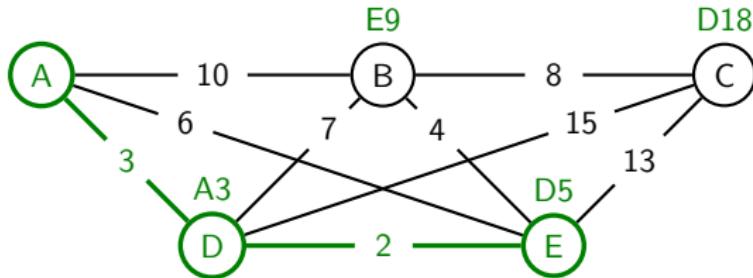
Example 1 — Slide 5

Vertex E is adjacent to two un-locked vertices, B and C:

Vertex B needs updating, since $5 + 4 = 9 < 10$. So it is re-marked with E and re-labelled 9; i.e. A10 is replaced by E9.

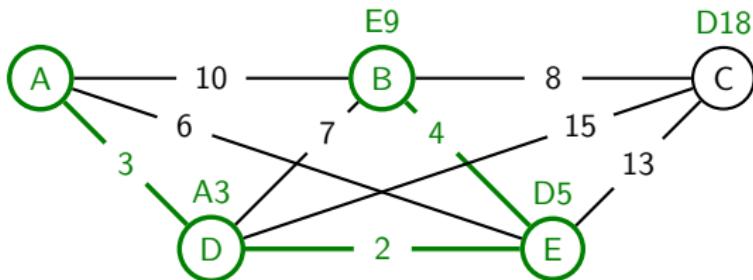
Vertex C does not need updating since $5 + 13 = 18$, and C is already labelled 18.

The updated graph now looks like this:



The lowest fringe value is now 9 on B, so we lock in B and its lead-in edge EB (next slide).

Example 1 — Slide 6



The new current vertex is the just locked-in B.

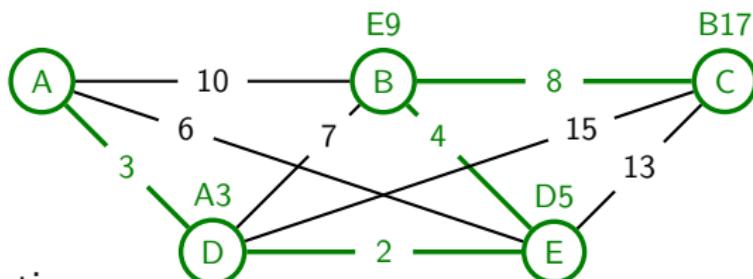
There is only one vertex adjacent to B that has not already been locked in, namely C. This needs updating since $9 + 8 = 17 < 18$.

So D18 is replaced by B17.

Since vertex C is the only vertex in the fringe it has the lowest label by default. So C and its lead-in edge BC are locked in.

We have now locked in the minimal distance 17 into our 'target' vertex C, so we can stop. The final diagram is

Example 1 — Slide 7; Results and Comments



Some Observations:

- Besides the shortest path from A to C, the solution provides the shortest path to all the vertices along that path. For this example that happens to be the entire vertex set.
- Had there remained un-locked vertices, we could have continued the process until there were none.
- Since no vertex is locked twice, the locked edges form a tree. The required shortest path is the unique path on that tree from A to C.
- With all vertices locked, the solution provides a spanning tree for the graph.

Dijkstra's Algorithm — A Formal Description

Input: (a) Connected simple graph G . Vertices A, Z from G .
 (b) Distance function $\text{dist}: E(G) \rightarrow \mathbb{Q}^+$.

Output: (a) Tree T containing A and Z as vertices.
 T is a subgraph of G .

The unique path $A \rightarrow Z$ in T has minimal total distance of all paths $A \rightarrow Z$ in G .

(b) 'Labelling' $L: V(T) \rightarrow \mathbb{Q}_+$; $L(v) = \min.\text{dist}(A \rightarrow v)$.

Method:

1. Initialize the tree T : Set $V(T) = \{A\}$, $E(T) = \emptyset$.
2. Initialize a 'Marking' function $M: V(G) \rightarrow V(G) \cup \{\text{blank}\}$: Set $M(v) = \text{blank}$ for all $v \in G$.
3. Set $L(A) = 0$. Set 'current vertex' c to A .

While $c \neq Z$:

4. For each vertex v adjacent to c but not in T :
 If v is unmarked (i.e. $M(v) = \text{blank}$)
 or if $L(v) > L(c) + \text{dist}(\{c, v\})$
 set $M(v) = c$, $L(v) = L(c) + \text{dist}(\{c, v\})$.

Dijkstra's Algorithm — A Formal Description (cont.)

5. From all marked $v \in G \setminus T$ (i.e. $M(v) \neq \text{blank}$ and $v \notin T$)
(such v are said to be 'on the fringe')
select one, say w , with minimal $L(v)$.
6. Insert vertex w and edge $\{M(w), w\}$ into the tree T .
(I call this "locking in" w and its lead-in edge.)
7. Update c to w . (i.e. make w the new current vertex.)

End of While Loop

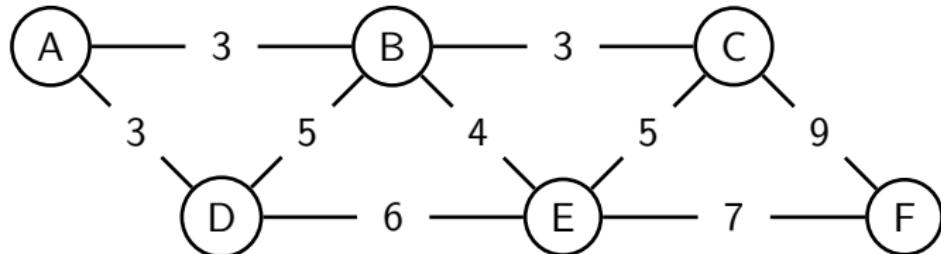
End of Method

This completes the formal description of Dijkstra's shortest path algorithm.

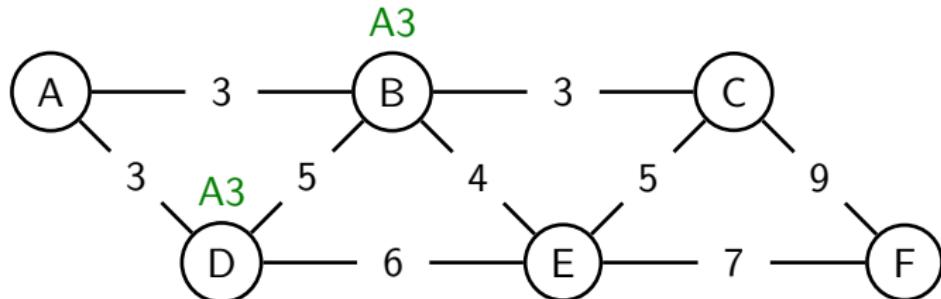
Next a second example, but this time with less commentary.

Example 2 – Slide 1

Find the shortest path for A to F:

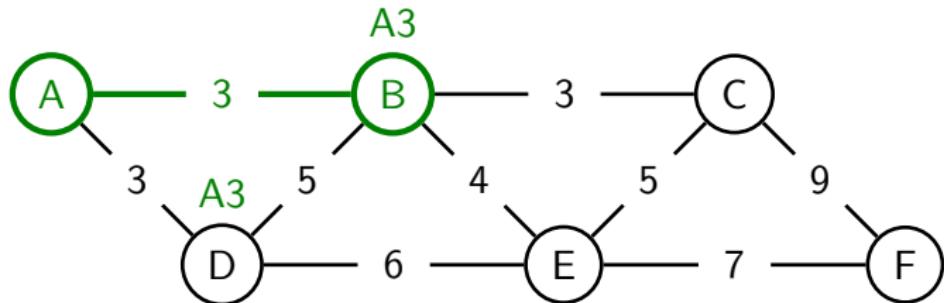


First annotate the vertices adjacent to the start vertex A:



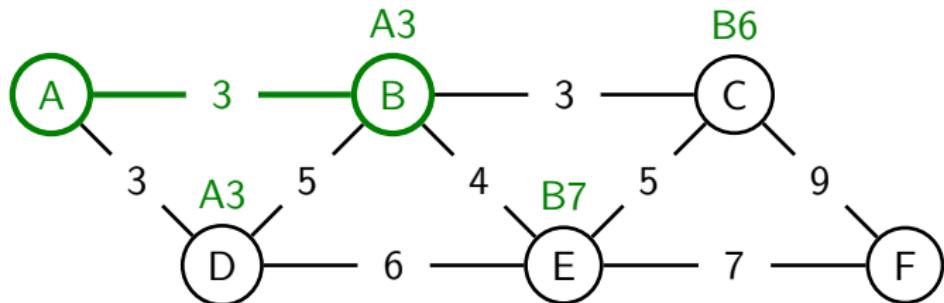
Vertices B and D have equal lowest label; let's lock in B:

Example 2 – Slide 2



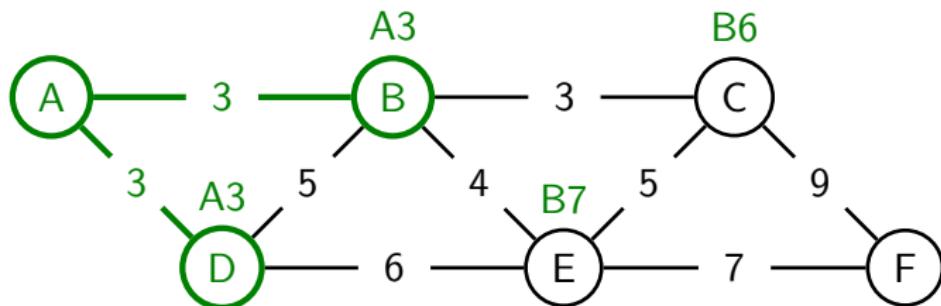
Current vertex is now B. Fringe vertices will be C,D,E.

Annotations required for C and E but D's does not need updating.

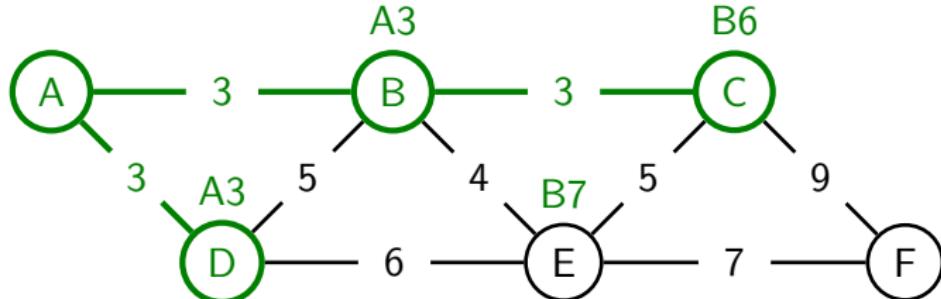


D now has lowest label so needs locking in next:

Example 2 – Slide 3

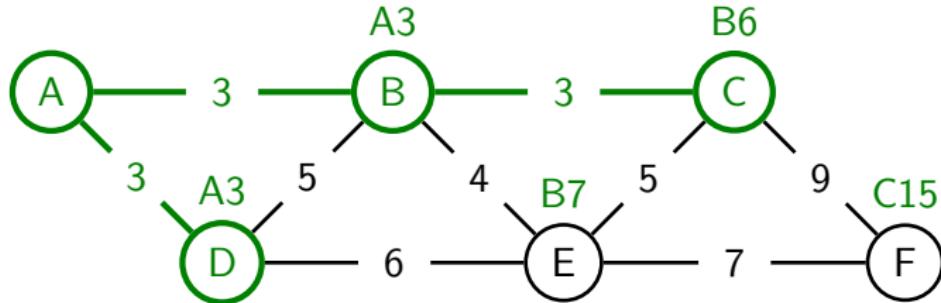


Current vertex is now D. Its only un-locked neighbour is E but no updating is required. The fringe vertices are C and E. Vertex C has lower label value so it is next to be locked:

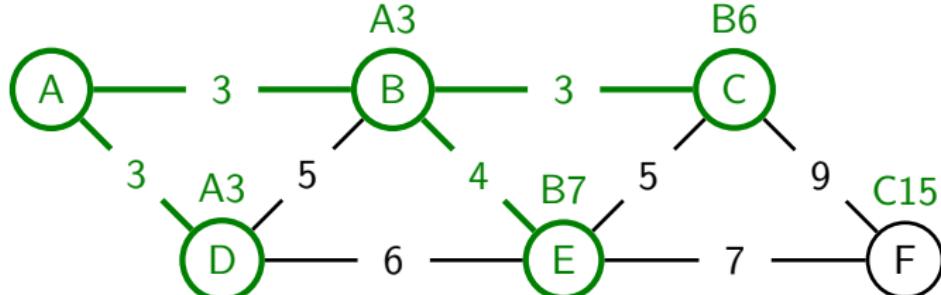


Example 2 – Slide 4

Of the two un-locked vertices adjacent to C, E is marked but does not need updating while F is unmarked and so needs annotating:

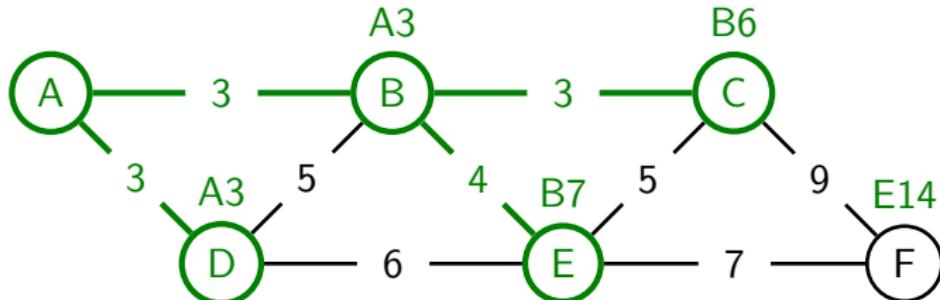


Of the two fringe vertices, E has the lower label value so is locked in. Its lead-in vertex is marked as B, so edge BE is also locked in.

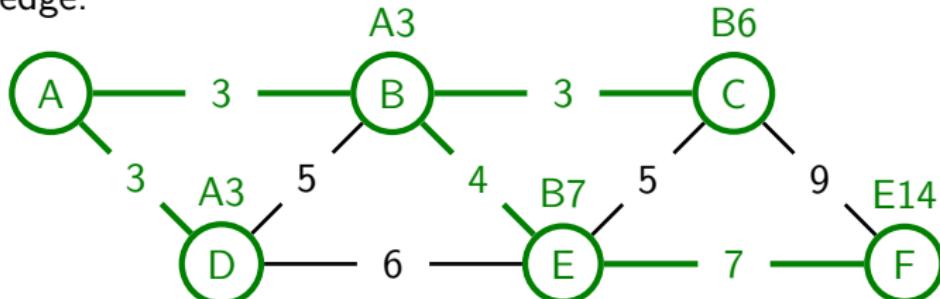


Example 2 – Slide 5

The new current vertex E has only one un-locked neighbour, F, and F needs updating since $7+7 < 15$:



Now F is the only fringe vertex, so F is locked in, together with its lead-in edge.

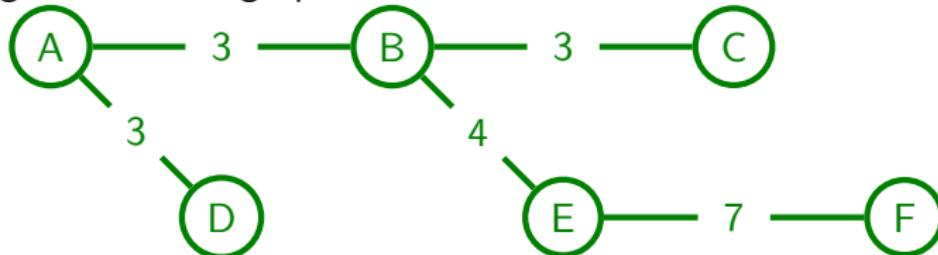


Example 2 — Slide 6; Results and Comment

Since the 'target' vertex F has now become the current vertex, the algorithm terminates. From the annotation E14 on F we read off that the shortest 'distance' between A and F is 14.

Backtracking using the annotations gives $F \leftarrow E \leftarrow B \leftarrow A$. That is, the shortest path from A to F is ABEF.

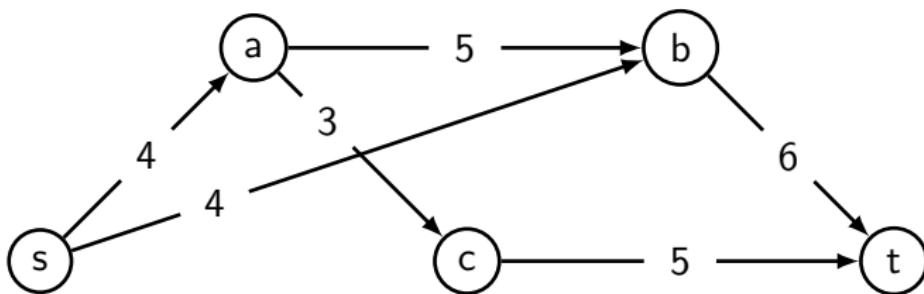
As can often happen with small examples, the algorithm has locked in all vertices of the graph, meaning that the locked-in edges provide a spanning tree for the graph:



As it happens, this is a minimal spanning tree. However, in general a spanning tree produced by Dijkstra's algorithm will not be minimal.

Transport networks

The digraph below is an example of a simple **transport network** :



The defining features of a simple transport network are:

- The edges are weighted with positive weights, called **capacities**.
- There is exactly one special vertex called the **source** (labelled 's') which is not the end vertex of any edge.
- There is exactly one special vertex called the **sink** or **target** (labelled 't') which is not the start vertex of any edge.
- Every edge lies on some simple (directed) path from s to t.

Flows

The motivation for transport networks is to model the *flow* of something, such as gas or information, through a network of links.

