DM

Sets

A set is a collection of objects called the elements (or members of the set).

- There are no repeated occurrences of elements.
- There is no order of the elements

Notation for sets

- The elements of a set are enclosed in braces
  - A = {1, 2, 3}, C = {Portsmouth, Brighton, London}
- If S is a set and x is an element of S, then we write x ∈ S
  - 1 ∈ A, London ∈ C, 2, 3 ∈ A
  - and x ∈/ S, if x is not an element of S

Describing sets

1. by listing the elements, used mainly for finite sets
   a. A = {3, 6, 9, 12}
2. by specifying a property that the elements of the set have in common
   a. B = {x | x is a multiple of 3 and 0 < x < 15}
   b. '|' is read 'such that'

The sets of numbers

- N is used for the set of natural numbers:
  - N = {0, 1, 2, 3, 4, . . . }
- Z is used for the set of integers
  - Z = {. . . , −2, −1, 0, 1, 2, . . . }
- Q is used for the set of rational numbers
  - $\mathbb{Q} = \left\{0, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots\right\}$

Empty, finite and infinite sets

- The set with no element is called the empty set (or null set) and is denoted by ∅, ∅ = { }
- If counting of the elements at a fixed rate (e.g. one per second) of a set X can finish in a finite amount of time, then the set is finite.
  - If X is a finite set, we call |X| the cardinality of X
  - |X| = number of elements in X
- If the counting never stops, then X is an infinite set (no stopping condition when specifying a property)

Subsets

- If A and B are sets and every element of A is also an element of B, then we say that A is a subset of B and write
  - A ⊆ B
- If A is not a subset of B, we write A ⊆ B
- If A ⊆ B and there is some element in B that does not occur in A, then A is called a proper subset of B, A ⊂ B

FORMULA FOR THE NUMBER OF SUBSETS OF A SET

A set containing n distinct objects has 2^n subsets. (Includes null $\emptyset$)

Equality of sets

Two sets are equal if they have the same elements

- We denote the fact that two sets A and B are equal by writing
    - $A = B$

Operations on sets

Intersection

- The intersection of two sets A and B is
    - $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$
- If A and B are disjoint then $A \cap B = \emptyset$ (they have no elements in common).

Union

- The union of two sets A and B is
    - $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
    - Example. If $A = \{a, b, c\}$ and $B = \{c, d\}$, then $A \cup B = \{a, b, c, d\}$.

Difference

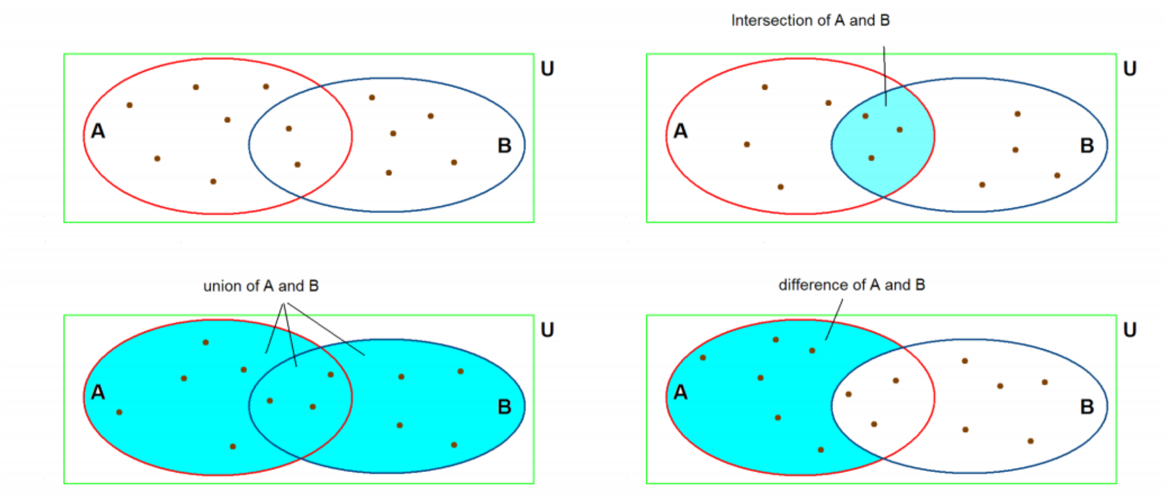
- The difference of two sets A and B is
- $A \setminus B = \{x \mid x \in A \text{ and } x \notin B\}$, (the same as $A - B$)

Complement of a set

If all considered subsets are the subsets of a particular set U (the universe of discourse), then the difference $U \setminus A$ is called the complement of A.

- The complement of A
    - $A' = \{x \mid x \in U \text{ and } x \notin A\}$
    - Example. If $U = \{a, b, c, d\}$ and $A = \{c, d\}$, then $A' = \{a, b\}$

Venn diagrams



Power Set

- The collection of all subsets of a set S is called the power set of S, denoted by P(S)
    - Example. If S = {a, b, c} then
    - P(S) = {∅, {a}, {b}, {c}, {a, b}, {b, c}, {a, c}, {a, b, c}}

Partition

- A collection of nonempty subsets of S is a partition of the set S is if every element in S belongs to exactly one member of S
    - Example. If S = {a, b, c, d, e, f } then {{a, e}, {c}, {f , d}, {b}} is a partition of S

Relations

Cartesian product

- If A and B are sets, we call A × B the Cartesian product of A and B
    - A × B = {(a, b) | a ∈ A and b ∈ B}
    - Example. If X = {1, 2, 3} and Y = {a, b}, then
    - X × Y = {(1, a),(1, b),(2, a),(2, b),(3, a),(3, b)}

Motivation example

- A is the set of all students in this class
- B is the set of all available units at the SOC department

Then the relation T can be defined between A and B:

- If the student x ∈ A is registered on the unit y ∈ B, then x is related to y by the relation T, e.g. (Stuart, MATHFUN) ∈ T
- The order matters, T is a relation from A to B
- T is a relation from A to B: T ⊆ A × B
    - T = {(Alice, DBPRIN), (Bryan, MATHFUN), (John, GUDE), . . . }

Relations "from – to" (a formal definition)

Let A and B be nonempty sets. A (binary) relation T from A to B is a subset of A × B.

If T ⊆ A×B and (a, b) ∈ T, we say that a is related to b by T, aTb.

Example

- Let A = {a, b, c, d, e} and B = {1, 2, 3}. Then
    - R1 = {(a, 1),(b, 1),(c, 2),(c, 3)}
    - R2 = {(a, 3),(a, 1),(c, 2),(c, 1),(b, 2)}

Describing relations

More often relations are described “by characteristics of their

elements”.

Example

- Let A = {1, 2} and B = {1, 2, 3} and define a binary relation R from A to B as follows:
    - x ∈ A is related to y ∈ B if and only if x 6 y.
    - Then (1, 3) ∈ R since 1 < 3, but (2, 1) ∈/ R since 2 6< 1
    - The elements of R are:
        - R = {(1, 1),(1, 2),(1, 3),(2, 2),(2, 3)}

Relations “on a set”

When A = B then a (binary) relation on A is a relation from A to A,

hence a subset of A × A.

Example

- An example of a relation R on the set A = {a, b, c}:
    - R = {(a, b),(a, a),(c, a)} ⊆ A × A.

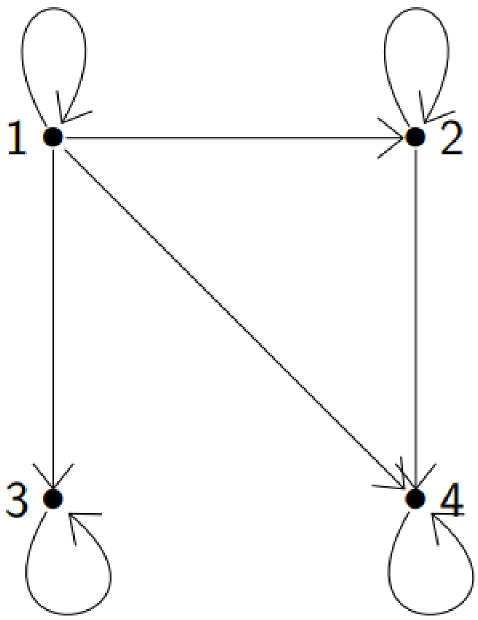
Example 2

- Let R be the relation on A = {1, 2, 3, 4} defined by
    - (x, y) ∈ R if and only if x divides y, for all x, y ∈ A.
        - Then R = {(1, 1),(1, 2),(1, 3),(1, 4),(2, 2),(2, 4),(3, 3),(4, 4)}.

Digraphs

An informative way to picture a relation on a set is to draw its

digraph.



- dots (vertices) represents the elements of A = {1, 2, 3, 4},

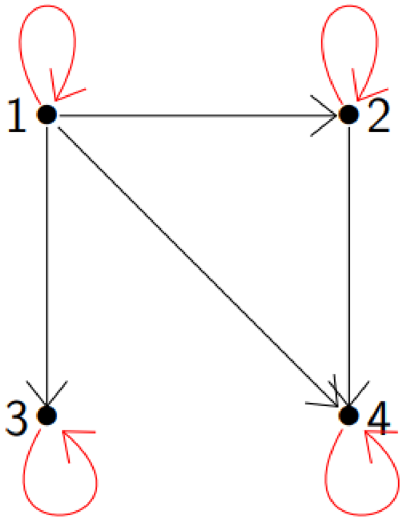
- if the element (x, y) is in the relation, an arrow (a directed edge) is drawn from x to y
- R = {(1, 1),(1, 2),(1, 3),(1, 4),(2, 2),(2, 4),(3, 3),(4, 4)}

Basic properties of relations

Reflexivity

Let R be a binary relation on a set A.

- R is reflexive if and only if (x, x) ∈ R for all x ∈ A.

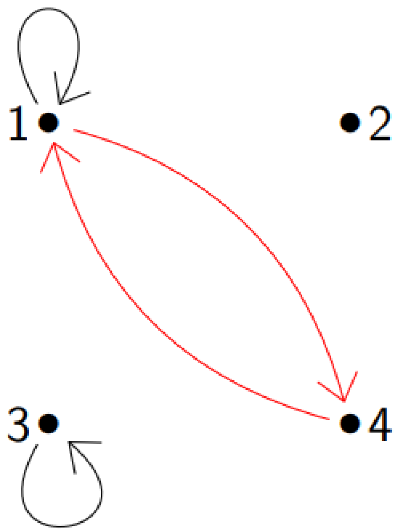


- The relation
  - R = {(1, 1),(1, 2),(1, 3),(1, 4),(2, 2), (2, 4),(3, 3),(4, 4)}
- on the set A = {1, 2, 3, 4} is reflexive
  - (1, 1),(2, 2),(3, 3),(4, 4) ∈ R

Symmetry

Let R be a binary relation on a set A.

- R is symmetric if and only if for all x, y ∈ A if (x, y) ∈ R then (y, x) ∈ R

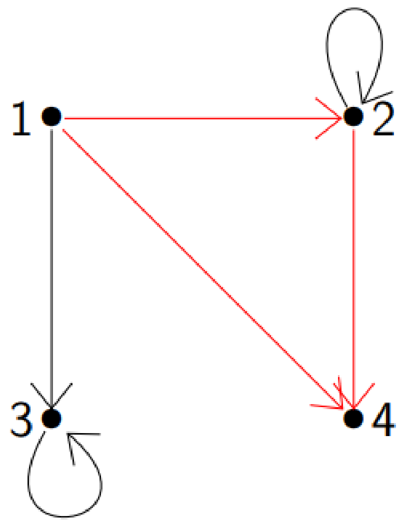


- The Relation
    - R = {(1, 1),(1, 4),(4, 1),(3, 3)}
- on the set A = {1, 2, 3, 4} is symmetric
    - (1, 4) ∈ R and also (4, 1) ∈ R

Transitivity

Let R be a binary relation on a set A.