A simple example to keep in mind is water flowing from a reservoir (the source) through various pipes (the edges) and exchange nodes (the vertices) to another reservoir (the sink, or target).

An ‘active’ transport network has a flow $F(e) \geq 0$ through each edge e . The **flow** F , like the capacities C , is a function $E(D) \rightarrow \mathbb{Q}_+$, where D is the digraph representing the network. However, each edge e has a *fixed capacity* $C(e)$ whereas its flow $F(e)$ can vary, subject to constraints:

- (1) **Flow cannot exceed capacity.** $[\forall e \in E(D) \quad F(e) \leq C(e).]$
- (2) **In each edge, flow direction = edge direction.**
- (3) **Total flow into a node equals total flow out, except for nodes s, t .**
 $[\forall v \in V(D) \setminus \{s, t\} \quad \sum_{e \in v_{\text{in}}} F(e) = \sum_{e \in v_{\text{out}}} F(e), \text{ where } v_{\text{in}}, v_{\text{out}} \text{ are the sets of edges coming } \mathbf{in} \text{ to, and } \mathbf{out} \text{ of, } v, \text{ respectively.}]$

Finding a maximum flow

For any flow F the constraints on the network imply that the total flow out of the source must equal the total flow into the target:

$$\sum_{e \in s_{\text{out}}} F(e) = \sum_{e \in t_{\text{in}}} F(e) = v(F) \text{ say.}$$

This common total $v(F)$ is the **flow value** and any flow achieving the maximum possible flow value is called a **maximum flow**, F_{\max} . (There may be several flows that achieve the maximum.)

I will demonstrate a vertex labelling algorithm for finding an F_{\max} . The algorithm starts with the **zero flow** F_0 , where $\forall e \in D F_0(e) = 0$. Then, over a number of stages, expanded flows F_1, F_2, \dots are obtained until F_{\max} is reached.

At stage i , flow F_i is constructed as $F_i = F_{i-1} + f_i$, where the incremental flow f_i is based on a constant $k_i \in \mathbb{Q}^+$ and a simple path p_i from s to t :

$$f_i(e) = \begin{cases} k_i & \text{for every edge } e \text{ on the path } p_i \\ 0 & \text{for every other edge } e. \end{cases}$$

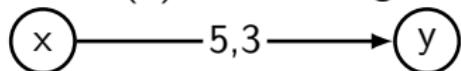
Spare capacity

In the actual lecture I skip this slide and the next, jumping straight to the first example and explaining notation and terminology as they arise. That makes them more digestible I believe.

Giving a formal description of the vertex labelling algorithm at this point may be a little overwhelming, so I will start with an example. But I need to first explain my terminology and mark-up conventions.

Given a transport network D with capacity and flow functions C, F , we know that $F(e) \leq C(e)$ for every edge $e \in E(D)$. I will call the non-negative difference $S(e) = C(e) - F(e)$ the **spare capacity** of the edge e . (Some authors use the term “excess capacity”.)

To depict a flow I will follow the capacity value $C(e)$ on each (directed) edge e with the flow value $F(e)$ for that edge. For example



represents a flow of 3 in the edge from x to y , with spare capacity 2.

Levels and labels

At each stage of the vertex labelling algorithm levels and labels are associated afresh with the vertices of the network.

The **level** of a vertex is determined iteratively as follows:

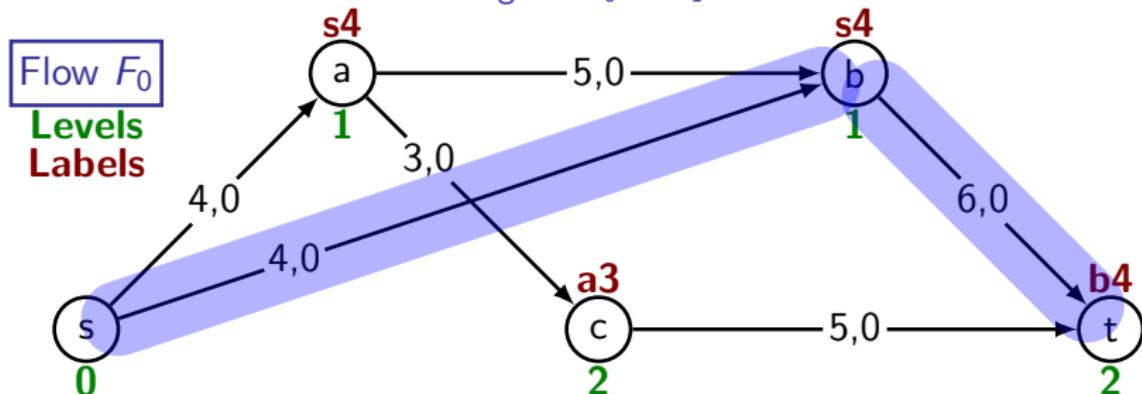
- The source vertex s always has level 0.
- If $e = (s, x)$ and $S(e) > 0$ then x has level 1.
- If x has level n and $S((x, y)) > 0$ then y has level $n + 1$ provided it has not already been assigned a lower level.

The **label** on a vertex v of level $n \geq 1$ has the form uk , where u is a vertex of level $n - 1$ and $(u, v) \in E(D)$ is an edge on the path for a potential incremental flow through v with flow value k .

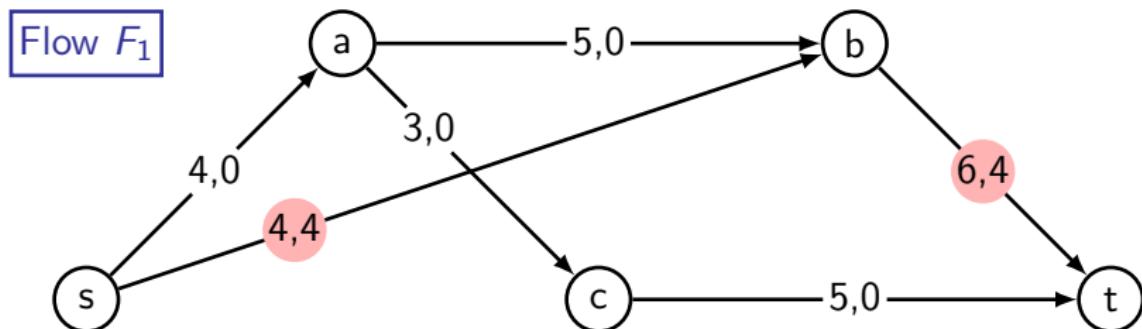
The algorithm assigns labels in ascending order of levels, and in alphabetical order within levels. Exactly how u and k are determined is explained in the formal description of the algorithm, but you may be able to infer the rule from the next example.

Vertex labelling algorithm, Example 1

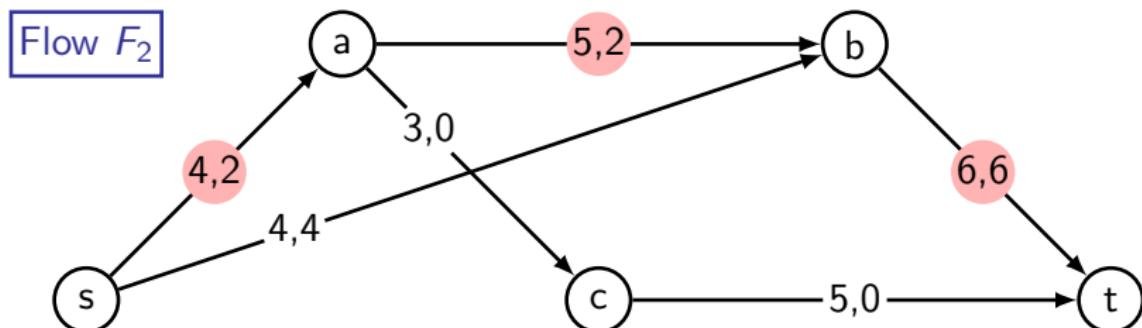
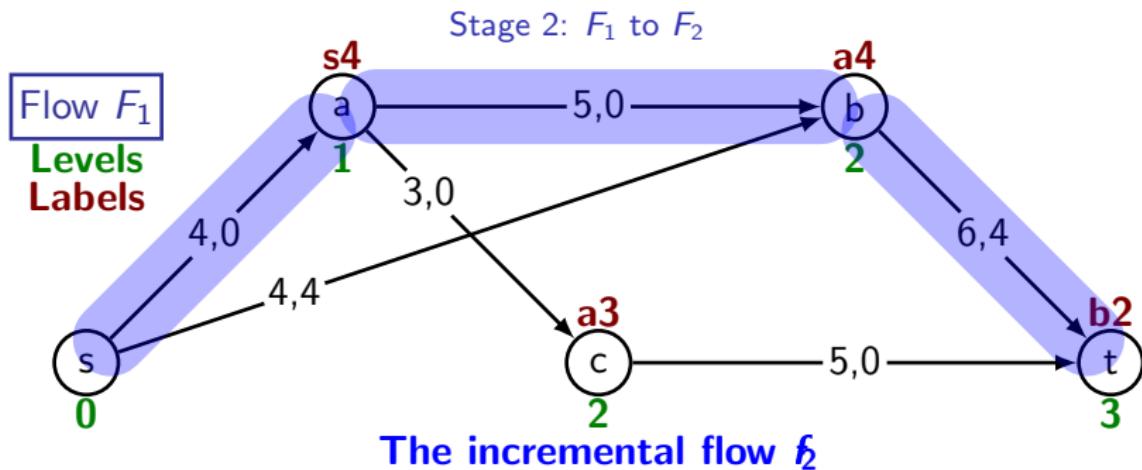
Stage 1: F_0 to F_1



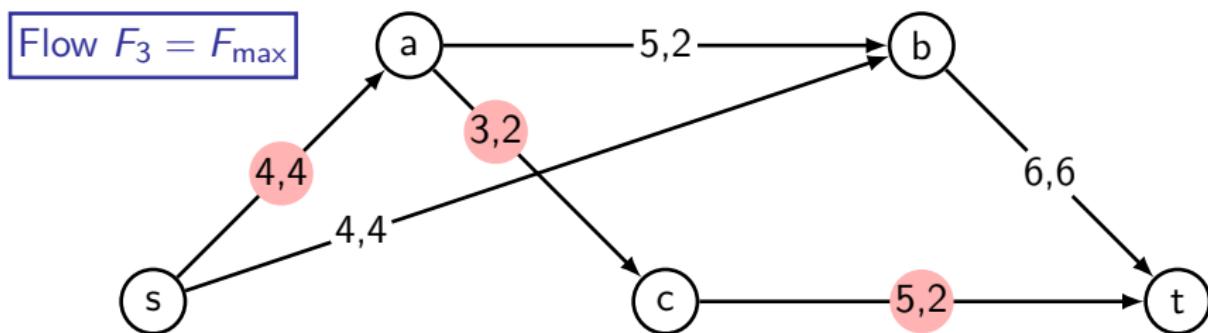
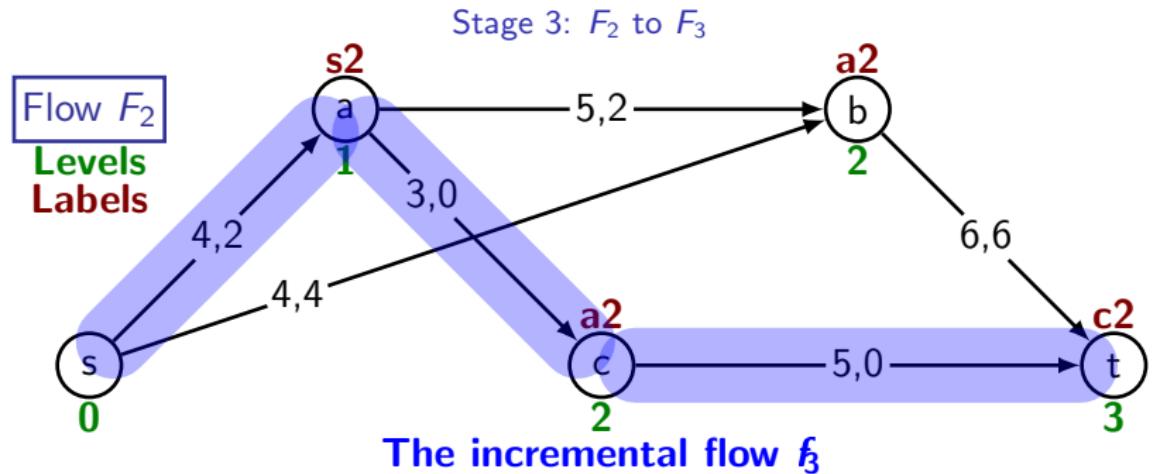
The incremental flow f_i



Vertex labelling algorithm, Example 1



Vertex labelling algorithm, Example 1



Vertex labelling algorithm, Formal description

The algorithm is unavoidably complicated to describe fully.

Moreover a further complication (virtual capacity) still needs adding.

Input: Transport network D with capacity function C .

Output: A maximum flow function F_{\max} for the network.

Method: Initialise F to the zero flow F_0 . Initialize i to 1.

For $i = 1, 2, \dots$ carry out stage i below to attempt to build an incremental flow f_i .

If stage i succeeds, define $F_i = F_{i-1} + f_i$ and proceed to stage $i+1$.

If stage i fails, define $F_{\max} = F_{i-1}$ and stop.

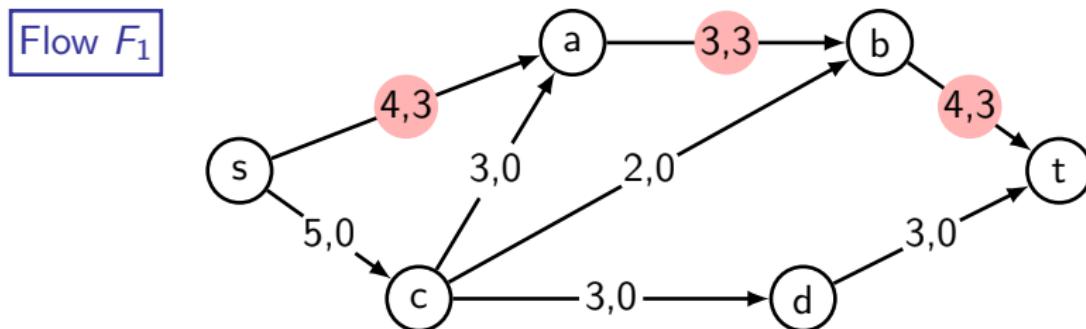
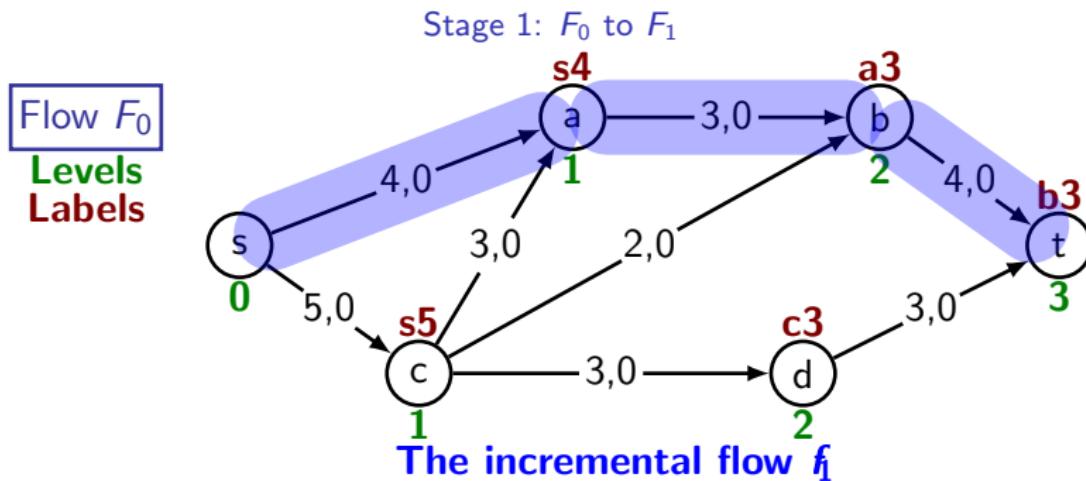
Stage i :

1. If $i > 1$, mark up the amended edge flows for F_{i-1} .
2. Mark up the levels for F_{i-1} , as explained earlier.
3. Next slide....

3. If t is assigned a level, stage i will succeed, so continue.
If not, then stage i fails, so return above to define F_{\max} and terminate.
4. Mark up labels for F_{i-1} as follows until t is labelled:
 - (a) Label each level 1 vertex v with sk_v , where $k_v = S((s,v))$.
 - (b) If t has level 2 or more now work through the level 2 vertices in alphabetical order, labelling each vertex v with uk_u , where
 - u is the alphabetically earliest level 1 vertex with $(u,v) \in E(D)$ and $S((u,v)) > 0$,
 - k_v is the minimum of $S((u,v))$ and the value part of u 's label.
 - (c) If t has level 3 or more now work through the level 3 vertices in a similar manner and so on.
5. Let p_i be the path $u_0u_1\dots u_n$ where $u_n = t$ and for $0 < j \leq n$ u_j has label $u_{j-1}k_j$.
Define f_i to be the incremental flow on p_i with flow value k_n .

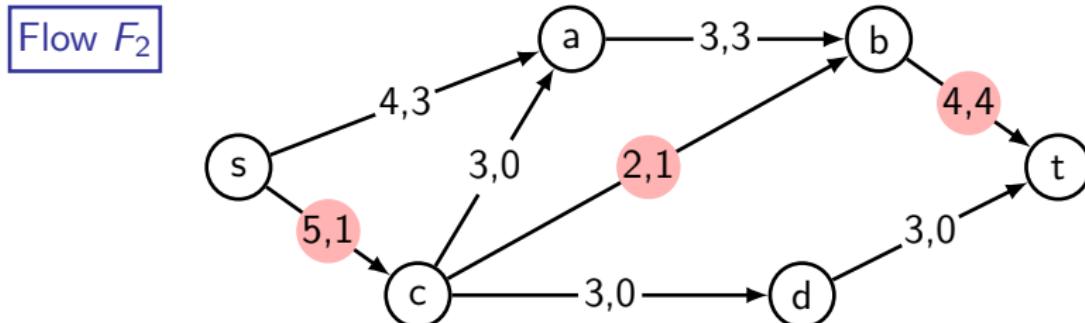
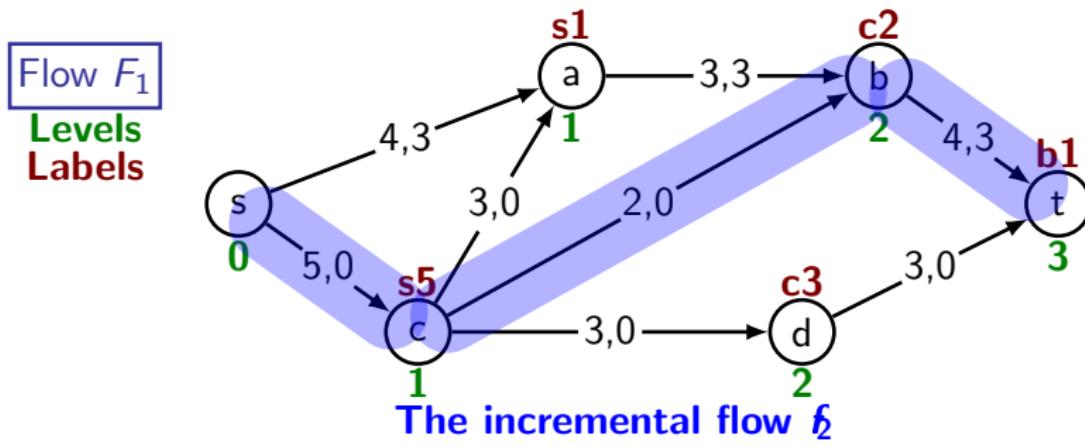
End of Method

Vertex labelling algorithm, Example 2



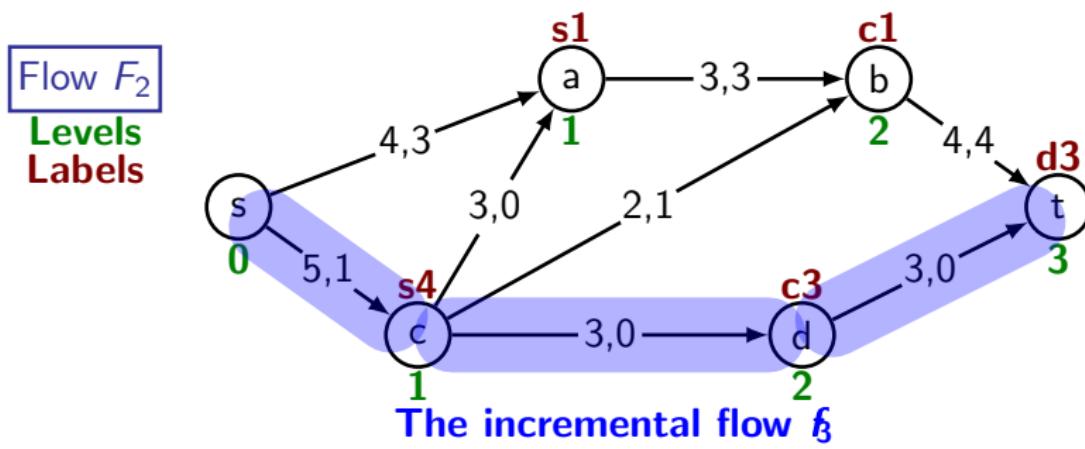
Vertex labelling algorithm, Example 2

Stage 2: F_1 to F_2



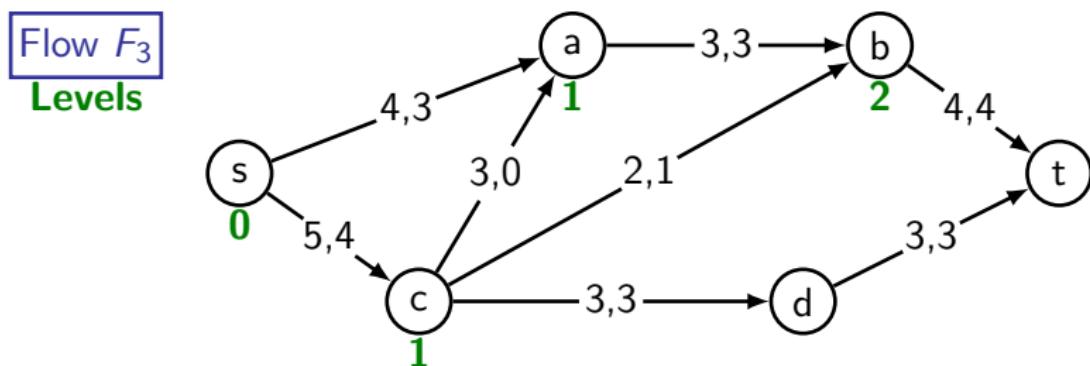
Vertex labelling algorithm, Example 2

Stage 3: F_2 to F_3



Vertex labelling algorithm, Example 2

Stage 4: F_3 is F_{\max}



No level can be assigned to t !

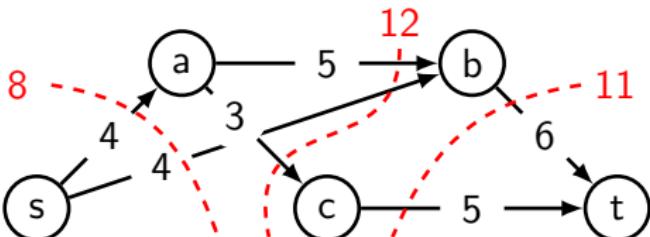
So the algorithm terminates with $F_{\max} = F_3$.

Cuts

Consider again Example 1 at right.

Since the total capacity of edges leading from the source is $4 + 4 = 8$ it is

clear that the maximum flow value cannot be more than 8.



In fact the maximum flow value *is* 8 as a flow with that value is easily found. (We found it using the vertex labelling algorithm, but that isn't really needed on such a simple example.)

The set of edges leading out of the source is an example of a network *cut*, so called because removing the edges would 'cut' the network in two, separating the source from the target.

The $4 + 4$ is the 'capacity' of the cut.

A cut may be indicated by drawing a (red) line through its edges. Some examples, with values, are shown on the digraph above.

Max Flow Min Cut

Observe that for Example 1 the minimum cut and maximum flow have the same value (8). As you probably expect, this is in fact true for all transport networks, as given by the theorem below.

First a formal definition of 'cut':

A set K of edges in a transport network D is called a **cut** when there is a partition $\{S, T\}$ of $V(D)$ with $s \in S$ and $t \in T$ such that K comprises all edges of D that start in S and finish in T ; i.e. $K = E(D) \cap (S \times T)$.

The **capacity of a cut K** is $\sum_{e \in K} C(e)$.

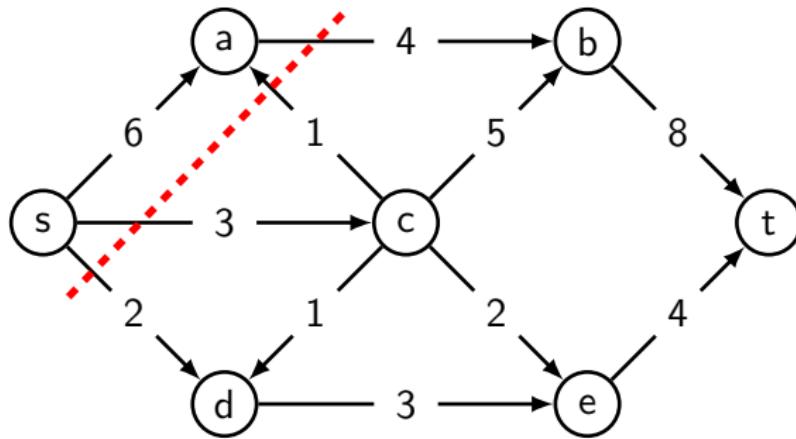
Max flow min cut theorem: For any transport network:

$$\text{maximum flow value} = \text{minimum cut capacity}$$

Though highly plausible, this theorem is little tricky to prove, and the proof will be omitted, as will the proof that the vertex labelling algorithm always finds a maximum flow.

Max flow min cut: Class example 1

What is the maximum flow value for this transport network?



Answer: After some searching we find the minimum cut shown, for which $S = \{s, a\}$ and $T = \{b, c, d, e, t\}$.

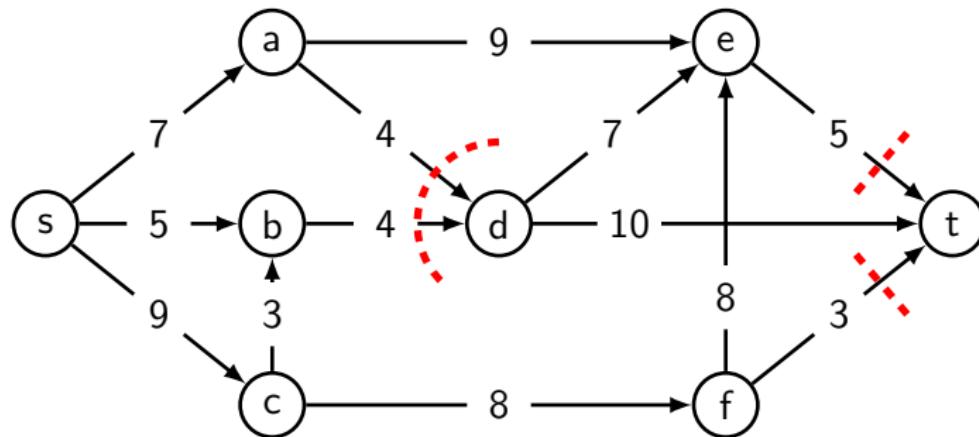
The capacity of this cut is $4+3+2=9$, so this is the maximum flow.

Note: Edge (c,a) is not in the cut since it's in the wrong direction.

Max flow min cut: Class example 2

from Kolman, Busby & Ross

What is the maximum flow value for this transport network?



Hint: This time the minimum cut cannot be drawn as a single line!

Answer: Use all the edges from $S = \{s, a, b, c, e, f\}$ to $T = \{d, t\}$.

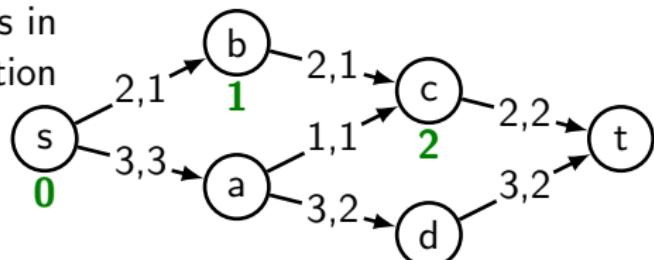
The capacity of this cut is $4+4+5+3=16$, so this is the max flow.

Introduction to virtual flows

The next example (Example 3) will show that in order to always achieve a maximum flow the labelling algorithm needs a modification.

Part way through assigning levels in

Stage 4 (of Example 3) the situation becomes as shown at right. We are 'stuck' at vertex c, but so far we only have a flow value of 4, whereas the min cut value is 5.



To escape from this we need to 'divert' the flow $a \rightarrow c \rightarrow t$ to $a \rightarrow d \rightarrow t$, thereby allowing another 1 unit of flow $s \rightarrow b \rightarrow c \rightarrow t$.

The algorithm accomplishes this diversion by allowing a 'virtual flow' of 1 unit $c \rightarrow a$ so that, in particular, vertex a becomes level 3.

What happens after that is shown as Stage 4 unfolds in the full description of the use of the algorithm in Example 3 that follows.

First, the definition and an explanation of how the algorithm is modified.

The complete vertex labelling algorithm

Let (u,v) be a (directed) edge in a transport network D , and suppose there is currently a flow of $f > 0$ along this edge. The vertex labelling algorithm can reduce this flow to $g < f$ by introducing a **virtual flow** of $f - g$ in the opposite direction, i.e. from v to u .

Virtual flows are *always* in a direction counter to that indicated on the digraph D , and if part of an incremental flow will *reduce* the flow in the relevant leg of the path.

The modification to the algorithm, affecting the assignment of levels and labels, is an amendment to the definition of spare capacity.

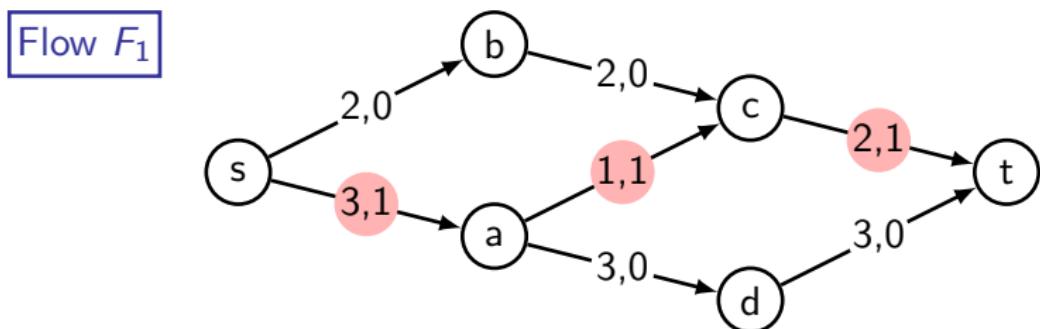
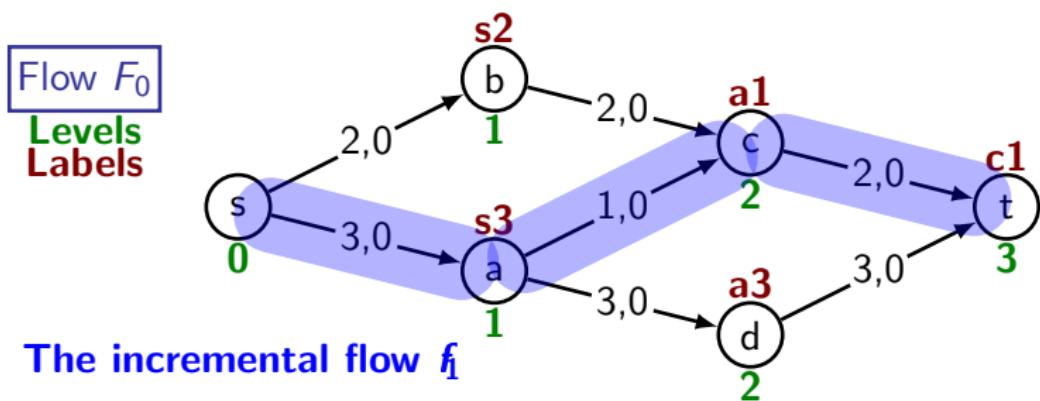
For vertices u,v of D , where D has capacity and flow functions C, F :

$$S((u,v)) = \begin{cases} C((u,v)) - F((u,v)) & \text{if } (u,v) \in E(D) \\ F((v,u)) & \text{if } (v,u) \in E(D) \\ 0 & \text{otherwise} \end{cases}$$

When $(v,u) \in E(D)$, $S((u,v))$ is called a **virtual capacity**.