- R is transitive if and only if for all x, y, z ∈ A if (x, y) ∈ R and (y, z) ∈ R then (x, z) ∈ R



- The relation
    - R = {(2, 2),(1, 2),(1, 3),(2, 4),(3, 3),(1, 4)}
    - on the set A = {1, 2, 3, 4} is transitive
        - (1, 2) ∈ R and (2, 4) ∈ R and also (1, 4) ∈ R

Equivalence

Let R be a binary relation on a set A

- R is an equivalence relation if and only if R is reflexive, symmetric, and transitive

Example

- The relation R on A = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12} defined by
    - R = {(x, y) | x, y ∈ A and x, y have the same remainder when divided by 3}
    - is an equivalence relation. Yes! Is it reflexive, symmetric, and transitive

Equivalence class

Suppose A is a set and R is an equivalence relation on A. For each element a in A, the equivalence class of a, [a], is the set of all element x in A such that x is related to a by R

- [a] = {x | x ∈ A and (x, a) ∈ R}
    - R is a symmetric relation, so we can also write (a, x) ∈ R

Example

- Let A = {0, 1, 2, 3} and define a binary relation R on A as follows: R = {(0, 0),(1, 1),(1, 3),(2, 2),(3, 3),(3, 1))}. R is an equivalence and an example of the equivalence class is:
    - = {x | x ∈ A and (x, 1) ∈ R} = {1, 3}

Example 2

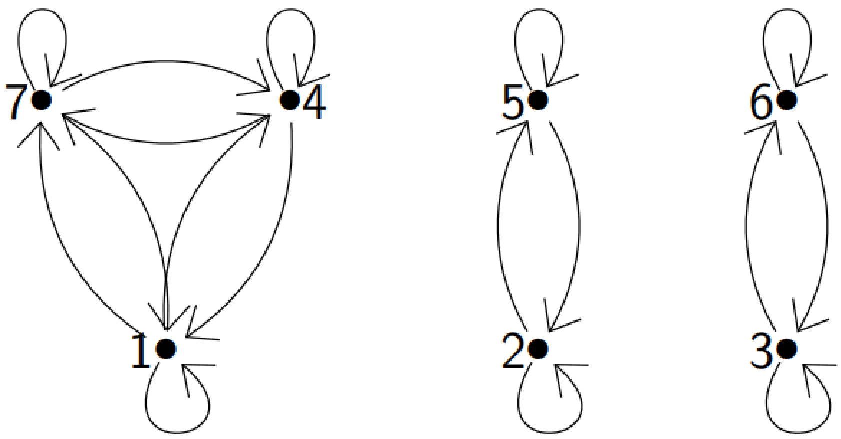
The relation R on A = {1, 2, 3, 4, 5, 6, 7} defined by:

R = {(x, y) | x, y ∈ A and x, y have the same remainder when divided by 3}

1. [1]= {x | x ∈ A and (x, 1) ∈ R} = {1, 4, 7}
2. [2] = {x | x ∈ A and (x, 2) ∈ R} = {2, 5}
3. [3] = {x | x ∈ A and (x, 3) ∈ R} = {3, 6}
4. [4] = {x | x ∈ A and (x, 4) ∈ R} = {1, 4, 7}
5. [5] = {x | x ∈ A and (x, 5) ∈ R} = {2, 5}
6. [6] = {x | x ∈ A and (x, 6) ∈ R} = {3, 6}
7. [7] = {x | x ∈ A and (x, 7) ∈ R} = {1, 4, 7}

Example 3

1. [1] = {x | x ∈ A and (x, 1) ∈ R} = {1, 4, 7}
2. [2] = {x | x ∈ A and (x, 2) ∈ R} = {2, 5}
3. [3] = {x | x ∈ A and (x, 3) ∈ R} = {3, 6}

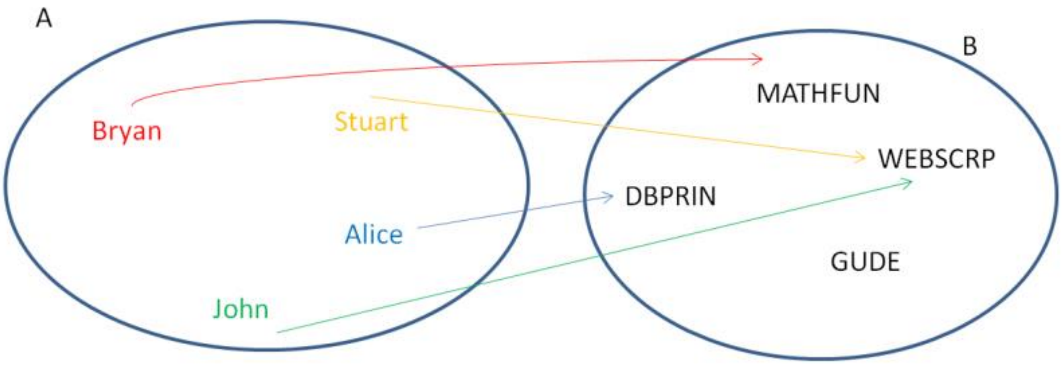


Functions

Definition

- Let A and B be nonempty sets.
    - A (total) function f from A to B, f : A → B, is a relation from A to B such that
        - for all x ∈ A there is exactly one element in B, f (x), associated with x by a relation f.

Example



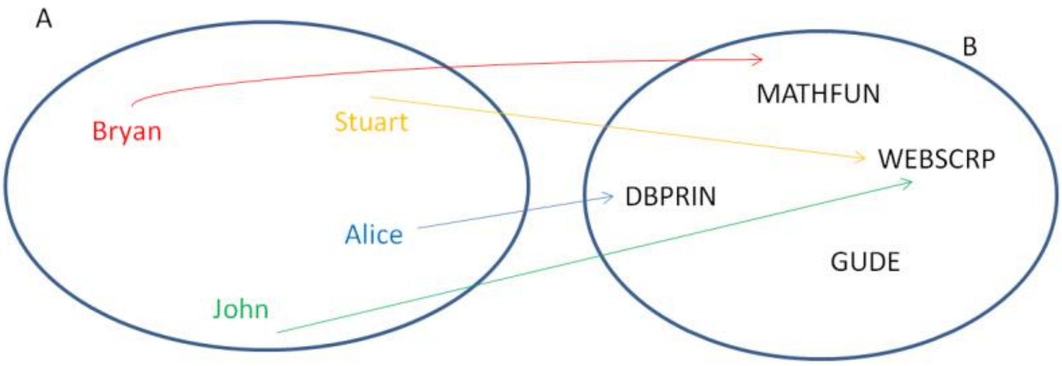
- each element of A is associated with exactly one element of B
- Such an association is called a function from A to B
- If x ∈ A is associated with y ∈ B, then x is not associated with any other element of B.

Total functions

- A (total) function f maps a set of inputs (the set A) to the outputs (the set B):
  - x ∈ A maps to y = f (x) ∈ B
  - (or might be undefined for some x ∈ A in case of a partial function)

Describing Functions

by drawing a figure:

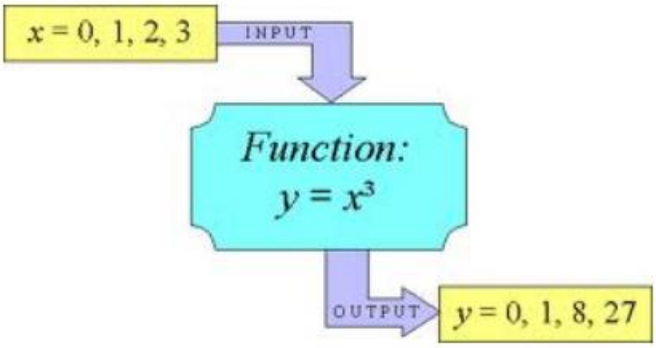


If f : A → B and f associates the element x ∈ A with the element y ∈ B then we write f (x) = y, "f maps x to y".

The expression f (x) is read "f of x" or "f at x" or "f applied to x" and is also called the image of x

by a formula:

- The function f from N → N that maps every natural number x to its cube x 3 can be described by
  - f (x) = x ^3

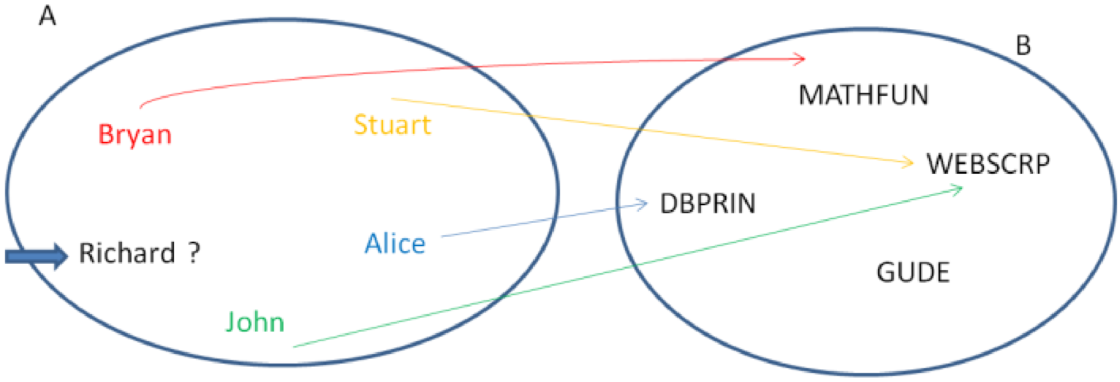


by all possible associations:

- The function g from A = {a, b, c} to B = {1, 2, 3}:
  - g(a) = 1, g(b) = 1, and g(c) = 2

Partial functions

A partial function from A to B is like a function except that it might not be defined for some elements of A.



Example

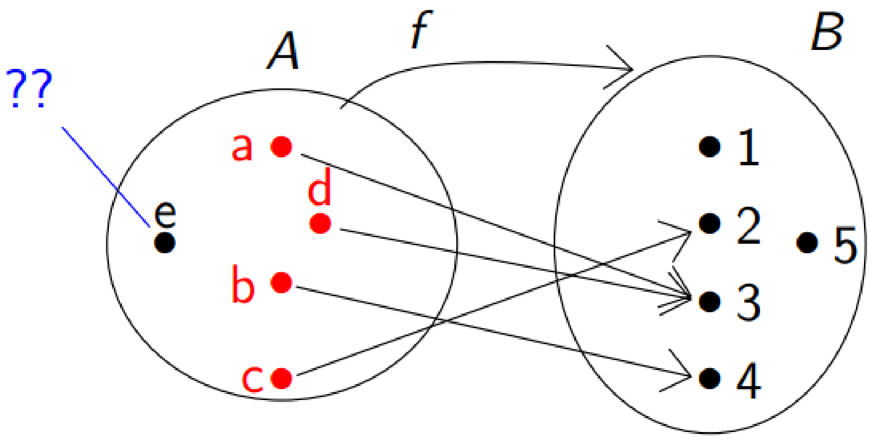
- A function f : Z → Q defined by f (x) = 1 x is an example of a partial function.
  - It is not defined for x = 0.

Domain

Let f : A → B (f is partial or total):

- The subset D ⊆ A of all elements for which f is defined is called the domain of f . In case of a total function D = A. In case of a partial function, D ⊂ A.