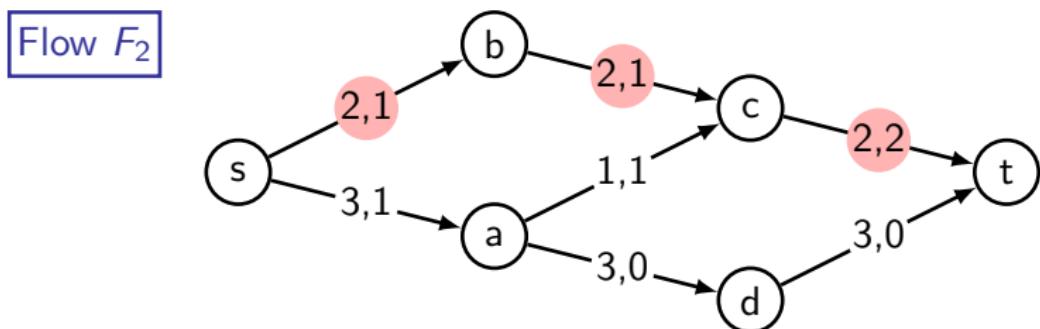
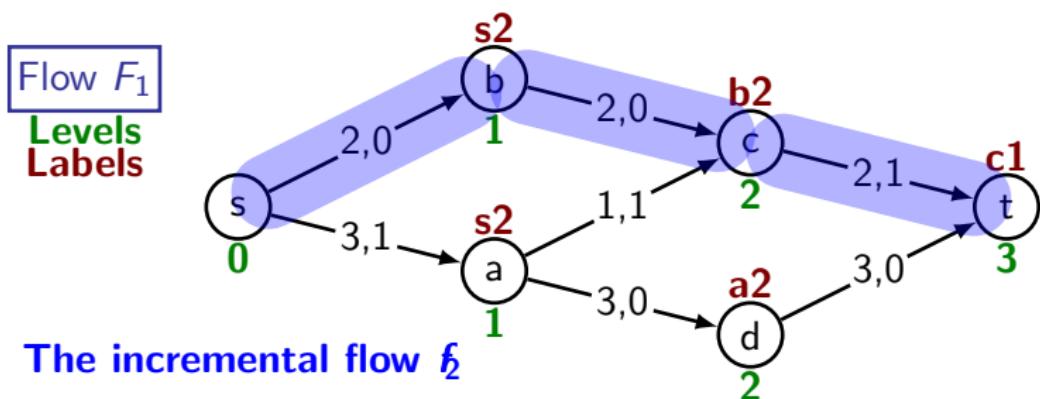
Vertex labelling algorithm, Example 3

Stage 1: F_0 to F_1



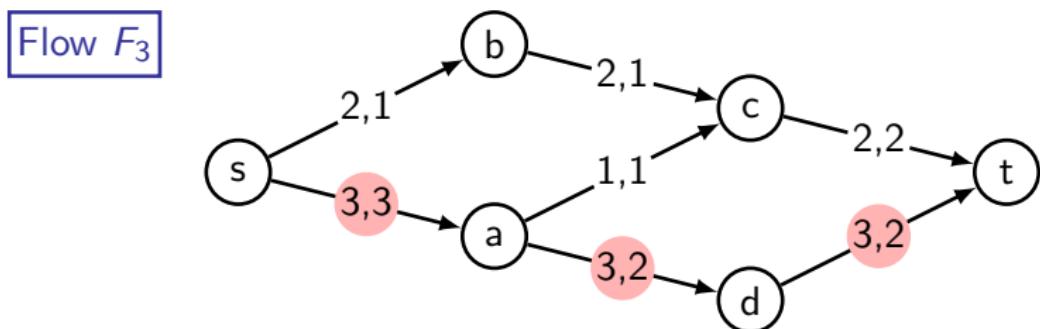
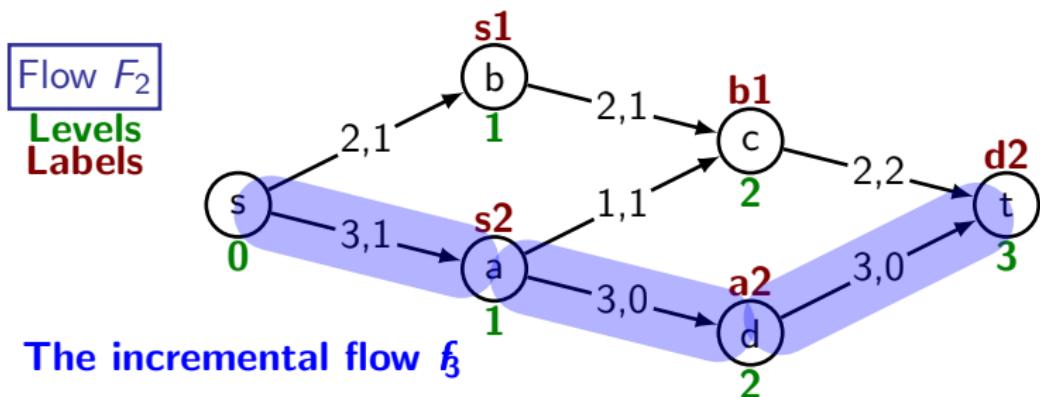
Vertex labelling algorithm, Example 3

Stage 2: F_1 to F_2

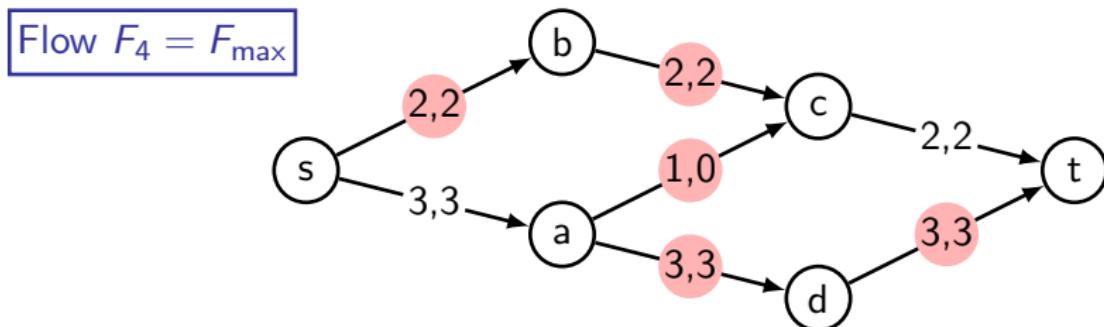
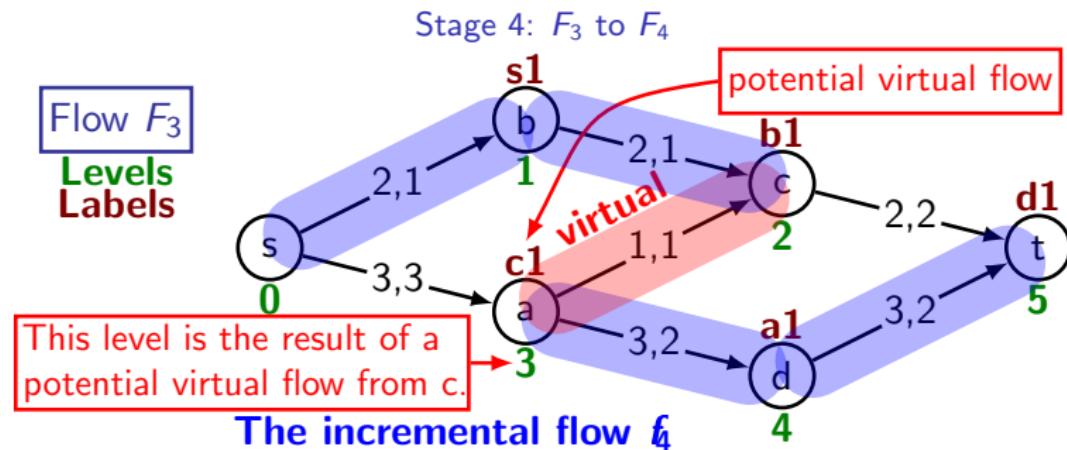


Vertex labelling algorithm, Example 3

Stage 3: F_2 to F_3



Vertex labelling algorithm, Example 3



A matching problem

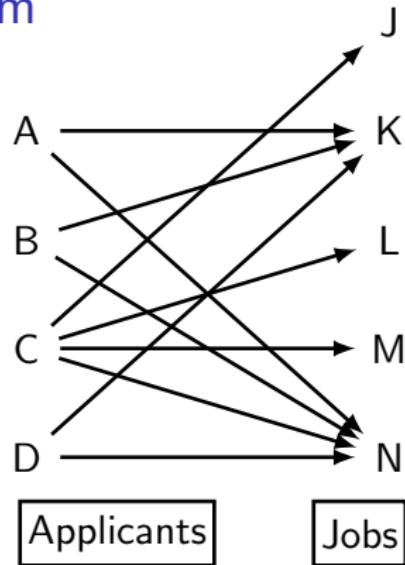
from Johnsonbaugh

Four people A,B,C,D are each interested in one or more of five jobs J,K,L,M,N on offer. The diagram indicates who is interested in what job. Is it possible to satisfy everyone?

That is, can each applicant x be *matched* with a job $m(x)$? If so, how?

If not, what's the best that can be done?

One answer:	$\begin{array}{c cccc} x & A & B & C & D \\ \hline m(x) & K & N & J & - \end{array}$
-------------	--



More generally, given a relation $R \subseteq S \times T$ a **matching problem** seeks a **maximal matching function** (or just a 'matching') $m \subseteq R$. This is a **injective** (one-to-one) function $f : S' \rightarrow T$ with domain $S' \subseteq S$ as large as possible subject to m being an injective subset of R .

Solving matching problems

The little 4-applicants-5-jobs matching problem can be easily solved ‘by eye’. Larger matching problems require a more systematic method. Perhaps surprisingly, the vertex labelling algorithm for transport networks can be used.

The matching problem for $R \subseteq S \times T$ can be converted to the max flow problem for a transport network as follows:

1. Create a digraph D by adding to the arrow diagram for R
 - a **supersource** s and edges (s,x) for each $x \in S$ and
 - a **supertarget** t and edges (y,t) for each $y \in T$.
2. Assign capacity 1 to every edge of D . (i.e. $\forall e \in E(D) C(e) = 1$.)

A solution to the max flow problem provides the matching:

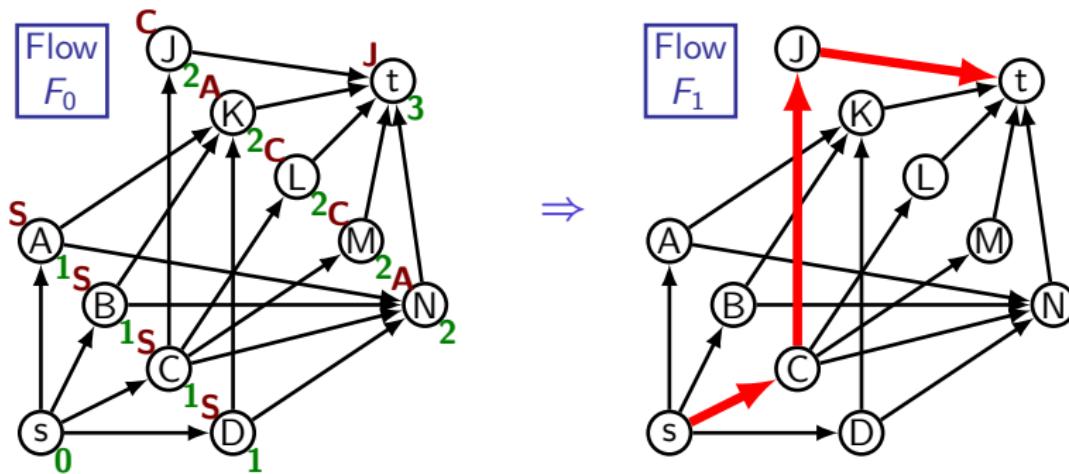
$$m = \{(x,y) \in S \times T : F_{\max}((x,y)) = 1\}.$$

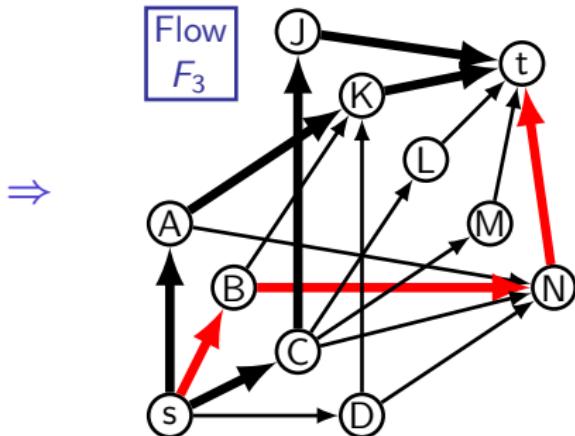
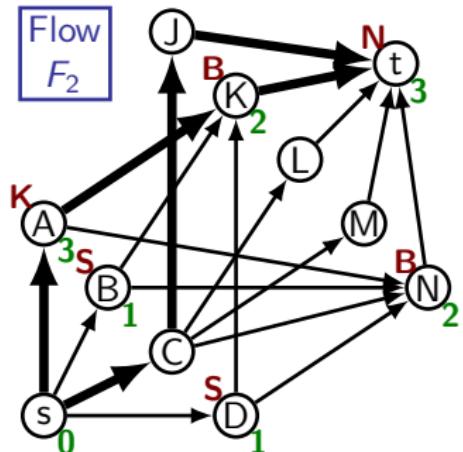
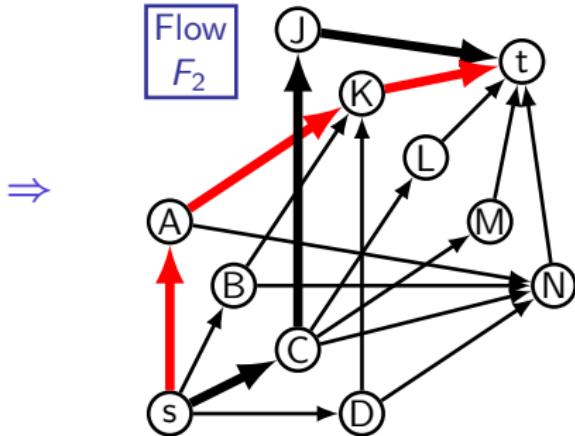
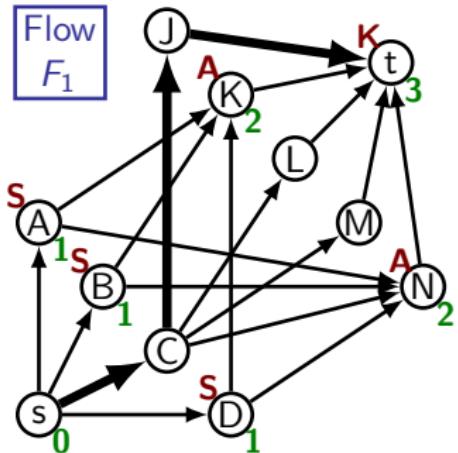
Vertex labelling for matching; Example 1

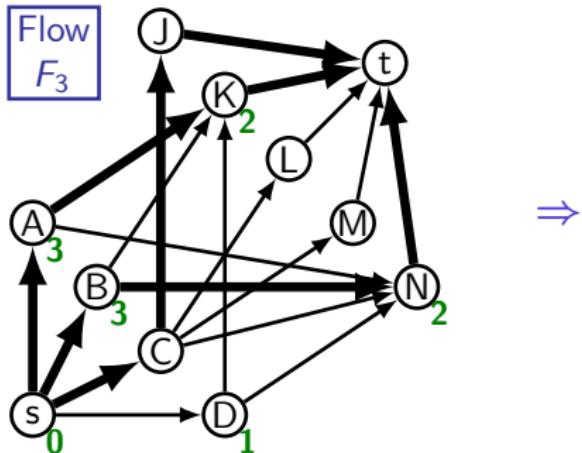
Here is how vertex labelling would be used on the Johnsonbaugh problem. This is just for demonstration, since, as we've seen, it's easily solved by eye.

With all edge capacities 1, edge flows are either 0 or 1. Notation will be simplified:

- Edge capacities not marked.
- (Non-zero) Edge flows marked by edge thickening.
- Potential flow values not indicated on labels.







Even using virtual capacities to reach vertices of level 3, it isn't possible to assign a level to t , so the algorithm terminates.

The matchings are indicated by the edges between $\{A, B, C, D\}$ and $\{J, K, L, M, N\}$ that have flow 1 (the thick edges).

So the matching is

$$m = \{(A, K), (B, N), (C, J)\}.$$

As remarked earlier, the above solution is very easy to get without the use of the vertex labelling algorithm. We have just matched the first member of T with the first 'available' member of S , then the next member of T with the next 'available' member of S and so on.

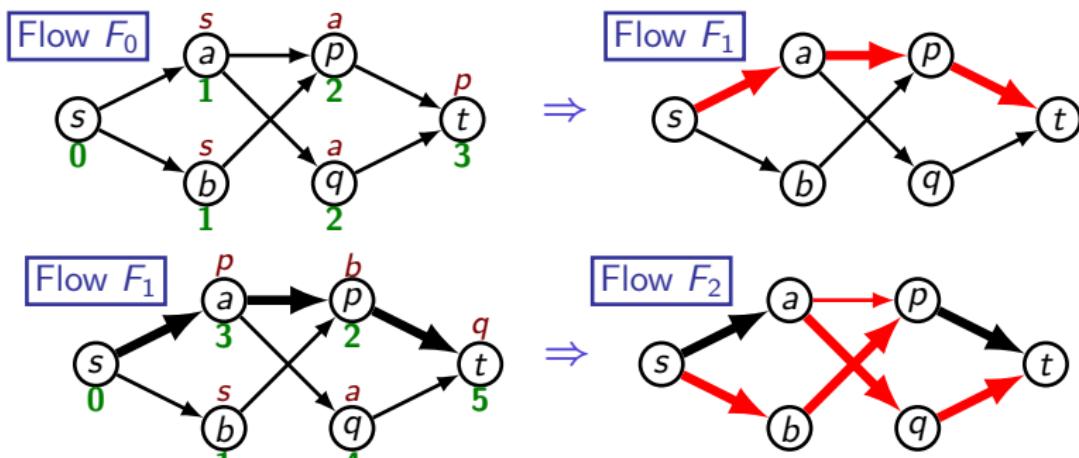
The benefit of using the algorithm lies in its ability to 'undo' initial pairings and replace them by new ones when this becomes necessary to enable later pairings.

The final example shows how it does this in the simplest possible case.

Vertex labelling for matching; Example 2

Find a (maximal) matching function m
for the relation $R = \{(a, p), (a, q), (b, p)\}$.

Obviously the only answer is $m = \{(a, q), (b, p)\}$, but the vertex labelling algorithm will first match a with p . However it will then use a virtual flow to undo this and match a with q , allowing b to be matched with p instead. Here are the diagrams:



END OF SECTION D2

D3. Random walks on graphs.

Notes by Malcolm Brooks and Adam Piggott,
expanded from notes of Pierre Portal
with influences from Judy-anne Osborn .

Unfortunately, Random walks are not covered in our text by Epp, nor in
the books by Johnsonbaugh or Kolman *et. al.*

Random walks: idea

Let G be a digraph with n vertices $V = V(G) = \{1, \dots, n\}$ and (directed) edge set $E = E(G)$.

[We will stick to these meanings for G, n, V and E throughout this final section of the notes.]

Imagine that you are walking on this digraph.

Travelling through an edge takes you one unit of time.

At time 0, you are at vertex x .

At time 1 you are at a vertex $y \in V$, with $(x, y) \in E$.

At time 2 you are at a vertex $z \in V$ with $(y, z) \in E$.

At time k you are at a vertex t and there is a walk of length k from x to t .

Before each step, you choose where to go next probabilistically:

If you are at a vertex i you go to vertex j with probability p_{ij} .

[If $(i, j) \notin E$, then, of course, $p_{ij} = 0$.]

Walks on graphs

A graph can model many things

Anything that can be modelled with an (undirected) graph can be modelled with a digraph, simply by replacing each (undirected) edge with two directed edges.

Depending on the nature of the system your digraph models, you may be interested in describing a walk on the graph.

Recall that a walk on a directed graph is a sequence $v_0, e_1, v_1, e_2, \dots, e_n, v_n$ of vertices alternating with edges with the property that $e_i = (v_{i-1}, v_i)$ for each i such that $1 \leq i \leq n$.