Example



Domain of f is

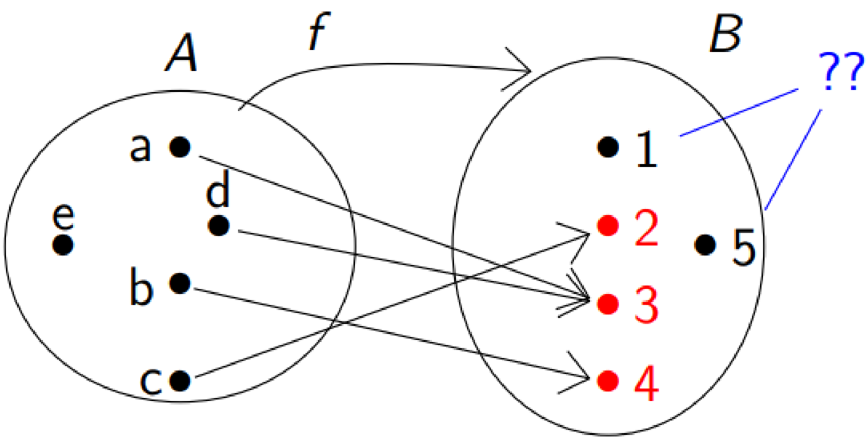
D = {a, b, c, d}

Co-domain and range (image)

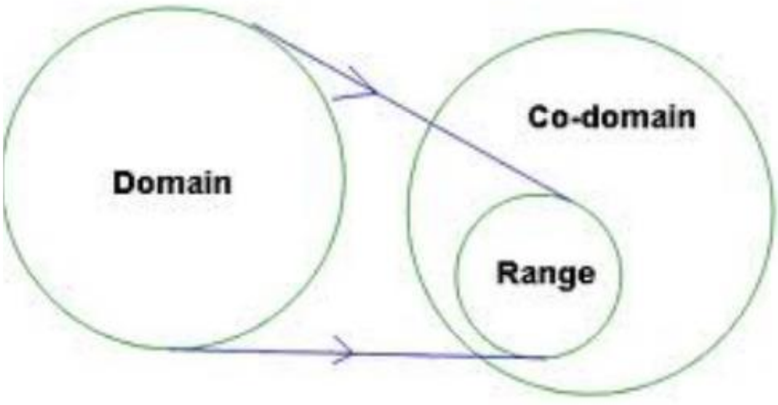
Let f : A → B (f is partial or total):

- The set B is the co-domain of f .
- The range (image) of f , denoted by range(f), is the set of elements in the co-domain B that are associated with some element of A:
    - range(f) = {f (x) | x ∈ A}.

Example



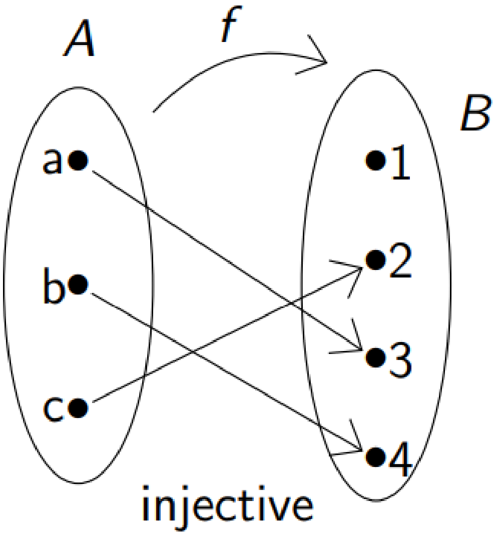
- Co-domain of f is B
    - Range of f is {2, 3, 4}

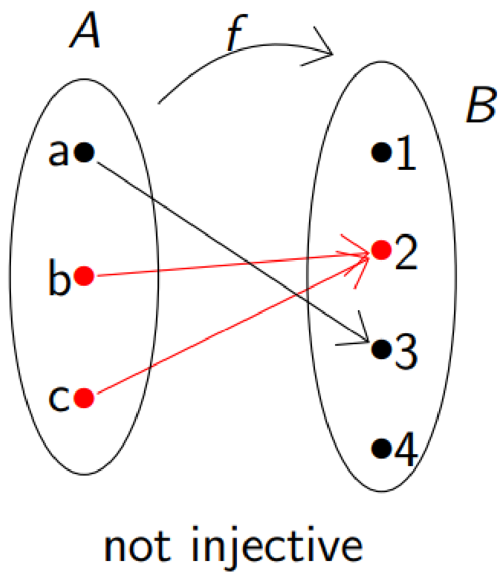


Properties of functions

Injective

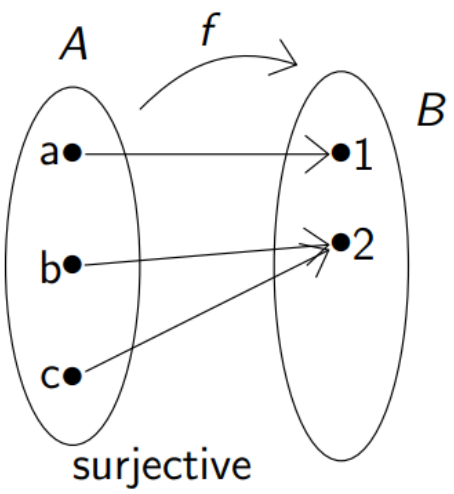
- A function f : A → B is called injective (also one-to-one) if it maps distinct elements of A to distinct elements of B

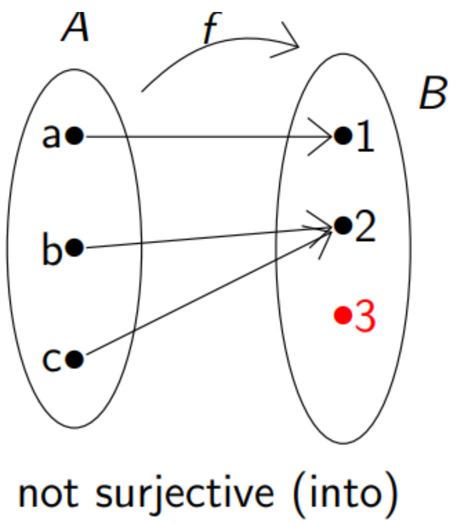




Surjective

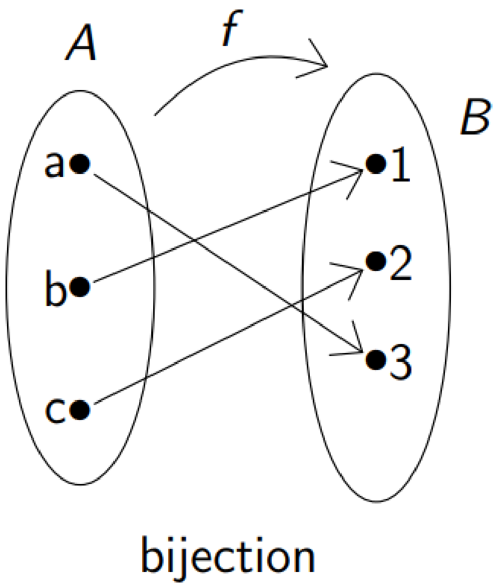
- A function f : A → B is called surjective (also onto) if the range(f) is the co-domain B.
  - for all y ∈ B there exists x ∈ A such that f (x) = y

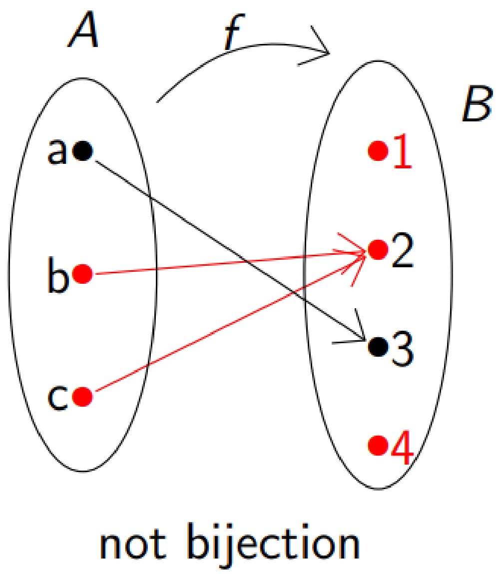




Bijective

A function f : A → B is called bijective (or one-to-one correspondence) if it is both injective and surjective.





Composite functions

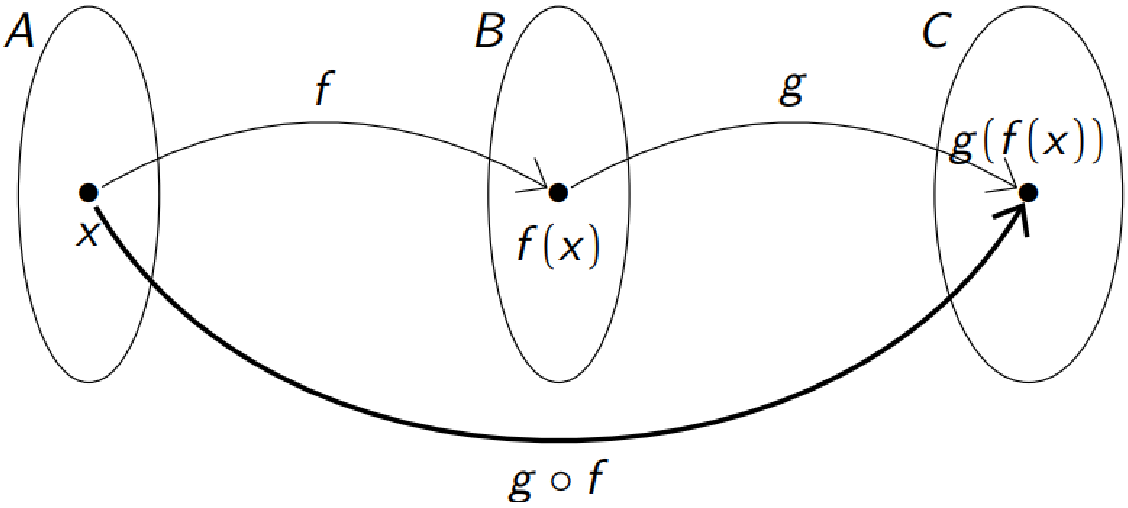
A new function can be constructed by combining other simpler functions in some way.

- Asc(First(00mat00)) = 109
  - because First(00mat00) = 0m0 and Asc(0m0) = 109
    - Asc ◦ First : S → {0, 1, . . . , 127}

Defintiton

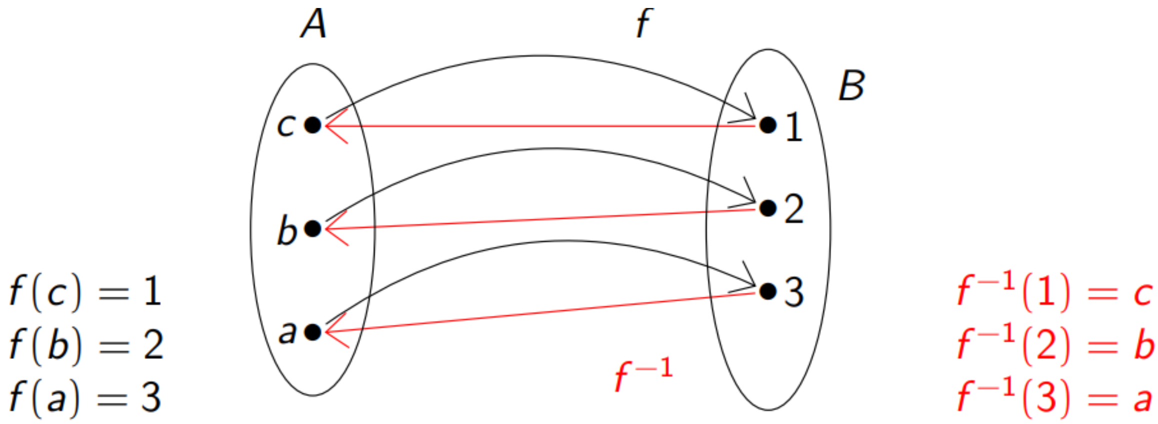
- Let f : A → B and g : B → C be functions
  - The composition of g with f is the function denoted by g ◦ f : A → C
    - defined by (g ◦ f)(x) = g(f (x)) for all x ∈ A.

(g ◦ f)(x) = g(f (x)) read as g of f , this means do f first then g.



Inverse function

- Suppose f : X → Y is a bijective function (injective and surjective).
- Then there is an inverse function f −1 : Y → X that is defined as follows:
    - f −1 (y) = x if and only if f (x) = y



Operators

- A function from A × A · · · × A to A is called an operator on A
- This means, an operator associates with each ordered pair of elements from A × · · · × A one element in A.

Example

- f (x, y) = x × y
    - where x, y ∈ N, then f is a binary operator on N
- The function Rest : S → S is an example of a unary operator.
- The number of copies of A involved in the domain of an operator is called the arity.
- Operators with arity 1 are called unary; operators with arity 2 are called binary

Logic

Propositions

Definition

A proposition is a statement (declarative sentence) that is either true or false, but not both.

Examples

- The earth is round. YES
- 2 + 3 = 7. YES
- Do you speak German? NO
- 3 − x = 5. NO
- Take two aspirins. NO
- The sun will come out tomorrow. YES

Propositional variables

In logic, the letters p, q, r, . . . denote propositional variables.

- Each propositional variable has one of two truth values: true or false.

Example

- p: Murray will win the Wimbledon next year. (True ? False)
- q: Federer is a Swiss tennis player. (True ? False)

Statements (or propositional variables) can be combined with logical connectives to obtain compound statements.

Example

- With the connectives and, or, . . . we can form composite statements using p and q:
    - Murray will the Wimbledon this year and Federer is a Swiss tennis player.

truth value of logical connectives

Example

When p is true and q is false, what is the truth value of p and q? What is the truth value of p or q?

The truth value of a compound statement depends only on:

- the truth values of the statements being combined and
- on the types of connectives being used

most important connectives:

Name	Connective	Symbol
negation	not	$\neg$
conjunction	and	$\wedge$
disjunction	or	$\vee$

Negation (not)

If p is a statement, the negation of p is the statement not p, denoted by ¬p.

p	$\neg p$
T	F
F	T

- q: It is cold.
- ¬q: It is not the case that it is cold.
  - More simply: ¬q: It is not cold.

Conjunction (and)

If p and q are statements, the conjunction of p and q is the compound statement p and q, denoted by p ∧ q.

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

Examples.

- p: 2 < 10 (True) q: 15 < 20 (True)
  - p ∧ q: “2 < 10” and “15 < 20” (True)
- p: Brighton is in France. (False) q: 2 < 3 (True)
  - p ∧ q: “Brighton is in France” and “2 < 3”. (False)

Disjunction (or)

If p and q are statements, the (inclusive) disjunction of p and q is the compound statement p or q, denoted by p ∨ q.

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

Examples.

- p: 2 < 10 (True) q: 15 < 20 (True)
  - p ∨ q: 2 < 10 or 15 < 20 (True)
- p: Brighton is in France. (False) q: 2 < 3 (True)
  - p ∨ q: "Brighton is in France" or "2 < 3". (True)

Conditional proposition (implication)

Name	Connective	Symbol
implication	if-then	$\rightarrow$ or $\Longrightarrow$

If p and q are statements, the compound statement "if p then q", denoted p → q, is called implication. p hypothesis, q conclusion

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

Example.

- p: It is raining. q: I get wet.
  - p → q: If it is raining, then I get wet.

Conditional proposition (biconditional)

Name	Connective	Symbol
biconditional	if-and-only-if	$\Leftrightarrow$ or $\leftrightarrow$

If p and q are statements, the compound statement "if and only if " (abbr. iff), denoted p ⇔ q, is called the biconditional of p and q.

p	q	$p \Leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Example

- p: 1 < 5 (True) q: 2 < 8 (True)
  - p ↔ q: 1 < 5 ⇔ 2 < 8 (True)

Truth of Compound Propositions

Truth tables for more complicated compound statements can be constructed using the truth tables we have seen so far.

Hierarchy of evaluation for the connectives (similar to algebraic expressions):

1. brackets (highest, do first)
2. ¬
3. ∧
4. ∨
5. →
6. ↔

Example

Compute the truth table of the statement p → ¬(p ∨ q) ∧ p (the

same as p → (¬(p ∨ q) ∧ p))

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

Tautology

A statement that is:

- true for all possible values of its propositional variables is called a tautology

example

For any proposition p: p ∨ ¬p is a tautology.

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

Contradiction

A statement that is:

- false for all possible values of its propositional variables is called a contradiction

Example

For any proposition p: p ∧ ¬p is a contradiction.

p	$\neg p$	$p \wedge \neg p$
T	F	F
F	T	F

Contingency

a statement that can be either true or false depending on the truth values of its propositional variables is called a contingency

Example

- p → ¬(p ∨ q) ∧ p

Logical Equivalence

Definition

Two statements are said to be logically equivalent, ≡, iff they have identical truth values for each possible value of their statement variables. (Corresponds to = with numbers)

Example

p → q ≡ ¬p ∨ q.

p	q	$p \to q$	$\neg p$	$\neg p \vee q$
T	T	T	F	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Logic of Quantified Statements

Propositional logic

Propositional logic applies to simple declarative statements where the basic propositions are either true or false.

examples of the propositions (in propositional logic):

- “Albert Einstein was a mathematical major.”
- “Mr Bean is a mathematical major.”
- But “He is a mathematical major.” is not a proposition, it may be either true or false depending on the value of he.

Predicates

Definition

A predicate (or propositional function) is a statement containing one or more variables. If values from a given set (domain) are assigned to all the variables, the resulting statement is a proposition.

In computer science we like variables and statements like

“p : x is an integer less than 80”

- The statement p is not a proposition, it is much more like a function with a variable x, hence also p(x).
- True/False value of p depends on the value of variable x,
  - e.g. if x = 103 then p(x) is false, if x = 2 then p(x) is true

Quantifiers

1. Universal quantifier:
   a. The symbol ∀ (a upside-down A) is called the universal quantifier; the meaning is for all (for each).
2. Existential quantifier:
   a. The symbol ∃ (a backwards E) is called the existential quantifier; the meaning is there exists.

Universal quantifier ∀

Definition

For a predicate p(x) with domain D the statement:

- “for every x from domain D, p(x)”
  - may be written $\forall x \in D\ p(x)$.

Example 1. “All students in this class are happy” can be rewritten:

- Let D be the set of all students in this class, then
  - $\forall x \in D$, x is happy.

Example 2. Let S = {1, 2, 3, 4, 5, 6} and consider the statement

- $\forall x \in S, x^2 > x$

Example 3. $\forall x \in R, x^2 > x$.

True statements with ∀

- The statement $\forall x \in D\ p(x)$ is true if p(x) is true for every $x \in D$.

False statements with ∀

- The statement $\forall x \in D\ p(x)$ is false if p(x) is false for at least one $x \in D$.

Existential quantifier ∃

Definition

- For a predicate p(x) with domain D the statement
  - “there exists an x from the domain D such that p(x)”
    - may be written $\exists x \in D, p(x)$.

Example 1.

- "There is a happy student in this class"
  - can be rewritten:
  - Let D be the set of all students in this class, then
    - $\exists x \in D$, x is happy.

Example 2.

- Let S = {1, 2, 3, 4, 5, 6} and consider the statement
  - $\exists x \in S, x^2 > x$.

True statements with ∃

The statement $\exists x \in D\ p(x)$ is true if p(x) is true for at least one $x \in D$.

False statements with ∃

The statement $\exists x \in D\ p(x)$ is false if p(x) is false for all $x \in D$.

Methods of proof

What is a proof?

Example.

Is it true that for all integers m and n, if m is odd and n is even, then m + n is odd?

- It is true for m = 7, n = 10 and also for m = 9, n = 2, . . . But is it really true for all integers m and n with the given properties?
- To be absolutely sure, we need to prove it!
- A mathematical proof is a carefully reasoned argument to convince a sceptical listener that a given statement is true

Theorem

Prove that for all integers m and n, if m is odd and n is even, then m + n is odd.

An argument (theorem) is a finite collection of statements $p_1, p_2, \ldots, p_n$ called premises (or hypotheses) followed by a statement q called the conclusion.

$(p_1 \land p_2 \land \cdots \land p_n) \Longrightarrow q$

- Premises: "m, n, integers, m is odd, n is even"
- Conclusion: "m + n is odd"

There are several different proof techniques: a direct proof, a proof by contradiction, a proof by contrapositive, a proof by mathematical induction, . . .

The choice of proof technique depends on the problem and experience.

A direct proof

Theorems are often of the form:

p (hypothesis) $\Longrightarrow$ q (conclusion)

- In a direct proof we start with the hypothesis of a statement (premises) and make one deduction after another until we reach the conclusion.

Suppose:

p is true

⋮

"proof"

⋮

Our task: prove that then q is also true

In the proof we can use:
– previously proven facts,
– definitions,
– known basic properties, . . .

Theorem

For all integers m and n, if m is odd and n is even, then m + n is odd

In a direct proof, we assume the hypotheses are true and derive the conclusion!

Suppose: (hypotheses) m, n integers

m is odd, n is even

???

Want to prove: $m + n$ is odd (conclusion)

Proof.

An integer r is even if and only if there exists an integer k such that r = 2k

Similarly, an integer r is odd if and only if there exists an integer k such that r = 2k + 1.

In our theorem:

- m is odd ==⇒ there exists an integer k such that m = 2k + 1
- n is even ==⇒ there exists an integer l such that n = 2l

The sum is:

- m + n = (2k + 1) + 2l = 2(k + l) + 1

hence m + n is odd.

Proof by Contradiction (an indirect proof)

- There are only two options for the truth value of a conclusion: true or false.

- If supposing that the premises are true and the conclusion is false we are able to arrive at a contradiction (a conclusion that is contradictory to our assumptions or something obviously untrue like 1 = 0) ==⇒ our conclusion must be true!

Suppose:

p is true
$\neg q$ is true
⋮
"proof"

In the proof we can use:
– previously proven facts,
– definitions,
– known basic properties, . . .

Our task: prove a contradiction
Then necessarily q must be true (assuming p is true)!

Theorem

For every n ∈ N, if n^2 is even, then n is even.

N^2 is even, n ∈ N ==⇒ n is even

Suppose: n^2 is even, n natural number (hypothesis)
n is not even, hence n is odd (the conclusion is false)
???

Want to prove: any contradiction ($r \wedge \neg r$ for a proposition r)

Proof.

- Since n is odd, there exists k ∈ N such that n = 2k + 1.
- Now n^2 = (2k + 1) 2 = 4k 2 + 4k + 1 = 2(2k 2 + 2k) + 1.
- Thus n^2 is odd
- We found a contradiction ==⇒ the conclusion must be true!

Proof by Contrapositive (an indirect proof)

- The contrapositive of the condition proposition p → q is the proposition ¬q → ¬p.
- The conditional proposition p → q and its contrapositive ¬q → ¬p are logically equivalent

To prove a statement by contrapositive, we prove the contrapositive statement by a direct proof and conclude that the original statement is true.