Depending again on the nature of the system your digraph models, you may be interested in a walk in which the decision of where to go next is made probabilistically (because this may model something that happens in your system)...

Random walks: definition

Let G be a digraph with n vertices $V = V(G) = \{1, \dots, n\}$ and (directed) edge set $E = E(G)$.

A square matrix with non-negative entries such that the entries in each row sum to 1 is called a **stochastic matrix**.

Associated with an n -vertex directed graph G , let $T = (t_{ij})_{1 \leq i,j \leq n}$ be an $n \times n$ stochastic matrix satisfying the rule $\forall (i,j) \notin E(G) t_{ij} = 0$.

For any given n , let B_n denote the set of **basis vectors** $\{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ where \mathbf{e}_i is the $n \times 1$ vector with 1 as the i -th entry (i.e. in row i) and all other entries zero. E.g, for $n = 3$: $\mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

For $X_0 = \mathbf{e}_i \in B_n$ the sequence $(X_k)_{k \in \mathbb{N}^*}$ (called a **Markov chain**) specified by G and T is called the **random walk** on G starting at vertex i (or “at \mathbf{e}_i ”), with transition matrix T .

Then $X_k = (T')^k \mathbf{e}_i = (q_j)_{1 \leq j \leq n}$ say gives, for $1 \leq j \leq n$, the probability q_j of being at the vertex j after k steps, starting from vertex i .

Steady States

Let $S = (q_j)_{1 \leq j \leq n} \in \mathbb{Q}^n$ be a probability vector (all entries are non-negative and they sum to 1), which is a steady state vector for T , i.e.
 $T'S = S$.

What does S represent in relation to our random walk? We have seen that a Markov process does not necessarily approach a steady state from every initial state. However it can be proved that, for any initial probability vector X ,

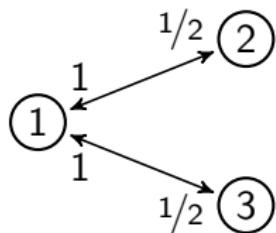
$$\frac{1}{N} \sum_{k=0}^{N-1} T^k X$$

approaches S as N gets large, so, on average, the walk is at vertex j with probability q_j , i.e. is at j 100 $q_j\%$ of the time.

Example 1

Consider a graph G with adjacency matrix A and a random walk on G with transition matrix T , where

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad T = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$



Our random walker alternates between vertex 1 and the other two vertices, with neither of these two vertices being favoured over the other.

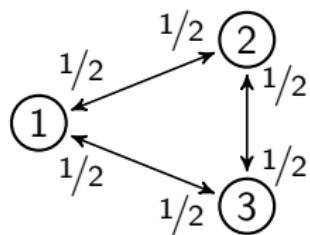
On average, the walker is at 1 half of the time and at 2, 3 a quarter of the time each, so the steady state vector is $S = \begin{bmatrix} 1/2 \\ 1/4 \\ 1/4 \end{bmatrix}$.

This is confirmed by checking that $T'S = S$.

Example 2

Consider a graph G with adjacency matrix A and a random walk on G with transition matrix T , where

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad T = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/2 & 0 & 1/2 \\ 1/2 & 1/2 & 0 \end{bmatrix}$$



We see that no one vertex is favoured over any other.

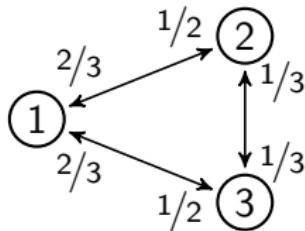
On average, the walker will spend equal time at each vertex, so the steady state vector is $S = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$.

Again, this is confirmed by checking that $T'S = S$.

Example 3

Consider a graph G with adjacency matrix A and a random walk on G with transition matrix T , where

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \quad T = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 2/3 & 0 & 1/3 \\ 2/3 & 1/3 & 0 \end{bmatrix}$$



Now our random walker slightly favours vertex 1 over the other two, with neither of these other two being favoured over the other.

So the steady state vector should have the form $S = \begin{bmatrix} p \\ q \\ q \end{bmatrix}$.

But what should the probabilities p and q be?
Can you guess?

Example 3 (concluded)

To find p and q we could use the method we used when investigating Markov chains earlier in the course.

That is, we could use the computer to solve the system of linear equations represented by the matrix equation

$$(T' - I)S = 0$$

modified by replacing all entries in the last row of both $T' - I$ and 0 by 1.

However, in this case the equations boil down to:

$$\begin{aligned} -p + \frac{4}{3}q &= 0 \\ p + 2q &= 1 \end{aligned}$$

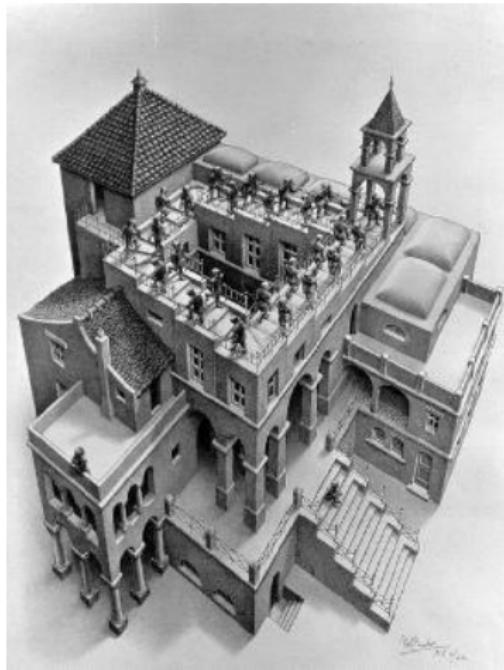
Thus $\begin{bmatrix} -1 & \frac{4}{3} \\ 1 & 2 \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, so $\begin{bmatrix} p \\ q \end{bmatrix} = \frac{1}{-10/3} \begin{bmatrix} 2 & -(\frac{4}{3}) \\ -1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 4/10 \\ 3/10 \end{bmatrix}$.

On average, our walker spends 40% of the time at vertex 1 and 30% at each of the other two vertices. Is that what you guessed?

The steady state vector is $S = \begin{bmatrix} 4/10 \\ 3/10 \\ 3/10 \end{bmatrix}$. As you can check, $T'S = S$.

A story about old people who lived long ago

Old timey people (such as myself) had it tough when we were young. Not only did we have to walk to school, and it was uphill both ways...



"Ascending and Descending," by M.C. Escher (1960). Lithograph.

Story continued

...but internet search engines did not sort hits very well. We often had to click "next page" many times to find the good hits to any search we ran.

Imagine a search for "Andrey Markov" returning 15,000 hits (relevant webpages), but the first page of hits is filled with 7th grade history projects and sneaky advertisements that list keywords irrelevant keyword so they can show up in searches. (I am exaggerating, but not as much as you think).

A new company named Google was launched by Larry Page and Sergey Brin in 1998 to market a new search engine called Google Search. The competitive advantage of Google Search, at least for this old timer, was the effective way it sorted the list of search hits before displaying them to the user. The "best" hits were invariably on the first or second page of hits. What was the new idea that Page and Brin used to outperform the existing search engines...they used some discrete mathematical modelling.

How can we decide which webpages are most awesome?

IDEA: Let the structure of an internet (the hyperlinks between pages) tell us which pages are most awesome.

HYPOTHESES OF THE MODEL: For a webpage W :

1. Each link to W (from some other page) is saying that W is a bit awesome;
2. A link to W from an awesome page is saying more than a link to W from a less awesome page.

The hard part...how to make this self-referential criteria operational.

First we model the structure of an internet

Represent an internet as a digraph called a **webgraph**. The vertices represent pages of the web, and there is a directed edge from vertex X to vertex Y if and only if there is a hyperlink from the page corresponding to X to the page corresponding to Y .

A webgraph is a discrete mathematical model of an internet.

How can we decide which webpages are most awesome?

Now consider a random walker (or random surfer) RS who surfs the web forever as follows:

1. RS chooses a page at random to start surfing.
2. At each step, the random surfer moves to a page on the web according to the following rule
 - 2.1 With probability α , the RS will type in the URL of a randomly chosen page (possibly even the current page)—we will call this “(unforced) teleporting.”
 - 2.2 With probability $(1 - \alpha)$ the RS will proceed as follows; if there is at least one hyperlink on the current page, the RS will choose at random one of these hyperlinks and click on it; if there is no hyperlink on the current page, then the random surfer will choose a *new* page at random and teleport there (“forced teleporting”).

The value $(1 - \alpha)$ is called the **damping factor**. A damping factor of 0.85 ($\alpha = 0.15$) is often used.

Some intuitive observations about the walk taken by the RS

Consider a webpage W :

1. The more hyperlinks that point to W , the more likely it is that the RS will be visiting W often;
2. If the RS is likely to visit a page X often, then a hyperlink from X to W will have a significant effect on how often the RS will visit W ; if the RS is unlikely to visit a page X often, then a hyperlink from X to W will not have a significant effect on how often the RS will visit W .

These observations appear to align well with our idea of how we might decide which webpages are most awesome. On average, the RS will visit an awesome page more often than a less awesome page.

PageRank algorithm

PageRank is a link analysis algorithm, named after Larry Page, co-founder of Google, used by the Google Internet search engine.

Let G be a webgraph for some internet with vertices

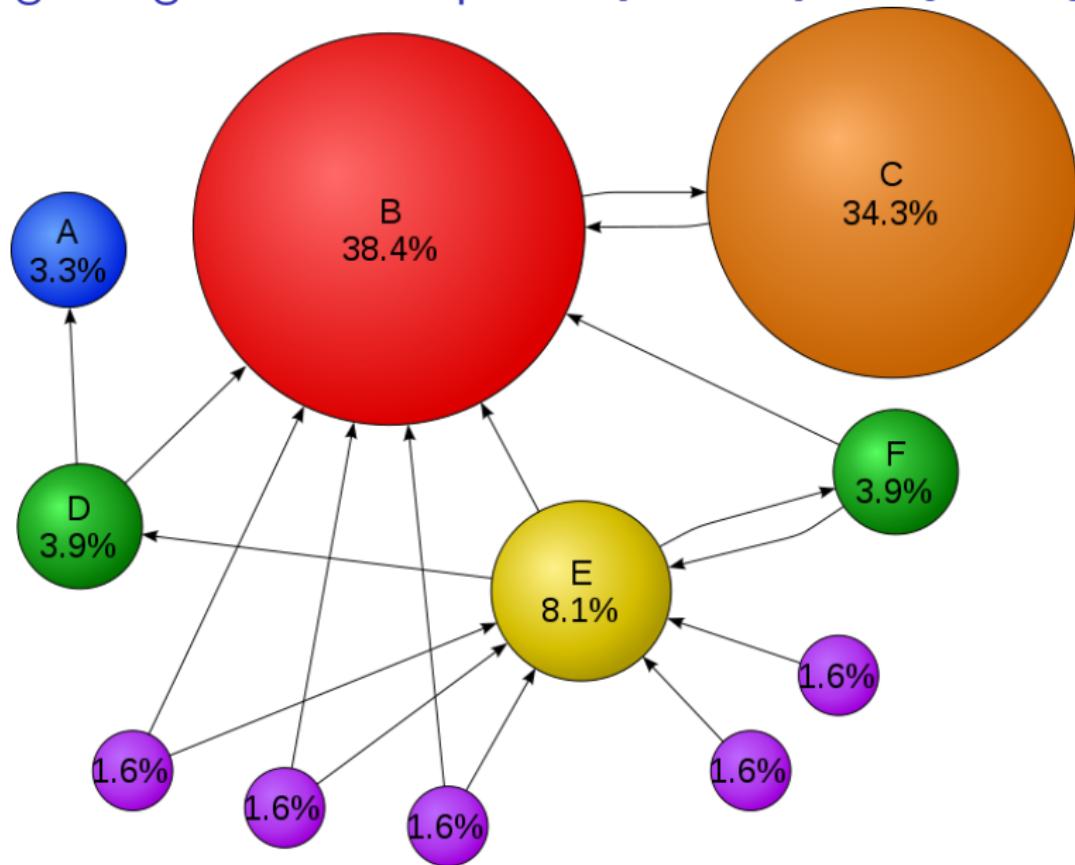
$V(G) = \{1, 2, \dots, n\}$. Let M be the transition matrix corresponding to the movement of the random surfer (with damping factor α chosen from the interval $(0, 1)$). Let $P = (p_1, p_2, \dots, p_n)$ be the steady state vector for the random walk on W using the transition matrix M . For each $i \in \{1, \dots, n\}$, p_i is the **PageRank** of page i .

The name “PageRank” is a trademark of Google, and the PageRank process has been patented (U.S. Patent 6,285,999).

[**Source:** <http://en.wikipedia.org/wiki/PageRank>]

Google PageRank Example

<http://en.wikipedia.org/wiki/PageRank>



Building the transition matrix M

We shall build M by combining a “basic transition matrix T ” with a matrix that encodes what happens when the RS decides to move about the webgraph using links or forced teleportation, and another matrix that encodes what happens when the RS decides to move by unforced teleportation.

Basic transition probabilities

Suppose that that RS never teleports unless there are no hyperlinks on the current page. Then the RS is equally likely to follow any link on a page, and, if there are no links, is equally likely to ‘teleport’ to any other page on the web.

For each vertex i , let n_i be the number vertices to which i is linked:

$$\begin{aligned} n &= |V(G)| \\ n_i &= |\{j : (i,j) \in E(G)\}| \end{aligned}$$

Then the basic probability t_{ij} of a transition from vertex i to j is given by

$$t_{ij} = \begin{cases} 1/n_i & \text{if } n_i \neq 0 \text{ and } (i,j) \in E(G) \\ 1/(n-1) & \text{if } n_i = 0 \text{ and } i \neq j \quad (\text{but see footnote})^1 \\ 0 & \text{otherwise} \end{cases}$$

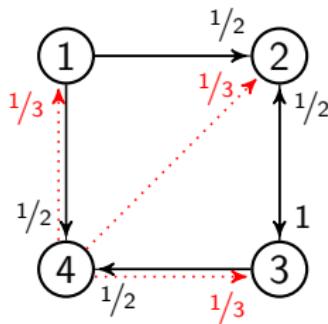
The basic transition matrix is $T = (t_{ij})_{1 \leq i,j \leq n}$.

¹When n is large (as in the WWW) this line can be simplified to: $1/n$ if $n_i = 0$.

Example 4A

Here is a tiny example of basic transition probabilities - with just $n = 4$ vertices :

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



$$T = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$$

Solving, by computer, $(T' - I)S = 0$ with the usual replacement last equation, gives steady state solution $S = \frac{1}{13} \begin{bmatrix} 1 \\ 4 \\ 5 \\ 3 \end{bmatrix} \approx \begin{bmatrix} .08 \\ .31 \\ .38 \\ .23 \end{bmatrix}$.

So on this basis, vertex 3 is most important and vertex 1 least.

The damping factor $(1 - \alpha)$

Recall that the PageRank algorithm assumes that, at any time k , there is a small probability α that, irrespective of what links are available at the current page, the surfer chooses to teleport randomly to any page on the web (including the current page).

The probability that the surfer takes the unforced teleport option and lands on any particular page is α/n .

Thus the modified probability for transition from vertex i to j is

$$m_{ij} = \alpha/n + (1 - \alpha)t_{ij}.$$

In practice, Google uses a damping factor of 85%, i.e. $\alpha = 0.15$.

The modified transition matrix M and PageRank vector PR

The modified transition probabilities lead to a modified transition matrix

$$\begin{aligned} M = (m_{ij})_{1 \leq i,j \leq n} &= (\alpha/n + (1 - \alpha)t_{ij})_{1 \leq i,j \leq n} \\ &= (\alpha/n)U + (1 - \alpha)T, \end{aligned}$$

where U is the $n \times n$ all-1's matrix and T is the basic transition matrix.

As indicated earlier, the PageRank algorithm defines the rank of page i of the webgraph to be the i -th entry in the **PageRank vector** PR , which in turn is defined as the steady state vector for the random walk on the webgraph with transition matrix M .

Thus PR is defined as the probability vector solution to the equation

$$M'\text{PR} = \text{PR}.$$

Calculating PR

Expanding the defining equation $M'PR = PR$ gives

$$\begin{aligned} PR &= ((\alpha/n)U + (1 - \alpha)T')PR \quad \text{since } U' = U \\ &= (\alpha/n)UPR + (1 - \alpha)T'PR \quad (***) \end{aligned}$$

Now each entry of the product UPR is the sum of all the entries in PR , and since PR is a probability vector that sum is 1. Hence $UPR = \mathbf{1}$ where $\mathbf{1}$ is the $n \times 1$ vector of all 1's. So equation $(**)$ can be rearranged as

$$(I - (1 - \alpha)T')PR = (\alpha/n)\mathbf{1}$$

For small to moderate n this equation can be solved directly (by computer).

- Notes:
- When $\alpha = 0$ this equation reverts to the basic steady state equation.
 - When $\alpha \neq 0$ the fact that we have used $UPR = \mathbf{1}$ means that it is no longer necessary, nor appropriate, to replace the last row of this matrix equation by all 1's, as in the case of $\alpha = 0$.

Example 4B

For example 4A we had:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$$

$$S = \begin{bmatrix} .08 \\ .31 \\ .38 \\ .23 \end{bmatrix}$$

Let's see what happens if we apply a damping factor of 90%; i.e. $\alpha = 0.1$.

We need to solve the equation $(I - (1 - \alpha)T')R = (\alpha/n)\mathbf{1}$ where:

$$(I - (1 - \alpha)T') = I - (0.9)T'$$

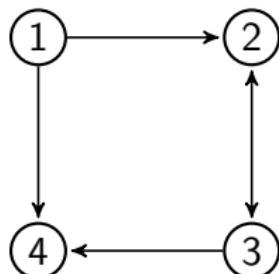
$$= \begin{bmatrix} 1 & 0 & 0 & -0.3 \\ -0.45 & 1 & -0.45 & -0.3 \\ 0 & -0.9 & 1 & -0.3 \\ -0.45 & 0 & -0.45 & 1 \end{bmatrix}$$

$$(\alpha/n)\mathbf{1} = \frac{0.1}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.025 \\ 0.025 \\ 0.025 \\ 0.025 \end{bmatrix}$$

Example 4B (cont.)