This means, instead of the original theorem

p → q

we prove by a direct proof the contrapositive theorem

¬q → ¬p.

Suppose:

$\neg q$ is true

⋮

"proof"

⋮

In the proof we can use:
– previously proven facts,
– definitions,
– known basic properties, . . .

Our task: prove that $\neg p$ is true

In this way we prove ¬q → ¬p and because of ¬q → ¬p ≡ p → q necessarily p → q must be true as well (the theorem p → q is valid)

Theorem

For every n ∈ N, if n 2 is even, then n is even.

A contrapositive statement: For every n ∈ N, if n is not even, then n 2 is not even.

Contrapositive: For every n ∈ N, if n is odd, then n 2 is odd.

Now prove the contrapositive statement using a direct proof.

Suppose: n is odd, n natural number (hypotheses)

???

Want to prove: n^2 is odd

Proof.

- Since n is odd, there exists k ∈ N such that n = 2k + 1.
- Now n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1
- Thus n^2 is odd.

Hence the contrapositive statement is true and by the logical equivalence also the theorem

"For every n ∈ N, if n 2 is even, then n is even."

is valid.

Introduction to Mathematical Induction

- Mathematical induction is one of the most basic methods of proof.
- It is very useful to establish the truth of a statement about all natural numbers.

Example. What is the sum of the first n natural numbers?

Let's look at this problem for n = 1, 2, and 3 and calculate the sum:

1 = 1

1 + 2 = 3

1 + 2 + 3 = 6

Is there any formula to calculate this sum? Yes!

Mathematical Induction is the most suitable tool for proving such statements.

Principle of Mathematical Induction

Let P be a predicate that is defined for integers n. Suppose

Basis step P(a) is true for some integer a;

Inductive step For all integers k > a, if P(k) is true, then P(k + 1)

is true.

Then P(n) is true for all integers n > a

Introduction to graphs

Basic terminology

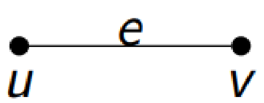
Definition

A graph G is a pair (V, E) of sets:

- V is a nonempty set of vertices (nodes),
- E is a set of edges, each element of E is a set of two distinct elements of V.

– if e ∈ E, then e = {u, v}, u and v are different elements of V called the end vertices of e

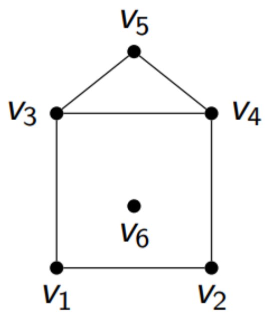
– e joins (connects) vertices u and v



- we can use uv instead of e = {u, v} for the edge e (this means uv and vu is the same edge)
- the vertices u and v are said to be incident with the edge uv or they are adjacent because they are the end vertices of an edge

Example

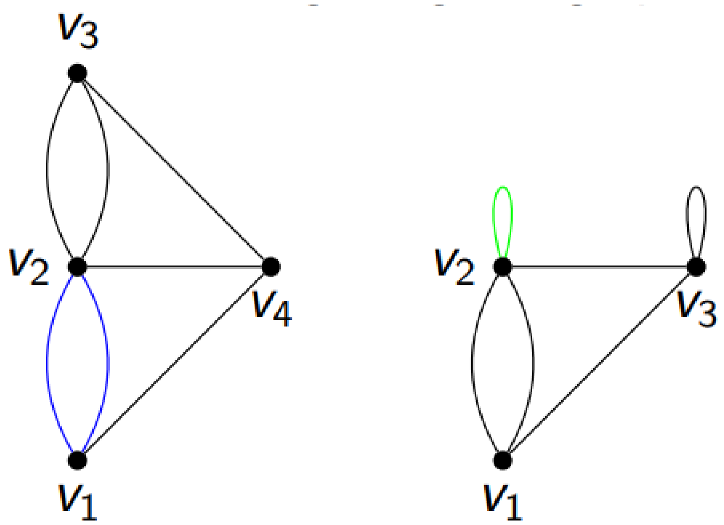
Usually we draw a picture of a graph, rather than presenting it formally as sets of vertices and degrees.



$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$
$$E = \{\{v_1, v_2\}, \{v_2, v_4\}, \{v_1, v_3\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_4, v_5\}\}$$
or $E = \{v_1 v_2, v_2 v_4, v_1 v_3, v_3 v_4, v_3 v_5, v_4 v_5\}$

Multigraphs

How about the following two “graphs” (the first one is the graph from the Konigsberg bridge problem)



They are not graphs according to our definition! Why? They have multiples edges and loops.

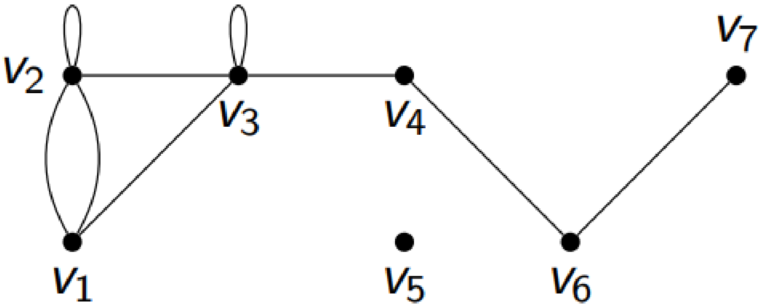
A multigraph/pseudograph is like a graph, but it may contain loops and/or multiple edges. No formal definition of multigraphs will be presented, but we can study some of their properties as well.

Degree of a vertex

Definition

The number of edges incident with a vertex v is called the degree of v and is denoted deg v. A vertex of degree 0 is said to be isolated.

Example



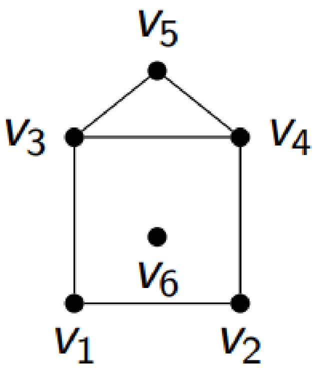
deg v4 = 2, deg v6 = 2, deg v7 = 1, deg v5 = 0 (v5 is an isolated vertex)

deg v1 = 3, deg v2 = 5, deg v3 = 5

Degree sequence

Definition

When d1, d2, . . . , dn are the degrees of the vertices of a graph (or multigraph) G ordered so that d1 6 d2 6 · · · 6 dn. Then (d1, d2, . . . , dn) is called the degree sequence of G.

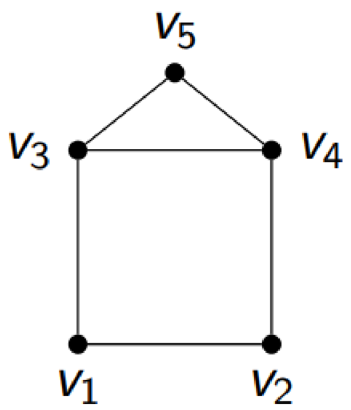


The degree sequence: (0, 2, 2, 2, 3, 3)

Euler Theorem

Theorem (Euler Theorem or Handshaking lemma)

In any graph $G = (V, E)$ the sum of all the vertex-degrees is equal to twice the number of edges, $\sum_{v \in V} \deg v = 2|E|$.



The sum of all vertex-degrees:

2 + 2 + 2 + 3 + 3 = 12

The number of edges: 6

2 × 6 = 12

Consequences of Euler Theorem

The Euler Theorem leads to several more consequences:

- In any graph, the sum of all the vertex-degrees is an even number.

Example. Is there a graph with the degree sequence: (1, 2, 3, 3)? No! Why?

3 + 3 + 2 + 1 is odd number!

- In any graph, the number of vertices of odd degree is even.

Example. Is there a graph with the degree sequence: (1, 2, 2, 2)? No! Why?

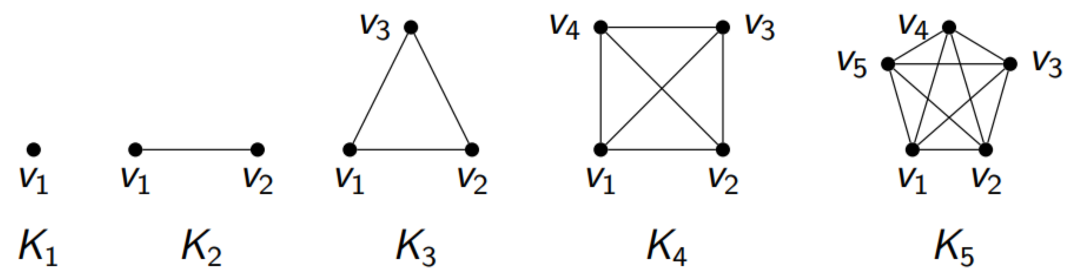
E.g. There is only one vertex of odd degree.

Special types of graphs

complete graphs

Definition

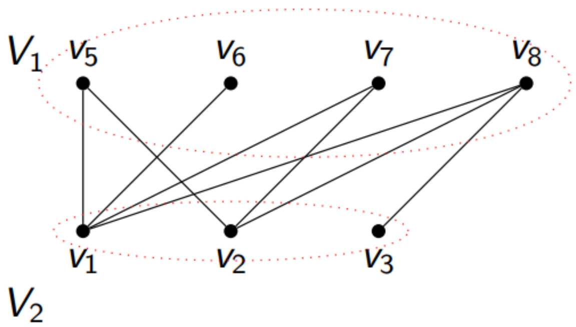
For any positive integer n, the complete graph on n vertices, denoted Kn, is that graph with n vertices every two of which are adjacent.

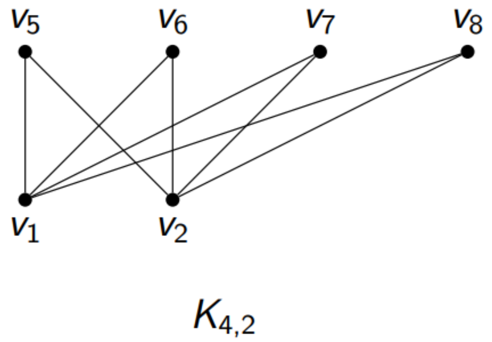


bipartite graphs

Definition

A bipartite graph is one whose vertices can be partitioned into (disjoint) sets V1 and V2 in such a way that every edge joins a vertex in V1 with a vertex in V2 (no edges within V1 nor within V2).



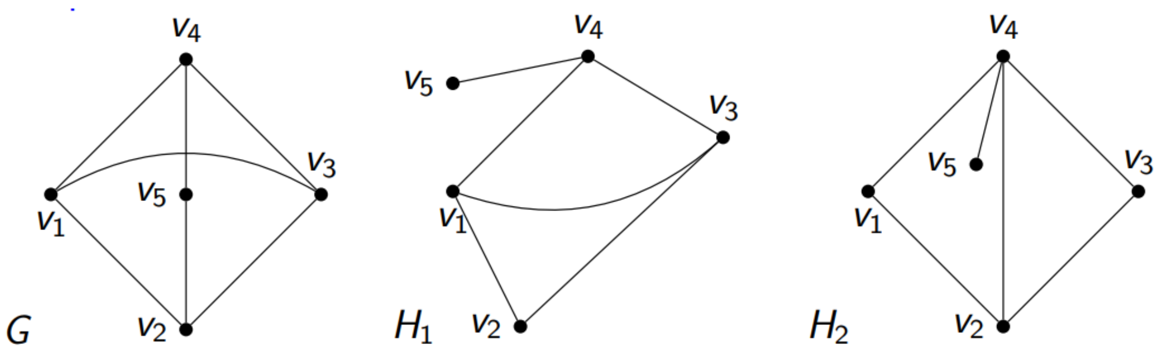


Subgraphs

Definition

A graph H is a subgraph of a graph G iff the vertex and edge sets of H are, respectively, subsets of the vertex and edge sets of G.