Solving the equation

$$\begin{bmatrix} 1 & 0 & 0 & -0.3 \\ -0.45 & 1 & -0.45 & -0.3 \\ 0 & -0.9 & 1 & -0.3 \\ -0.45 & 0 & -0.45 & 1 \end{bmatrix} PR = \begin{bmatrix} 0.025 \\ 0.025 \\ 0.025 \\ 0.025 \end{bmatrix}$$

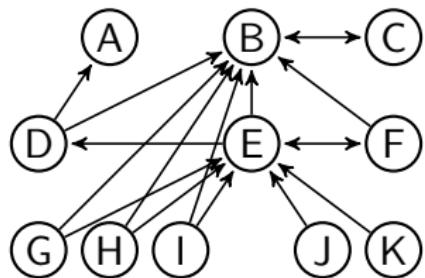


on the computer gives, to two decimal places,

$$PR = \begin{bmatrix} .10 \\ .30 \\ .37 \\ .23 \end{bmatrix} \text{ compared to } S = \begin{bmatrix} .08 \\ .31 \\ .38 \\ .23 \end{bmatrix} \text{ without damping.}$$

The PageRank of vertex 1 increases because it now has teleporting ‘inputs’ from all vertices, not just vertex 4. This increase is at the expense of the stronger vertices 2 and 3. Vertex 4 gains about as much as it loses. This is what you expect with damping.

Example 5



At left is the Wikipedia example we saw earlier of a miniweb of 11 pages and 17 hyperlinks. Colours, variable sizes and PageRanks have been removed and the bottom five vertices have been labelled G to K, following the given labelling of the top six vertices. The layout is similar to that in the original.

Using the steady state method we have been discussing, we will derive the PageRanks given on the Wikipedia diagram.

Step 1: Compile the adjacency matrix A .

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Example 5 (cont.)

Step 2: Compile the basic transition matrix T .

In each row i of A , count the number n_i of 1's in the row then:

- If $n_i \neq 0$ replace each 1 with $1/n_i$.
- If $n_i = 0$ then
 - if n (the total number of pages) is small (less than 10 say) replace all but the i -th (diagonal) entry by $1/(n-1)$ (we did this in Example 4B)
 - but for $n \geq 10$ (as here and for WWW) replace every entry by $1/n$.

For our Wikipedia example we get

$$T = \begin{bmatrix} 1/11 & 1/11 & 1/11 & 1/11 & 1/11 & 1/11 & 1/11 & 1/11 & 1/11 & 1/11 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Example 5 (cont.)

Step 3: Compile the matrix $(I - (1 - \alpha)T')$

For each row i of T

1. multiply each entry by $(1 - \alpha)$ and put the *negative* of this in the corresponding position in the i -th *column* of $(I - (1 - \alpha)T')$;
 2. add 1 to each diagonal entry of the resulting matrix.

Google uses an 85% damping factor, so we set $(1 - \alpha) = 0.85$.

$$(I - (1 - \alpha)T') =$$

Example 5 (cont.)

Step 4: Compile vector $(\alpha/n)\mathbf{1}$.

Since $(1 - \alpha) = .85$ use $\alpha = .15$.

Thus $\alpha/n = \frac{15}{11} = .01\overline{36}$ and hence

$$(\alpha/n)\mathbf{1} = \begin{bmatrix} .01\overline{36} \\ .01\overline{36} \end{bmatrix}$$

Step 5: Use a computer to solve $(I - (1 - \alpha)T')PR = (\alpha/n)\mathbf{1}$

For example 'Gauss-Jordan Elimination' in the *Matrix Reshish* online matrix calculator, gives

$$PR = \begin{bmatrix} 0.032919 \\ 0.384644 \\ 0.343127 \\ 0.039112 \\ 0.080937 \\ 0.039112 \\ 0.016180 \\ 0.016180 \\ 0.016180 \\ 0.016180 \\ 0.016180 \\ 0.016180 \end{bmatrix} \approx \begin{bmatrix} 3.3\% \\ 38.4\% \\ 34.3\% \\ 3.9\% \\ 8.1\% \\ 3.9\% \\ 1.6\% \\ 1.6\% \\ 1.6\% \\ 1.6\% \\ 1.6\% \\ 1.6\% \end{bmatrix}$$

Iterative Approximation method

When n is huge, as it is with the WWW, solving the the $n \times n$ linear system $(I - (1 - \alpha)T')PR = (\alpha/n)\mathbf{1}$ becomes computationally infeasible.

A computationally simpler method starts from the defining equation:

$$M'PR = PR.$$

Recall that $M = (\alpha/n)U + (1 - \alpha)T$ is the modified transition matrix.

As with Markov chains in general we can attempt to find PR by iteratively calculating the chain of probability vectors P_0, P_1, \dots where P_0 is arbitrary and $P_k = M'P_{k-1}$ for $k \geq 1$ (so $P_k = (M')^k P_0$).

A steady state is reached when $P_k \approx P_{k-1}$. Then declare that $PR \approx P_k$.

$$\begin{aligned} P_k &= M'P_{k-1} = [(\alpha/n)U + (1 - \alpha)T']P_{k-1} \\ &= (\alpha/n)\mathbf{1} + (1 - \alpha)T'P_{k-1} \end{aligned}$$

and it is natural to start with all ranks equal. So the iterative scheme is

$$P_0 = (1/n)\mathbf{1}; \quad P_k = \alpha P_0 + (1 - \alpha)T'P_{k-1}, \quad k \geq 1.$$

Each iteration takes a weighted average of teleporting and hyperlinking.

Example 4C

In Example 4B we used the equation-solving method to find PR .

For $T = \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 1/3 & 1/3 & 1/3 & 0 \end{bmatrix}$ and $\alpha = 0.1$ we found $PR = \begin{bmatrix} .10 \\ .30 \\ .37 \\ .23 \end{bmatrix}$ to 2d.p.

Let's try the same problem using iterative approximation:

$$P_0 = \begin{bmatrix} .25 \\ .25 \\ .25 \\ .25 \end{bmatrix} \quad P_k = \begin{bmatrix} .025 \\ .025 \\ .025 \\ .025 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & .3 \\ .45 & 0 & .45 & .3 \\ 0 & .9 & 0 & .3 \\ .45 & 0 & .45 & 0 \end{bmatrix} P_{k-1}$$

Results of the first ten iterations, rounded to 2d.p. Calcs used 15d.p.

k	1	2	3	4	5	6	7	8	9	10
P_k	[.10]	[.10]	[.09]	[.10]	[.09]	[.10]	[.09]	[.10]	[.09]	[.10]
	.33	.29	.31	.30	.31	.30	.31	.30	.30	.30
	.33	.39	.35	.38	.36	.37	.36	.37	.37	.37
	.25	.22	.25	.22	.24	.23	.24	.23	.24	.23

Pretty good after just 2 iterations! Within 1%-point after 4 iterations.

END OF SECTION D3

Revisiting sets

The first five slides were prepared by Pierre Portal and Malcolm Brooks; the next three are based very closely on the presentation in our optional text; blame Adam Piggott for the slides after that.

Text Reference (Epp) 5ed: Section 6.4

The Barber Puzzle

In a certain town, there is a barber who shaves all those townsfolk, and only those townsfolk, who do not shave themselves. Does the barber shave themselves?

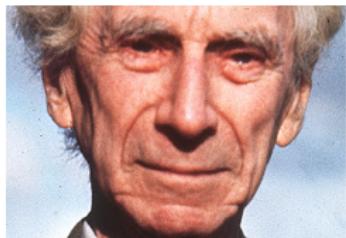
Let S denote the set of townsfolk who shave themselves. Either the barber is a member of S (they shave themselves), or the barber is not a member of S (they do not shave themselves). We consider cases.

Consider first the case that the barber is a member of S . Then they are a member of the set of townsfolk who shave themselves. But no member of this set is shaved by the barber. We have a contradiction. So this case is impossible.

Next consider the case that the barber is not a member of S . Then they are not a member of the set of townsfolk who shave themselves. The barber shaves every member of this set. We have a contradiction. So this case is impossible too...This is a problem!

(Bertrand) Russell's paradox

(within naive set theory)

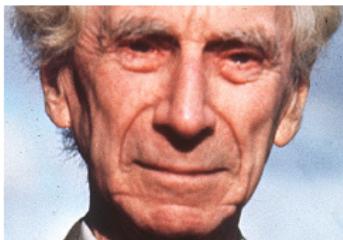


Bertrand Russell 1872 – 1970

(Bertrand) Russell's paradox

(within naive set theory)

Most sets are not members of themselves.

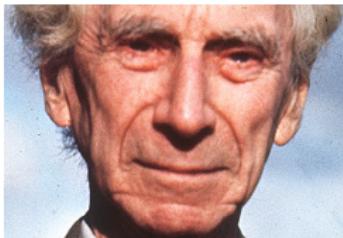


Bertrand Russell 1872 – 1970

(Bertrand) Russell's paradox (within naive set theory)

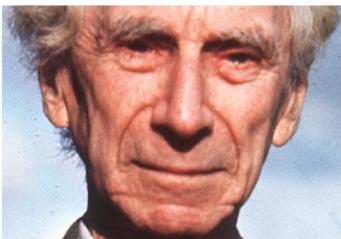
Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
(since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot.}$)



Bertrand Russell 1872 – 1970

(Bertrand) Russell's paradox (within naive set theory)



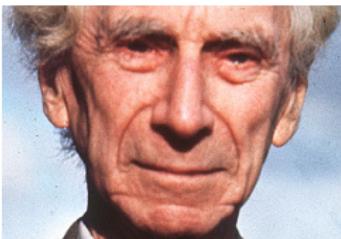
Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
(since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot.}$)

Call a set **regular** if (and only if) it is not a member of itself.

(Bertrand) Russell's paradox (within naive set theory)



Bertrand Russell 1872 – 1970

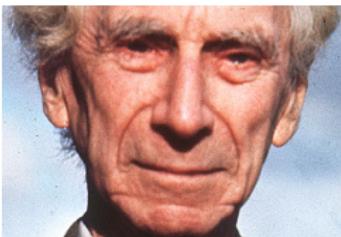
Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
(since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

(Bertrand) Russell's paradox (within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

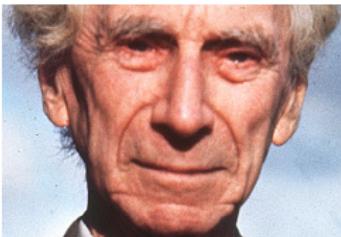
E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
(since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

(Bertrand) Russell's paradox (within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
(since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

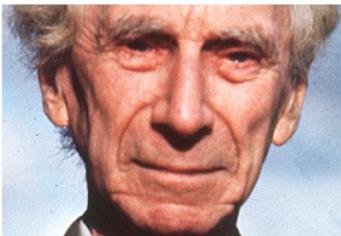
Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

Now let \mathcal{R} be the set of all regular sets .

(Bertrand) Russell's paradox (within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
(since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

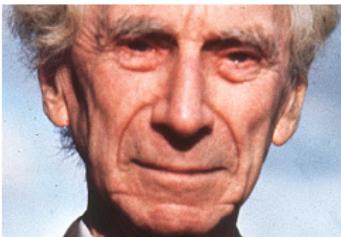
E.g.: The set of all sets. The set of all things that are not birds.

Now let \mathcal{R} be the set of all regular sets .

Question: Is \mathcal{R} itself regular?

(Bertrand) Russell's paradox

(within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
 (since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

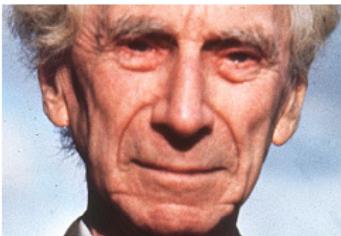
Now let \mathcal{R} be the set of all regular sets.

Question: **Is \mathcal{R} itself regular?**

\mathcal{R} not regular $\implies \mathcal{R}$ is a member of itself

(Bertrand) Russell's paradox

(within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
 (since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

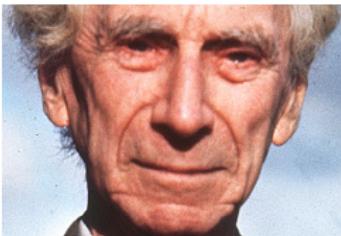
Now let \mathcal{R} be the set of all regular sets.

Question: **Is \mathcal{R} itself regular?**

\mathcal{R} not regular $\implies \mathcal{R}$ is a member of itself $\implies \mathcal{R} \in \mathcal{R}$

(Bertrand) Russell's paradox

(within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
 (since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

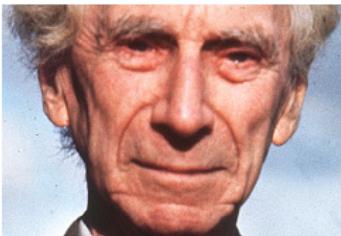
Now let \mathcal{R} be the set of all regular sets.

Question: **Is \mathcal{R} itself regular?**

\mathcal{R} not regular $\implies \mathcal{R}$ is a member of itself $\implies \mathcal{R} \in \mathcal{R}$
 $\implies \mathcal{R} \in$ the set of all regular sets

(Bertrand) Russell's paradox

(within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
 (since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

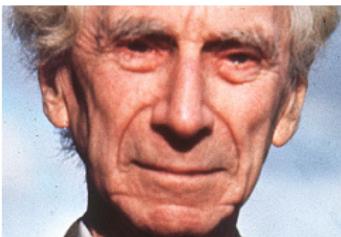
Now let \mathcal{R} be the set of all regular sets.

Question: **Is \mathcal{R} itself regular?**

\mathcal{R} not regular $\implies \mathcal{R}$ is a member of itself $\implies \mathcal{R} \in \mathcal{R}$
 $\implies \mathcal{R} \in$ the set of all regular sets $\implies \mathcal{R}$ is regular

(Bertrand) Russell's paradox

(within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
 (since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

Now let \mathcal{R} be the set of all regular sets.

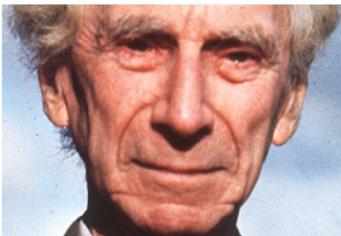
Question: **Is \mathcal{R} itself regular?**

\mathcal{R} not regular $\implies \mathcal{R}$ is a member of itself $\implies \mathcal{R} \in \mathcal{R}$
 $\implies \mathcal{R} \in$ the set of all regular sets $\implies \mathcal{R}$ is regular

\mathcal{R} is regular $\implies \mathcal{R}$ is not a member of itself

(Bertrand) Russell's paradox

(within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
 (since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

Now let \mathcal{R} be the set of all regular sets.

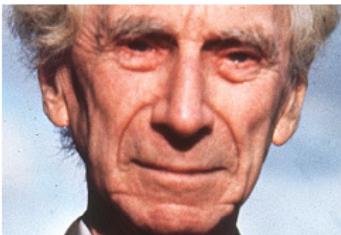
Question: **Is \mathcal{R} itself regular?**

\mathcal{R} not regular $\implies \mathcal{R}$ is a member of itself $\implies \mathcal{R} \in \mathcal{R}$
 $\implies \mathcal{R} \in$ the set of all regular sets $\implies \mathcal{R}$ is regular

\mathcal{R} is regular $\implies \mathcal{R}$ is not a member of itself $\implies \mathcal{R} \notin \mathcal{R}$

(Bertrand) Russell's paradox

(within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
 (since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

Now let \mathcal{R} be the set of all regular sets.

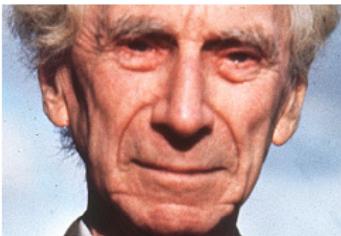
Question: **Is \mathcal{R} itself regular?**

\mathcal{R} not regular $\implies \mathcal{R}$ is a member of itself $\implies \mathcal{R} \in \mathcal{R}$
 $\implies \mathcal{R} \in$ the set of all regular sets $\implies \mathcal{R}$ is regular

\mathcal{R} is regular $\implies \mathcal{R}$ is not a member of itself $\implies \mathcal{R} \notin \mathcal{R}$
 $\implies \mathcal{R} \notin$ the set of all regular sets

(Bertrand) Russell's paradox

(within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
 (since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

Now let \mathcal{R} be the set of all regular sets.

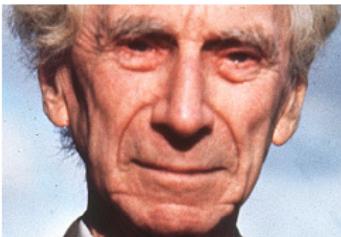
Question: **Is \mathcal{R} itself regular?**

\mathcal{R} not regular $\implies \mathcal{R}$ is a member of itself $\implies \mathcal{R} \in \mathcal{R}$
 $\implies \mathcal{R} \in$ the set of all regular sets $\implies \mathcal{R}$ is regular

\mathcal{R} is regular $\implies \mathcal{R}$ is not a member of itself $\implies \mathcal{R} \notin \mathcal{R}$
 $\implies \mathcal{R} \notin$ the set of all regular sets $\implies \mathcal{R}$ not regular

(Bertrand) Russell's paradox

(within naive set theory)



Bertrand Russell 1872 – 1970

Most sets are not members of themselves.

E.g. If $P = \{\text{pigeon, parrot}\}$ then $P \notin P$
 (since $\{\text{pigeon, parrot}\} \neq \text{pigeon}$, and $\{\text{pigeon, parrot}\} \neq \text{parrot}$.)

Call a set **regular** if (and only if) it is not a member of itself.

But some sets **are** members of themselves.

E.g.: The set of all sets. The set of all things that are not birds.

Now let \mathcal{R} be the set of all regular sets.

Question: **Is \mathcal{R} itself regular?**

\mathcal{R} not regular $\implies \mathcal{R}$ is a member of itself $\implies \mathcal{R} \in \mathcal{R}$
 $\implies \mathcal{R} \in$ the set of all regular sets $\implies \mathcal{R}$ is regular

\mathcal{R} is regular $\implies \mathcal{R}$ is not a member of itself $\implies \mathcal{R} \notin \mathcal{R}$
 $\implies \mathcal{R} \notin$ the set of all regular sets $\implies \mathcal{R}$ not regular

A contradiction either way! (paradox)

Naive set theory fails!

Axiomatic Set Theory

To avoid problems such as Russell's paradox we need to move to a more formal approach. The most classical is an axiomatic system called ZFC (Zermelo-Fraenkel with choice). We will not study it in this course, but will have assumed some of its axioms.

Axiomatic Set Theory

To avoid problems such as Russell's paradox we need to move to a more formal approach. The most classical is an axiomatic system called ZFC (Zermelo-Fraenkel with choice). We will not study it in this course, but will have assumed some of its axioms. In particular:

Axiom 1: A set T can only be defined as a subset of a known set U .

Axiomatic Set Theory

To avoid problems such as Russell's paradox we need to move to a more formal approach. The most classical is an axiomatic system called ZFC (Zermelo-Fraenkel with choice). We will not study it in this course, but will have assumed some of its axioms. In particular:

Axiom 1: A set T can only be defined as a subset of a known set U . That is, the definition must have the form:

$$T = \{x \in U ; p(x)\} \quad \text{where the domain of predicate } p \text{ includes all elements of } U.$$

Axiomatic Set Theory

To avoid problems such as Russell's paradox we need to move to a more formal approach. The most classical is an axiomatic system called ZFC (Zermelo-Fraenkel with choice). We will not study it in this course, but will have assumed some of its axioms. In particular:

Axiom 1: A set T can only be defined as a subset of a known set U . That is, the definition must have the form:

$$T = \{x \in U ; p(x)\} \quad \text{where the domain of predicate } p \text{ includes all elements of } U.$$

Example:

The phrase "the set of all birds" does not define a set in ZFC.

Axiomatic Set Theory

To avoid problems such as Russell's paradox we need to move to a more formal approach. The most classical is an axiomatic system called ZFC (Zermelo-Fraenkel with choice). We will not study it in this course, but will have assumed some of its axioms. In particular:

Axiom 1: A set T can only be defined as a subset of a known set U . That is, the definition must have the form:

$$T = \{x \in U ; p(x)\} \quad \text{where the domain of predicate } p \text{ includes all elements of } U.$$

Example:

The phrase "the set of all birds" does not define a set in ZFC.

But, if we have managed to define A as the set of all animals, then "the sets of all animals that are birds" does define a set in ZFC:

$$A: \text{set of all animals.} \qquad B = \{b \in A ; b \text{ is a bird}\} \subseteq A.$$

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$\mathcal{R}_{\mathcal{U}}$ not regular $\implies \mathcal{R}_{\mathcal{U}}$ is a member of itself

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$\mathcal{R}_{\mathcal{U}}$ not regular $\implies \mathcal{R}_{\mathcal{U}}$ is a member of itself $\implies \mathcal{R}_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}}$

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ not regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is a member of itself} \implies \mathcal{R}_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \in \{S \in \mathcal{U} ; S \text{ is regular}\} \end{aligned}$$

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$\mathcal{R}_{\mathcal{U}}$ not regular $\implies \mathcal{R}_{\mathcal{U}}$ is a member of itself $\implies \mathcal{R}_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}}$
 $\implies \mathcal{R}_{\mathcal{U}} \in \{S \in \mathcal{U} ; S \text{ is regular}\}$
 $\implies \mathcal{R}_{\mathcal{U}}$ is regular Contradiction!

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$\mathcal{R}_{\mathcal{U}}$ not regular $\implies \mathcal{R}_{\mathcal{U}}$ is a member of itself $\implies \mathcal{R}_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}}$
 $\implies \mathcal{R}_{\mathcal{U}} \in \{S \in \mathcal{U} ; S \text{ is regular}\}$
 $\implies \mathcal{R}_{\mathcal{U}}$ is regular Contradiction!

$\mathcal{R}_{\mathcal{U}}$ regular $\implies \mathcal{R}_{\mathcal{U}}$ is not a member of itself

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ not regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is a member of itself} \implies \mathcal{R}_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \in \{S \in \mathcal{U} ; S \text{ is regular}\} \\ &\implies \mathcal{R}_{\mathcal{U}} \text{ is regular} \quad \text{Contradiction!} \end{aligned}$$

$$\mathcal{R}_{\mathcal{U}} \text{ regular} \implies \mathcal{R}_{\mathcal{U}} \text{ is not a member of itself} \implies \mathcal{R}_{\mathcal{U}} \notin \mathcal{R}_{\mathcal{U}}$$

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ not regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is a member of itself} \implies \mathcal{R}_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \in \{S \in \mathcal{U} ; S \text{ is regular}\} \\ &\implies \mathcal{R}_{\mathcal{U}} \text{ is regular} \quad \text{Contradiction!} \end{aligned}$$

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is not a member of itself} \implies \mathcal{R}_{\mathcal{U}} \notin \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \notin \{S \in \mathcal{U} ; S \text{ is regular}\} \end{aligned}$$

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ not regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is a member of itself} \implies \mathcal{R}_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \in \{S \in \mathcal{U} ; S \text{ is regular}\} \\ &\implies \mathcal{R}_{\mathcal{U}} \text{ is regular} \quad \text{Contradiction!} \end{aligned}$$

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is not a member of itself} \implies \mathcal{R}_{\mathcal{U}} \notin \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \notin \{S \in \mathcal{U} ; S \text{ is regular}\} \\ &\implies (\mathcal{R}_{\mathcal{U}} \notin \mathcal{U}) \vee (\mathcal{R}_{\mathcal{U}} \text{ is not regular}) \end{aligned}$$

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ not regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is a member of itself} \implies \mathcal{R}_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \in \{S \in \mathcal{U} ; S \text{ is regular}\} \\ &\implies \mathcal{R}_{\mathcal{U}} \text{ is regular} \quad \text{Contradiction!} \end{aligned}$$

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is not a member of itself} \implies \mathcal{R}_{\mathcal{U}} \notin \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \notin \{S \in \mathcal{U} ; S \text{ is regular}\} \\ &\implies (\mathcal{R}_{\mathcal{U}} \notin \mathcal{U}) \vee (\mathcal{R}_{\mathcal{U}} \text{ is not regular}) \end{aligned}$$

Since $[(\mathcal{R}_{\mathcal{U}} \text{ regular}) \implies (\mathcal{R}_{\mathcal{U}} \text{ is not regular})]$ is contradictory we conclude that $\mathcal{R}_{\mathcal{U}}$ is regular but not in \mathcal{U} .

Russell's paradox removed (for interest only)

In ZFC we can still say that a set S is regular if and only if it satisfies the condition (predicate) $S \notin S$.

However we cannot define \mathcal{R} as the set of *all* regular sets.
Instead, for any *known set of sets* \mathcal{U} we can define $\mathcal{R}_{\mathcal{U}}$ by

$$\mathcal{R}_{\mathcal{U}} = \{S \in \mathcal{U} ; S \text{ is regular}\} = \{S \in \mathcal{U} ; S \notin S\}$$

So we again ask: **Is $\mathcal{R}_{\mathcal{U}}$ itself regular?**

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ not regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is a member of itself} \implies \mathcal{R}_{\mathcal{U}} \in \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \in \{S \in \mathcal{U} ; S \text{ is regular}\} \\ &\implies \mathcal{R}_{\mathcal{U}} \text{ is regular} \quad \text{Contradiction!} \end{aligned}$$

$$\begin{aligned} \mathcal{R}_{\mathcal{U}} \text{ regular} &\implies \mathcal{R}_{\mathcal{U}} \text{ is not a member of itself} \implies \mathcal{R}_{\mathcal{U}} \notin \mathcal{R}_{\mathcal{U}} \\ &\implies \mathcal{R}_{\mathcal{U}} \notin \{S \in \mathcal{U} ; S \text{ is regular}\} \\ &\implies (\mathcal{R}_{\mathcal{U}} \notin \mathcal{U}) \vee (\mathcal{R}_{\mathcal{U}} \text{ is not regular}) \end{aligned}$$

Since $[(\mathcal{R}_{\mathcal{U}} \text{ regular}) \implies (\mathcal{R}_{\mathcal{U}} \text{ is not regular})]$ is contradictory we conclude that $\mathcal{R}_{\mathcal{U}}$ is regular but not in \mathcal{U} . No overall contradiction.

The Halting Problem

Theorem (Alan Turing, 1936): There is no algorithm that will accept any algorithm X and data set D as input and then will output “halts” or “loops forever” to indicate whether or not X terminates in a finite number of steps when X is run with data set D

Proof: We shall use a proof by contradiction. Suppose that there is an algorithm, CheckHalt, such that if algorithm X and a data set D are input, then:

- $\text{CheckHalt}(X, D)$ prints “halts” if X terminates in a finite number of steps when run with data set D ;
- $\text{CheckHalt}(X, D)$ prints “loops forever” if X does not terminate in a finite number of steps when run with data set D .

The Halting Problem (continued)

A sequence of characters making up an algorithm X can be regarded as a data set itself. So it is possible to call $\text{CheckHalt}(X, X)$.

Let Test be a new algorithm that takes as input an algorithm X and so that:

- $\text{Test}(X)$ loops forever if $\text{CheckHalt}(X, X)$ prints “halts”;
- $\text{Test}(X)$ stops if $\text{CheckHalt}(X, X)$ prints “loop forever”.

(The existence of such an algorithm follows immediately the existence of CheckHalt .)

Now consider what happens when we run the algorithm Test with the input Test . Either it terminates after a finite number of steps or it loops forever.

The Halting Problem (continued)

Consider first the case that $\text{Test}(\text{Test})$ terminates after a finite number of steps. Then $\text{CheckHalt}(\text{Test}, \text{Test})$ prints “halts” and so $\text{Test}(\text{Test})$ loops forever. This is a contradiction.

Consider next the case that $\text{Test}(\text{Test})$ loops forever. Then $\text{CheckHalt}(\text{Test}, \text{Test})$ prints “loops forever” and so $\text{Test}(\text{Test})$ terminates in a finite number of steps. This is also a contradiction.

In each case, we reached a contradiction. Our supposition that an algorithm such as CheckHalt exists allowed us to deduce a false statement. It is therefore false itself. \square

An example to pull some ideas together

In parallel computing, multiple CPUs are connected into a network. Each CPU gets busy working on different parts of a problem. Occasionally, the CPUs need to communicate.

PROBLEM: How can we connect many CPUs into network so that:

1. We don't use too many connections, because connections cost money and take up physical space
2. When one CPU needs to communicate with another, the message does not need to be passed through too many intermediary CPUs
3. CPUs are labelled in a logical way that allows us to write easily write algorithms for getting messages around the network.

A possible solution

Connect your CPUs so as to make a hypercube.

For each $n \in \{1, 2, 3, \dots\}$, the **hypercube** H_n is the graph H_n such that

- The vertex set of H_n is $\{0, 1\}^n$; that is, the vertex set of H_n is the set of bit strings of length n
- Two vertices of H_n are adjacent if and only if they differ in exactly one bit.

Getting comfortable with H_n

When trying to become comfortable with a new family of objects, it helps to spend some time building intuition...just thinking about them

Let's try drawing some hypercubes

The vertex set of H_1 is $\{0, 1\}$. We can identify the vertices of H_1 as points in \mathbb{R} (the number line), and then draw the edge between them to get ... an interval.

The vertex set of H_2 is $\{00, 01, 10, 11\}$. We can identify the vertices of H_2 as the points $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$ in \mathbb{R}^2 (the Euclidean plane), and then draw the edges between them to get ... the frame of a square.

Getting comfortable with H_n (cont.)

The vertex set of H_3 is $\{000, 001, 010, 011, 100, 101, 110, 111\}$. We can identify the vertices of H_3 as the points

$$\{(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1), (1, 1, 0), (1, 1, 1)\}$$

in \mathbb{R}^3 , and then draw the edges between them to get ... the frame of a cube.

What do we get when represent H_4 in \mathbb{R}^4 ?

Some questions

Let $n \in \mathbb{N}$.

1. How many vertices in H_n ?
2. What is the degree of each vertex in H_n ?
3. How many edges in H_n ?
4. What is the greatest distance between any pair of vertices in H_n ?
5. Find a Hamilton circuit in H_n , if one exists.

Now suppose that a parallel computing network has the structure of H_n . In this context, what is the importance of each question and answer above?

A Hamilton Circuit in H_n

We use the n -bit reflected binary code to build a Hamilton circuit in H_n .

0, 1, 0 is a Hamilton circuit on H_1 .

start with rbc for $n - 1$ reflect add 0's above and 1's below

0	0	00
1	1	01
	1	11
	0	10

Note that 00, 01, 11, 10, 00 is a Hamilton circuit in H_2 .

A Hamilton Circuit in H_n (cont.)

start with rbc for $n - 1$ reflect add 0's above and 1's below

00	00	000
01	01	001
11	11	011
10	10	010
	10	110
	11	111
	01	101
	00	100

Note that 000, 001, 011, 010, 110, 111, 101, 100, 000 is a Hamilton circuit in H_3 .

Pros and Cons

What do you think are the pros and cons of arranging your parallel computing network so that it has the structure of a hypercube?

Addendum to D2: A proof that Kruskal's Algorithm works

Text Reference (Epp) 5ed: Section 10.6

Some familiar definitions

A **tree** is a connected graph with no circuits other than the trivial ones.

A **forest** is a disjoint collection of trees.

Note that we may consider a tree to be a forest (with just one tree).
We shall do this. So a forest is a graph with no circuits.

Recall: Characterising Trees (D2 Slide 91)

Theorem: Let T be a graph with n vertices. The following statements are logically equivalent:

- (i) T is a tree (it is connected and has no non-trivial circuits).
- (ii) T has no simple circuits and $n - 1$ edges.
- (iii) T is connected and has $n - 1$ edges.
- (iv) T is connected and every edge is a bridge.
- (v) Any two vertices of T are connected by exactly one simple path.
- (vi) T contains no non-trivial circuits, but the addition of any new edge (connecting an existing pair of vertices) creates a simple circuit.

Kruskal's algorithm for minimal spanning tree

Kruskal's algorithm is an obvious modification to the spanning-tree-finding algorithm we discussed previously. We simply choose the 'cheapest' possible edge at each step:

Kruskal's algorithm for minimal spanning tree

Kruskal's algorithm is an obvious modification to the spanning-tree-finding algorithm we discussed previously. We simply choose the 'cheapest' possible edge at each step:

Input: Weighted connected graph G with n vertices.

Output: Minimal spanning tree T for G .

Total weight W of this tree.

Kruskal's algorithm for minimal spanning tree

Kruskal's algorithm is an obvious modification to the spanning-tree-finding algorithm we discussed previously. We simply choose the 'cheapest' possible edge at each step:

Input: Weighted connected graph G with n vertices.

Output: Minimal spanning tree T for G .

Total weight W of this tree.

Method: 1. Initialise T to have all the vertices of G but no edges. Initialise W to 0.

Kruskal's algorithm for minimal spanning tree

Kruskal's algorithm is an obvious modification to the spanning-tree-finding algorithm we discussed previously. We simply choose the 'cheapest' possible edge at each step:

Input: Weighted connected graph G with n vertices.

Output: Minimal spanning tree T for G .

Total weight W of this tree.

- Method:**
1. Initialise T to have all the vertices of G but no edges. Initialise W to 0.
 2. From the edges currently in G pick one, e , of least weight and remove it from G .

Kruskal's algorithm for minimal spanning tree

Kruskal's algorithm is an obvious modification to the spanning-tree-finding algorithm we discussed previously. We simply choose the 'cheapest' possible edge at each step:

Input: Weighted connected graph G with n vertices.

Output: Minimal spanning tree T for G .

Total weight W of this tree.

- Method:**
1. Initialise T to have all the vertices of G but no edges. Initialise W to 0.
 2. From the edges currently in G pick one, e , of least weight and remove it from G .
 3. If adding e to T does not create a circuit in T , add e to T and add $\text{weight}(e)$ to W .

Kruskal's algorithm for minimal spanning tree

Kruskal's algorithm is an obvious modification to the spanning-tree-finding algorithm we discussed previously. We simply choose the 'cheapest' possible edge at each step:

Input: Weighted connected graph G with n vertices.

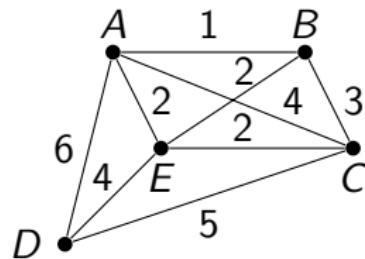
Output: Minimal spanning tree T for G .

Total weight W of this tree.

- Method:**
1. Initialise T to have all the vertices of G but no edges. Initialise W to 0.
 2. From the edges currently in G pick one, e , of least weight and remove it from G .
 3. If adding e to T does not create a circuit in T , add e to T and add $\text{weight}(e)$ to W .
 4. Repeat steps 2 and 3 until T has $n - 1$ edges.