Example

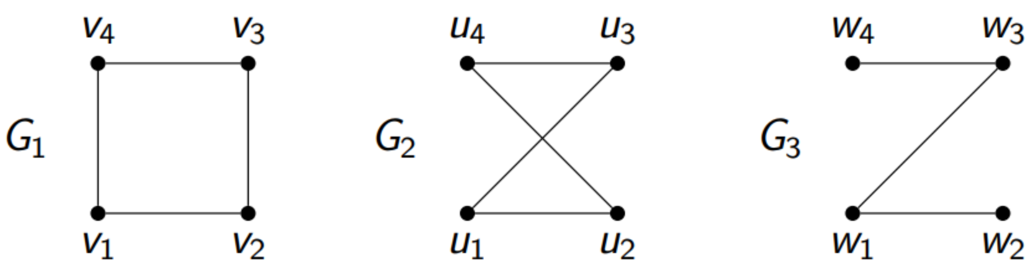


How the graph is drawn is not important! H1 is a subgraph of G, H2 is not a subgraph of G

Isomorphic graphs

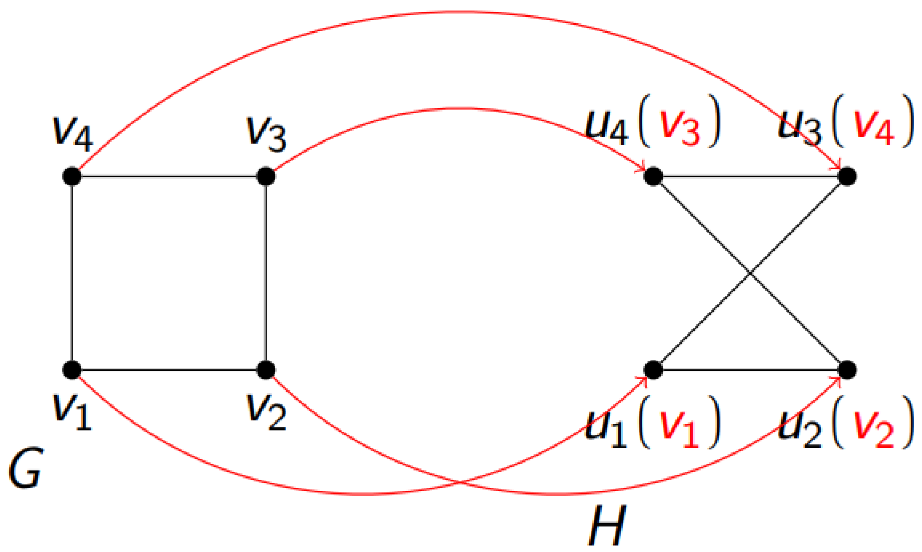
It is important to know when two graphs are essentially the same and when they are essentially different. How the graph is drawn is not important, the vertices and the edges are important!

Example



The number of edges in G3 is smaller than the number of edges in G1, but G1 and G2 are "the same", G1 is isomorphic to G2.

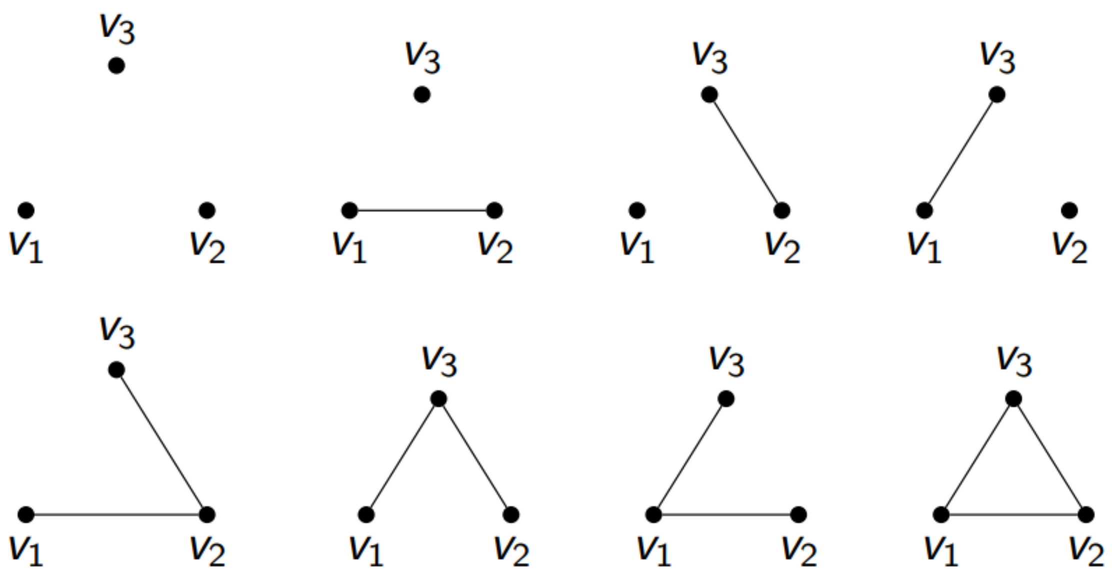
Two graphs G and H are isomorphic if H can be obtained from G be by re-labelling the vertices.



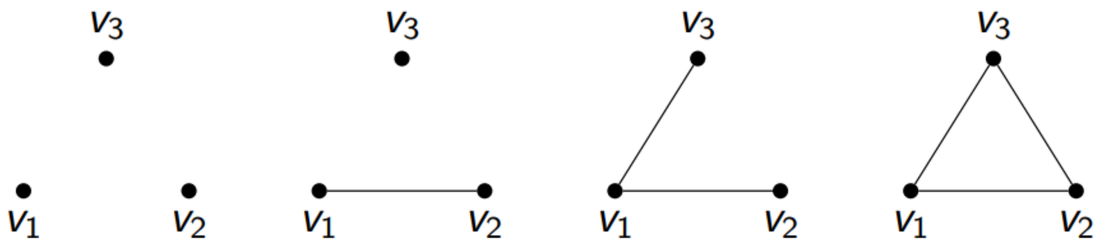
G is isomorphic to H if there is a bijective function (injective and surjective) f : V(G) → V(H) such that

- if u and v are adjacent in G, then f (u) and f (v) are adjacent in H;
- if u and v are not adjacent in G, then f (u) and f (v) are not adjacent in H;

Draw all possible graphs on the vertex set v1, v2, v3 (labelled graphs):



Draw all non-isomorphic graphs on the vertex set v1, v2, v3 (unlabelled graphs):



- Generally, it is difficult to prove that two graphs are isomorphic: we have to try all the bijections between vertex sets and check.
- It can be shown, that if G and H are isomorphic graphs, then G and H:
- 1 have the same number of vertices,
- 2 have the same number of edges,
- 3 have the same degree sequence,
- 4 either both are connected or both are not connected

So sometimes it is easier to show that graphs are not isomorphic, e.g. it is enough when one of the properties above is broken.

Key Terms

- graph, vertex, edge
- multigraph, multiple edges, loops
- incident, adjacent
- degree, degree sequence, isolated vertex
- complete graph
- bipartite graph, complete bipartite graph
- isomorphic graphs, isomorphism of graphs
- Euler Theorem (Handshaking lemma)

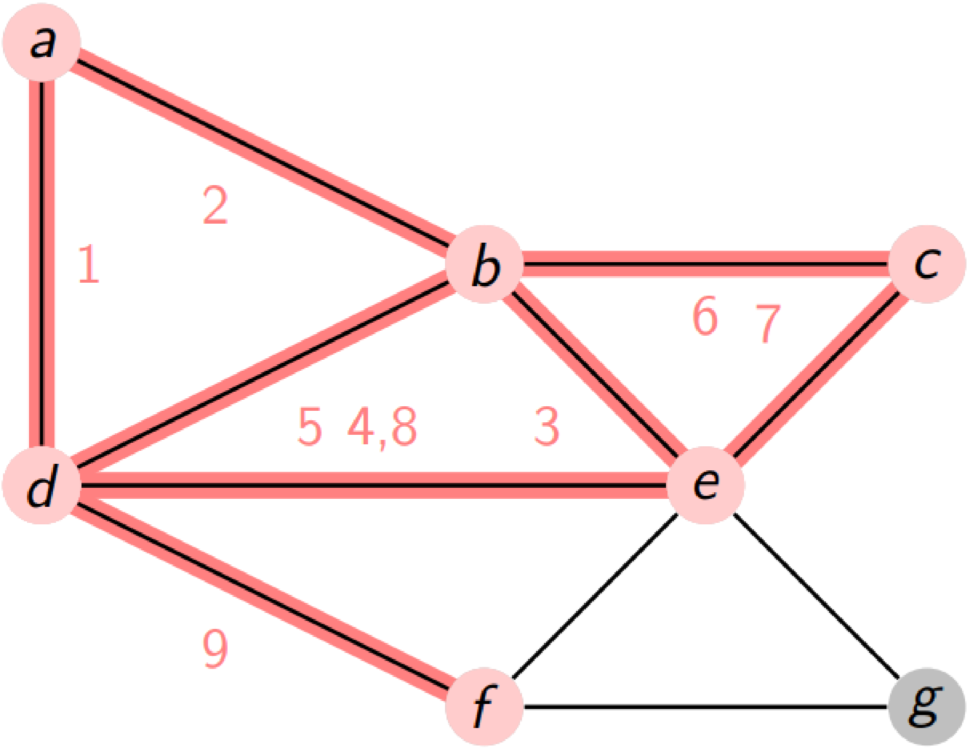
Walks, trails, paths

Walks

- many real problems, when translated to graph theory enquire about the possibility of walking through a graph
- most of the definitions and results about walks are also valid for multigraphs (even when we don't mention it)

Definition

A walk in a multigraph is an alternating sequence of vertices and edges (beginning and ending with a vertex), each edge is incident with the vertex immediately preceding and following it. The length of a walk is the number of edges in it.



A walk of length 9:

d–da–a–ab–b–be–

e–ed–d–db–b–bc–

c –ce –e –ed –d –df

our notation:

(d, a, b, e, d, b, c, e, d, f)

a walk can go through the same vertex/use the same edge several times

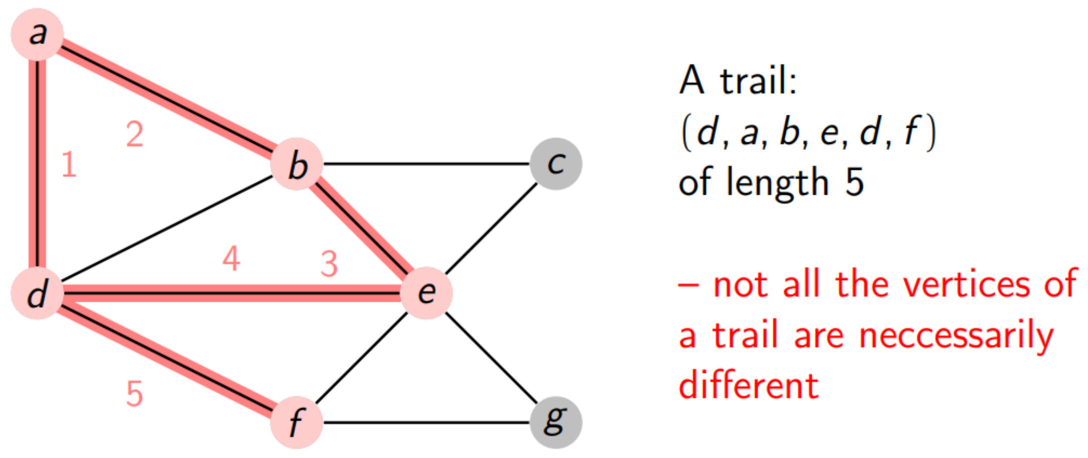
A walk is closed if the first vertex is the same as the last, e.g. the walk (d, a, b, e, d, b, c, e, d), and is otherwise set to be an open.

Trails and paths

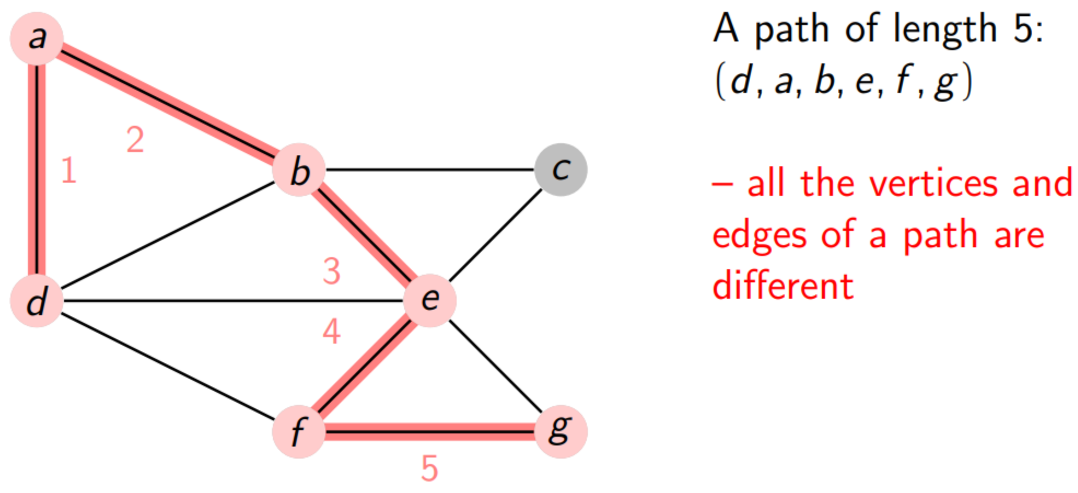
Definition

- A trail is a walk in which all edges are distinct;
- a path is a walk in which all vertices are distinct.

Example (trail).



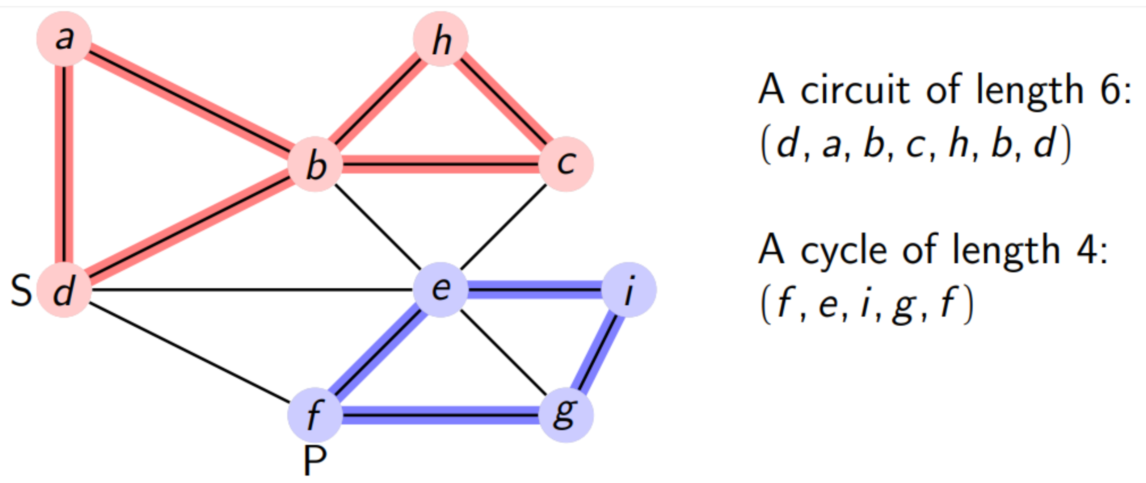
Example (path).



<u>Circuits and cycles</u>

Definition

- A closed walk in which all edges are different is called a circuit (a closed trail).
- A closed walk in which all vertices (except the first and the last vertex) are different is called a cycle (a closed path).



Overview – various types of walks

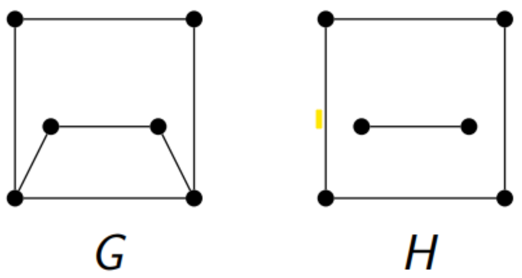
Term	Vertices	Edges	Start	Stop
Walk	may repeat	may repeat	anywhere	anywhere
Trail	may repeat	not repeat	anywhere	anywhere
Path	not repeat	not repeat	anywhere	anywhere
Circuit	may repeat	not repeat	same vertex	same vertex
Cycle	not repeat	not repeat	same vertex	same vertex

Connected Graphs

Definition

A graph G is connected if there is a path in G between any pair of vertices and disconnected otherwise.

Example.

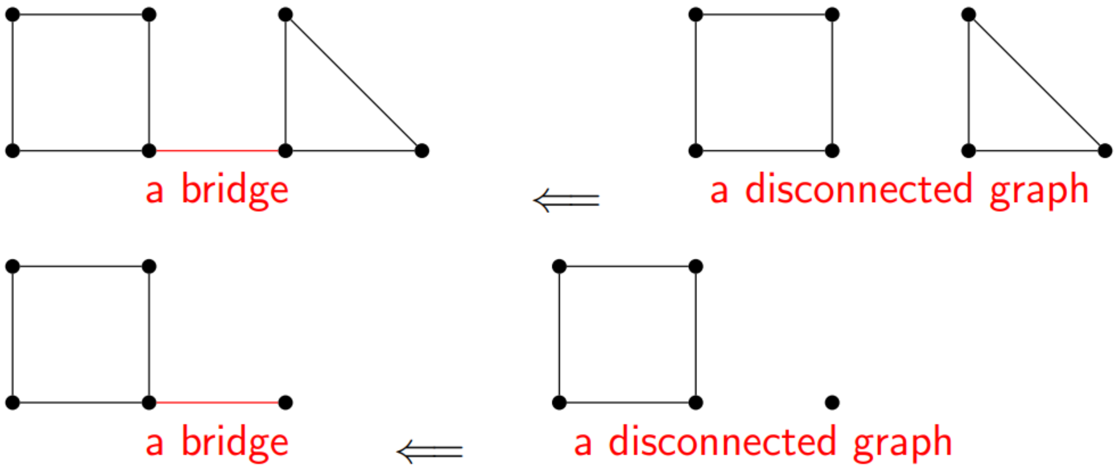


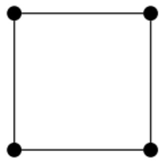
G is a connected graph (w
H is a disconnected graph

Definition

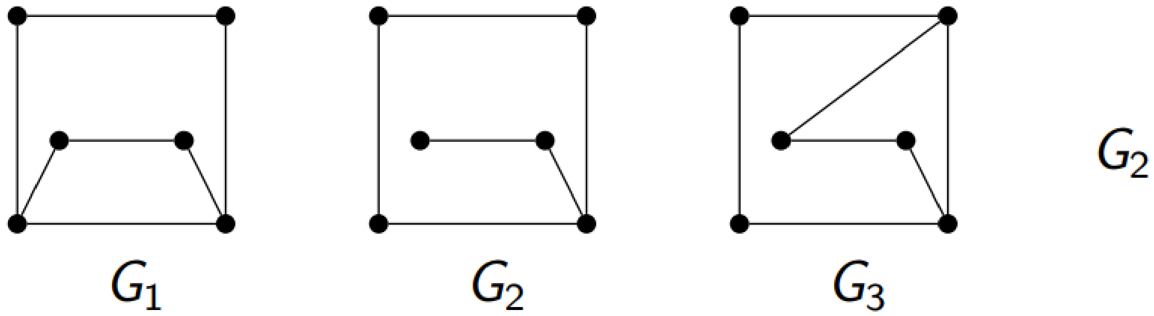
An edge in a connected graph is a bridge if deleting it would create a disconnected graph.

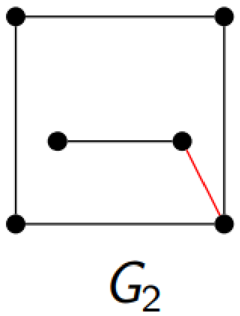
Examples



A graph without a bridge: 

Which of the following graphs has a bridge?

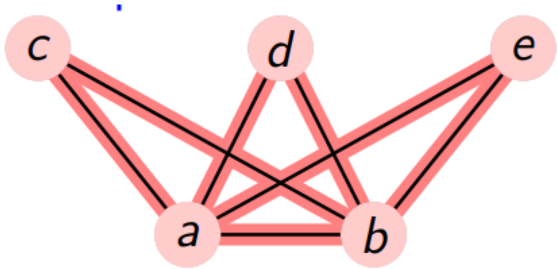




Eulerian graphs

A graph is Eulerian iff it has a circuit that contains every edge – a closed walk using each edge exactly once (called an Eulerian circuit)

Example

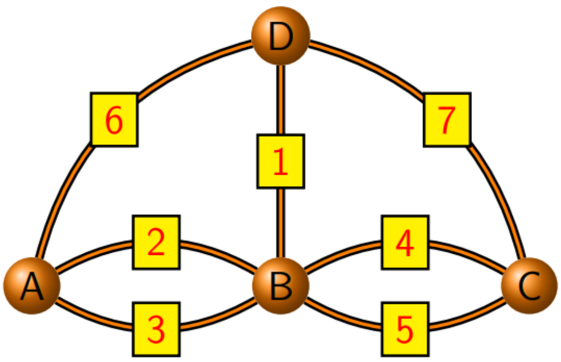


Eulerian circuit:
(a, c, b, d, a, e, b, a)

Characterisation of Eulerian graphs

Theorem

A multigraph is Eulerian if and only if it is connected and every vertex has even degree.



the graph has vertices of odd degree $\Longrightarrow$ the graph is not Eulerian $\Longrightarrow$ a closed walk containing each edge exactly once does not exist :-(Euler has to destroy one bridge!

Fleury's Algorithm

Suppose there is an Eulerian graph on the input.

Step 0: Choose any vertex to start.

Step 1: From that vertex choose an edge to traverse, choosing a bridge only if there is no alternative.

Step 2: After traversing that edge, erase it (and vertices of degree 0), coming to the next vertex.

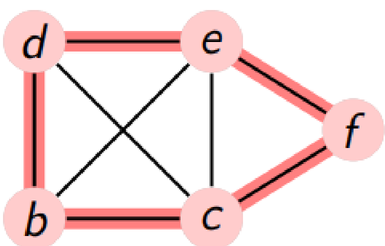
Step 3: Repeat steps 1-2 until all edges have been traversed, and you should be back at the starting vertex

Example

Travelling Salesman and Hamiltonian graphs

A graph is Hamiltonian iff it has a cycle that contains every vertex – a closed path using each vertex exactly once (called a Hamiltonian cycle).

Example



Hamiltonian cycle:
(d, e, f, c, b, d)

How can we construct a Hamiltonian cycle?