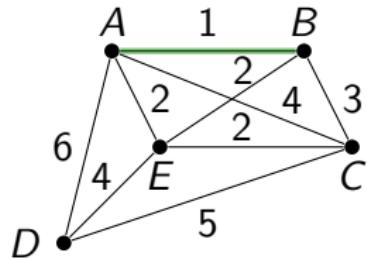
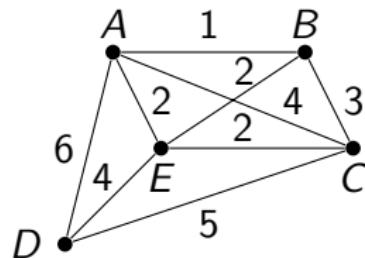
Example: Applying Kruskal's algorithm

Find a minimal spanning tree for this weighted graph:



Example: Applying Kruskal's algorithm

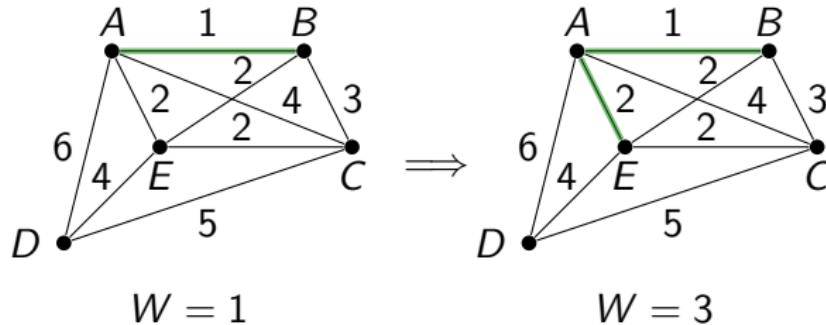
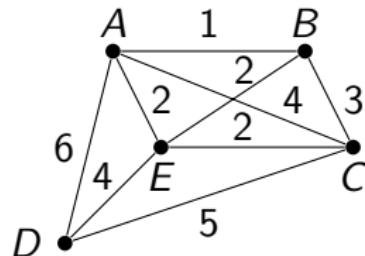
Find a minimal spanning tree for this weighted graph:



$$W = 1$$

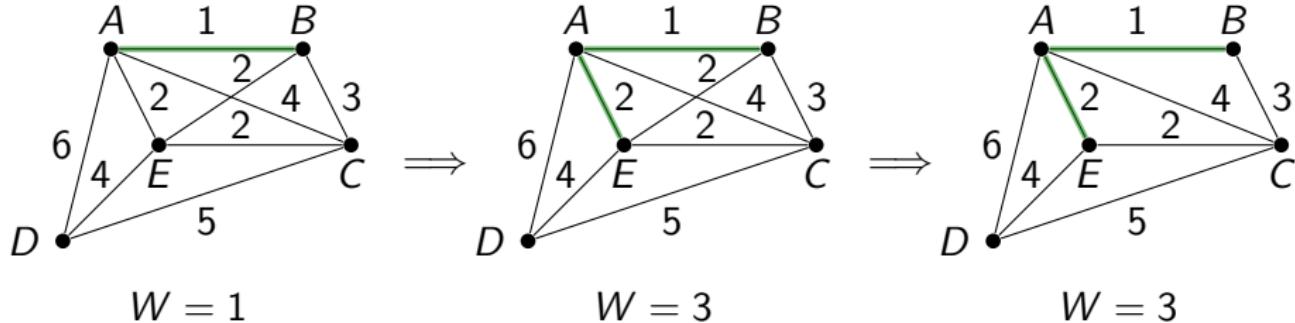
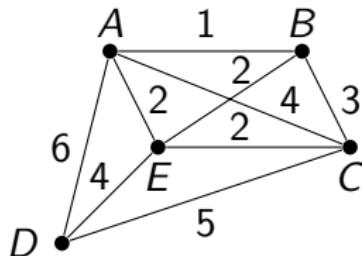
Example: Applying Kruskal's algorithm

Find a minimal spanning tree for this weighted graph:

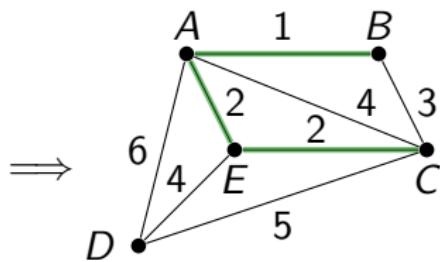


Example: Applying Kruskal's algorithm

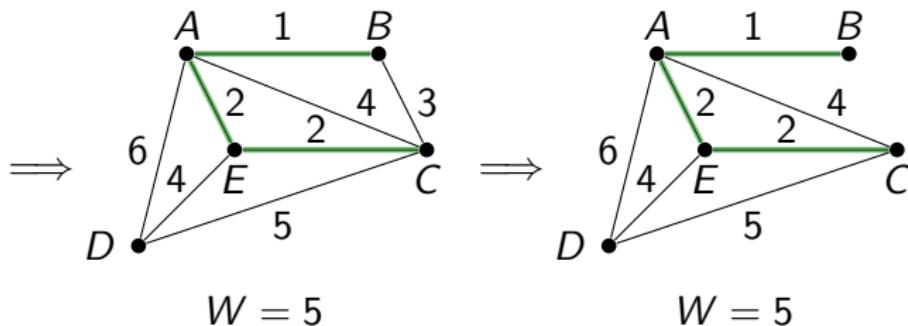
Find a minimal spanning tree for this weighted graph:



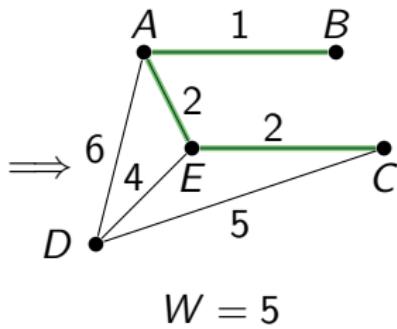
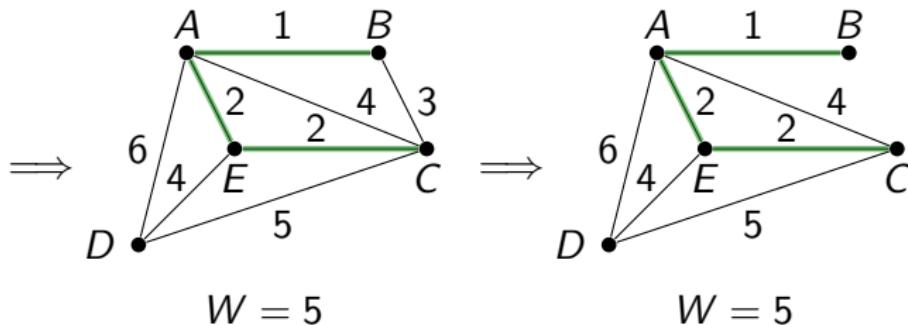
Example: Applying Kruskal's algorithm (cont.)



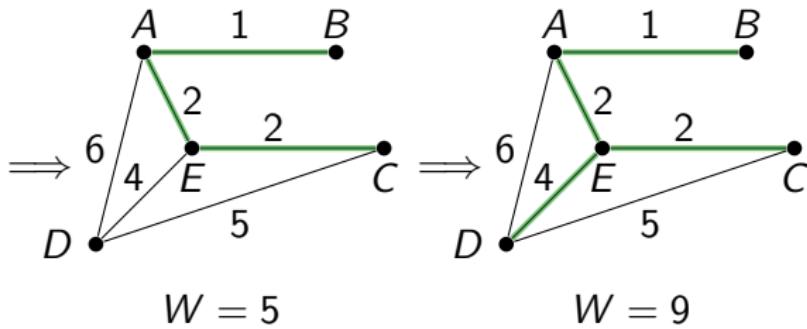
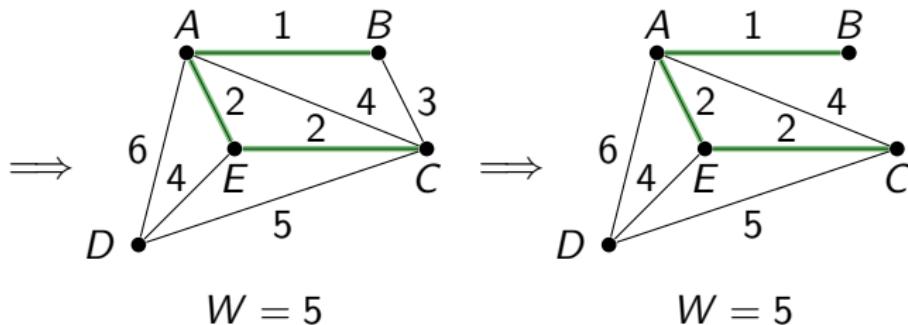
Example: Applying Kruskal's algorithm (cont.)



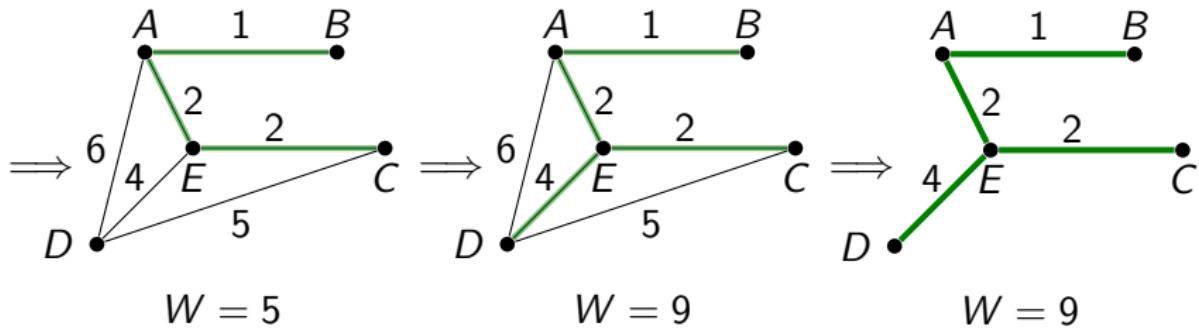
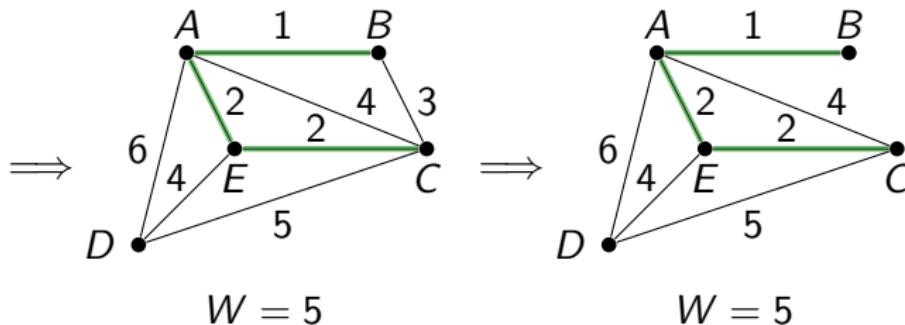
Example: Applying Kruskal's algorithm (cont.)



Example: Applying Kruskal's algorithm (cont.)



Example: Applying Kruskal's algorithm (cont.)



(The generated spanning tree)

Modified Kruskal's algorithm for minimal spanning tree

Input: Weighted connected graph G with n vertices.

Output: Minimal spanning tree T for G .

Total weight W of this tree.

- Method:**
1. Initialise T to have all the vertices of G but no edges. Initialise W to 0.
 2. From the edges currently in G pick one, e , of least weight and remove it from G .
 3. If adding e to T does not create a circuit in T , add e to T and add $\text{weight}(e)$ to W .
 4. Repeat steps 2 and 3 **until** G **has no edges left to consider.**

A lemma and a theorem

Lemma: Suppose that G is a connected, weighted graph. The output from Kruskal's algorithm on input G is the same as the output of the modified Kruskal's algorithm on input G .

Theorem: (The correctness of Kruskal's Algorithm) Suppose that G is a connected, weighted graph. The output of Kruskal's algorithm on input G is a minimal spanning tree T of G .

Correctness of (modified) Kruskal's Algorithm

Proof: Suppose that G is a connected, weighted graph with n vertices and that $T = T_m$ is the subgraph of G produced when G is input to the modified Kruskal's algorithm. We shall prove the following, in sequence:

1. T_m is a forest
2. T_m is connected
3. T_m is a spanning tree of G
4. T_m is a minimal spanning tree of G .

By the Lemma on the previous slide, this proves the theorem.

Correctness of (modified) Kruskal's Algorithm

We introduce some notation to help us talk about the algorithm, and what things look like during the execution.

Let m be the number of edges in G . Let (e_1, e_2, \dots, e_m) be the edges of G , in sequence, such that

$$\forall i \in \{1, 2, \dots, m-1\} \text{ weight}(e_i) \leq \text{weight}(e_{i+1}).$$

Without loss of generality, we may assume that the modified Kruskal's algorithm considers the edges in order e_1, e_2, \dots .

Let T_0 be T as it looks after step 1 (T_0 has all the vertices of G , but no edges). For $i = 1, 2, \dots, m$, let T_i be T as it looks after step 3 has been executed i times (that is, after e_i is considered).

Correctness of the (modified) Kruskal's Algorithm

Let $p(i)$: T_i is a forest.

We use mathematical induction to prove $\forall i \in \{0, 1, \dots, m\} p(i)$.

It is clear that T_0 is a graph with n vertices and 0 edges. Since T_0 has no edges, it is a forest (with n connected components). Hence $p(0)$ holds.

Let $i \in \{0, 1, \dots, m-1\}$, and suppose that $p(0), p(1), \dots, p(i)$ all hold. Since $p(i)$ is true, T_i is a forest. Since $i \leq m-1 < m$, step 2 of the algorithm will be executed for an $(i+1)$ -th time. The edge removed from G in step 2 is e_{i+1} . Step 3 of the algorithm will be executed an $(i+1)$ -th time. We consider cases, based on whether adding e_{i+1} to T_i creates a circuit.

Correctness of (modified) Kruskal's Algorithm

Consider first the case that adding e_{i+1} to T_i makes a circuit. Then $T_{i+1} = T_i$. Since T_i is a forest, T_{i+1} is a forest.

Consider next the case that adding e_{i+1} to T_i does not make a circuit. Then T_{i+1} is constructed from T_i by adding e_{i+1} . Since T_i is circuit-free, and adding e_{i+1} does not add a circuit, T_{i+1} is circuit-free. That is, T_{i+1} is a forest.

In each case, T_{i+1} is a forest. Hence $p(i + 1)$ is true.

By the principle of mathematical induction, $\forall i \in \{0, 1, \dots, m\} p(i)$. Since $p(m)$ is true, T_m is a forest.

Correctness of (modified) Kruskal's Algorithm

Next we show that T_m is a tree. Since we know that T_m is a forest, we need only show that T_m is connected. We shall use a proof by contradiction. Suppose that T_m is not connected. Let F_1, F_2, \dots, F_s be the connected components of G . Let $v_1 \in V(F_1)$ and let $v_2 \in V(F_2)$. Since G is connected, there exists a path α in G from v_1 to v_2 . Since v_2 is not in $V(F_1)$, α visits vertices not in $V(F_1)$. Let v_3 be the first vertex visited by α that is not in $V(F_1)$. Let e_j be the edge traversed by α to reach v_3 for the first time. Then one endpoint of e_j is in F_1 , and the other endpoint is not. Since the end-points of e_j lie in distinct connected components of T_m , and T_m is a forest, adding e_j to T_m will not create a circuit. Since T_{j-1} is a subgraph of T_m , adding e_j to T_{j-1} does not create a circuit. It follows that e_j will be added to T_{j-1} when executing step 3 for the j -th time. But then $e_j \in T_m$. This is a contradiction. Since our supposition that T_m is not connected allowed us to deduce a contradiction, it must be false. Hence T_m is connected.

Correctness of (modified) Kruskal's Algorithm

Since T_m is a connected forest it is a tree. Since T_0 contains every vertex from G , T_m does too. Hence T_m is a spanning tree of G .

Correctness of (modified) Kruskal's Algorithm

Now we show that T_m is a minimal spanning tree. We shall use a proof by contradiction. Suppose that T is not a minimal spanning tree. Let S be a minimal spanning tree for G such that the number of edges that T_m and G have in common is maximal among all minimal spanning trees (no minimal spanning tree of G shares more edges with T_m than S does—some might share different edges, but none share more).

Since S is a minimal spanning tree for G and T_m is not, the set $C = \{e \in E(T_m) \mid e \notin E(S)\}$ is not empty. Let k be the minimum integer such that $e_k \in C$. Since $e_k \notin E(S)$, adding e_k to S would add a circuit. Let e_ℓ be one of the edges in that circuit such that $e_\ell \notin T_m$ (such an edge must exist because T_m is circuit-free). Let S' be the graph obtained from S by adding e_k (creating a connected graph with a circuit that contains e_ℓ) and then removing e_ℓ . Removing an edge from a circuit in a connected graph leaves a connected graph, hence S' is connected. Since S' is a connected graph with n vertices and $n - 1$ edges, it is a tree.

Correctness of (modified) Kruskal's Algorithm

Suppose that $\text{weight}(e_\ell) < \text{weight}(e_k)$. Then $\ell < k$, and e_ℓ would have been considered for inclusion in T before e_k . By our choice of k , all of the edges in T_{k-1} are in S ; hence all of the edges in $T_{\ell-1}$ are in S . Since $e_\ell \in E(S)$ and S is circuit-free and $T_{\ell-1}$ is a subgraph of S , adding e_ℓ to $T_{\ell-1}$ does not create a circuit. Hence e_ℓ will be added to T when step 3 is executed for the ℓ -th time. Hence $e_\ell \in T_m$. But we chose $e_\ell \notin T_m$. This contradiction shows that $\text{weight}(e_\ell) \geq \text{weight}(e_k)$.

Since S' is constructed from S by removing e_ℓ and adding e_k , and $\text{weight}(e_\ell) \geq \text{weight}(e_k)$, we have that $\text{weight}(S) \geq \text{weight}(S')$. Since S is a minimal spanning tree, $\text{weight}(S') = \text{weight}(S)$. It follows that S' is a minimal spanning tree of G . But S' shares one more edge with T than S does, contradicting our choice of S . Since our supposition that T is not a minimal spanning tree allowed us to deduce a contradiction, it must be false. \square

Discussion

What was the point of that?