- There are known algorithms for finding a Hamiltonian cycle but at present none are known that would guarantee to find it in a “reasonable amount” time.
- The known algorithms use an exhaustive search of all possibilities – require exponential or factorial time in worst case
- Bad news. The problem of finding a Hamiltonian cycle is difficult, hence also the travelling salesman problem

Adjacency matrix

There are several possibilities of how the information about a graph can be coded when working in a program, e.g. using sets.

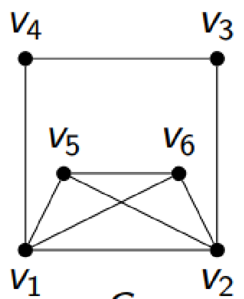
Commonly is also used the adjacency matrix.

Definition

Let G be a graph with n vertices labeled v1, v2, . . . , vn. The adjacency matrix of G is the n × n matrix A = (aij) whose (i, j) entry is aij , where for each i and j with 1 6 i, j, 6 n, define

$$a_{ij} = \begin{cases} 1 & \text{if } v_i v_j \text{ is an edge} \\ 0 & \text{if } v_i v_j \text{ is not an edge.} \end{cases}$$

Example

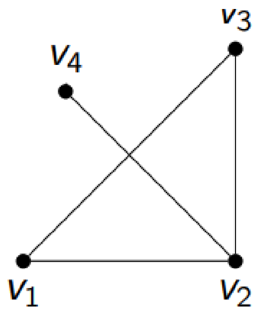


G

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	0	1	1	1
v_2	1	0	1	0	1	1
v_3	0	1	0	1	0	0
v_4	1	0	1	0	0	0
v_5	1	1	0	0	0	1
v_6	1	1	0	0	1	0

$$\text{The adjacency matrix: } A = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Let G be a graph with vertices v1, v2, . . . , vn and let A = (aij) be the adjacency matrix of G.



$$\text{The adjacency matrix: } A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

- The diagonal entries of A are all 0; that is, aii = 0 for i = 1, . . . , n.
- The adjacency matrix is symmetric, that is aij = aji for all i, j.
- deg vi is the number of 1's in row i; this is also the number of 1's in column i (row i and column i are the same)

revise how to multiply matrices

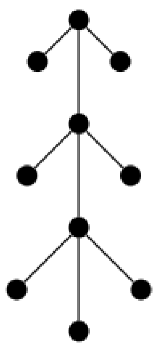
Trees

introduction to trees

For us trees form a (very nice!) subclass of graphs which is used extensively in computer science, chemistry, linguistics

Definition:

A tree is a connected graph that contains no cycles.



Alternative definitions

Let G be a graph. Then the following statements are equivalent.

Definition (Alternative definitions of tree)

- G is a tree.
- G is connected and acyclic, that is, without cycles.
- Between any two vertices of G there is precisely one path.

How to recognise a tree

Question. You are given a graph G. How can you decide whether G is a tree or not?

Theorem

A connected graph with n vertices is a tree if and only if it has n − 1 edges.

Two important pieces of information follow from this theorem:

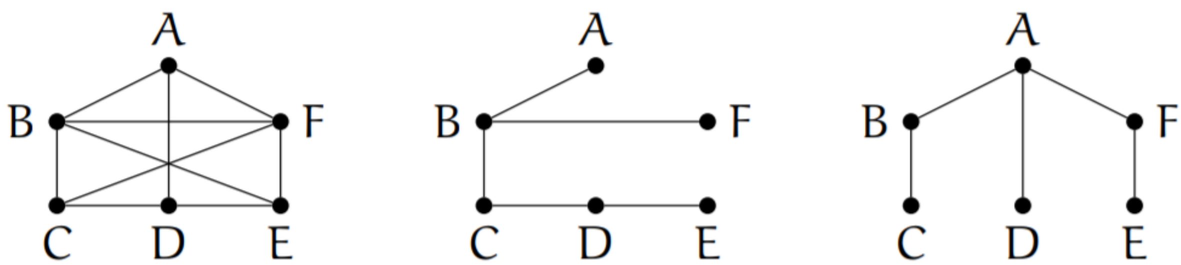
- If a connected graph with n vertices has n − 1 edges, it must be a tree!
- If a connected graph with n vertices is a tree, the graph has exactly n − 1 edges.

spanning trees

Definition

A spanning tree of a connected graph G is a subgraph that is a tree and that includes every vertex of G

Example. A graph and several of its spanning trees

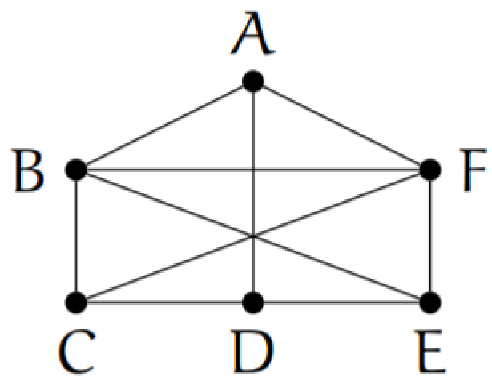


Spanning trees are considered to be different if they make use of different edges of the graph.

How to find a spanning tree?

Finding a spanning tree in a connected graph G is not hard (a connected graph on n vertices is a tree iff it has n − 1 edges)

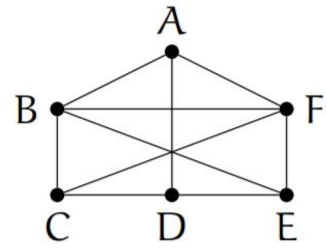
- If G has no cycles, then it is already a tree, so G itself is a spanning tree for G.



Delete an edge from a cycle (without deleting any vertices), e.g. AD

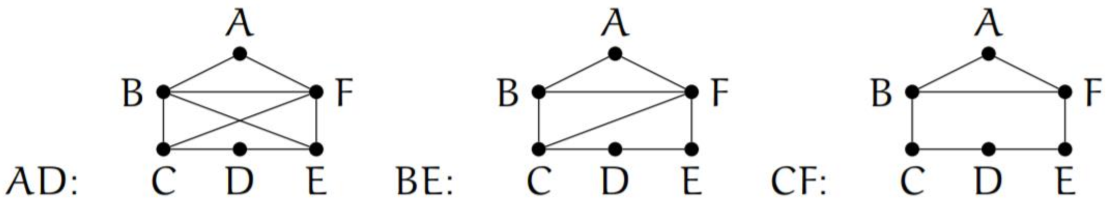
By repeating the above procedure, we eventually find a connected subgraph without cycles containing all vertices of G, that is, a spanning tree of G.

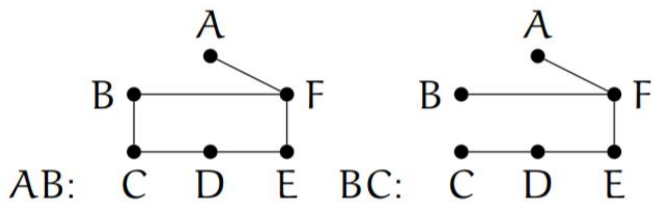
Example



Delete an edge from a cycle (without deleting any vertices), e.g. AD

. . .





. . . finally a tree!

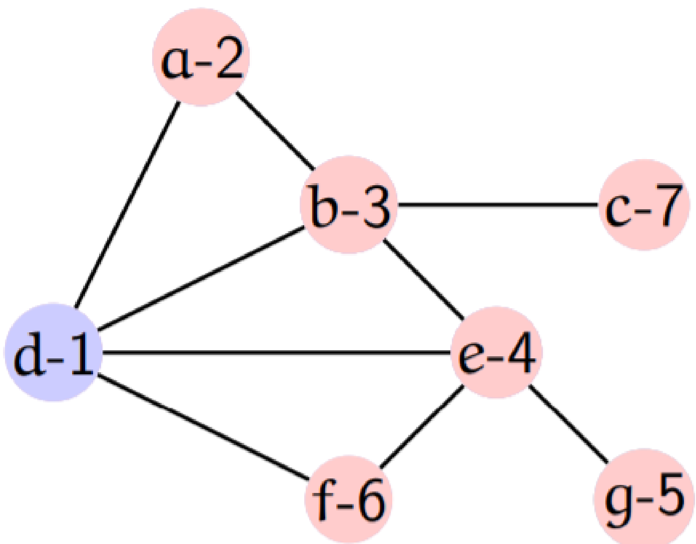
Depth-First Search

- how do we test for a cycle in a graph in the program?
- Is it really a good idea to always seek an edge which is in a cycle?

It is better to use an algorithm based on a depth-first search method.

- ''Depth-first search' can be useful for other important algorithms, e.g. to test whether a graph is connected, produce a spanning tree in the connected case.
- The method is based on the exploring of vertices in some way

Depth-First Search (idea)

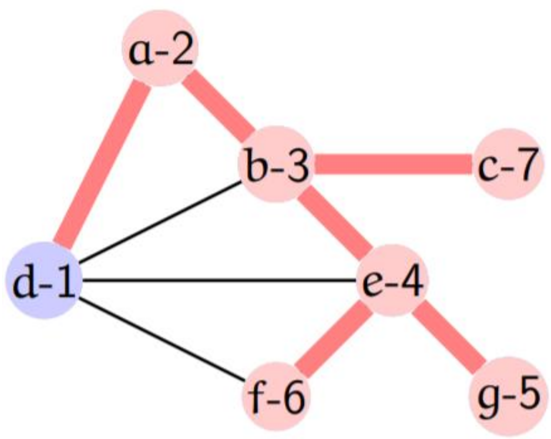


Step 1: Start at any vertex (label it), Step 2: choose any adjacent unlabelled vertex to it (label it and move to it), . . . go to Step 2

Step 3: when there is no unlabeled adjacent vertex to it, – find the last labelled vertex with an unlabelled adjacent vertex (backtrack) and . . . go to Step 2 – or finish when back at the first labelled vertex

Depth-First Search and Spanning Tree

- The depth-first search method can be used for finding a spanning tree.
- When a new vertex is labelled, we always add the "'exploring" edge.



A spanning tree $T = (V, E)$,
$V = \{a, b, c, d, e, f, g, \}$,
$E = \{da, ab, be, eg, ef, bc\}$

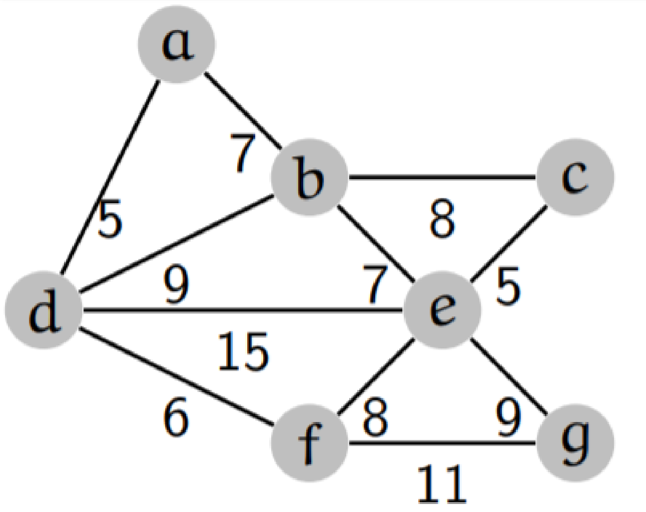
Algorithm for Spanning Tree (DFS)

Input: A connected graph G with vertices ordered $v_1, v_2, \ldots, v_n$
Output: A spanning tree $T = (V', E')$.
$V' = \{v_1\}$, $E' = \emptyset$, $w = v_1$,
while (true)
  while ($\exists wv \in E$ such that T and wv don't create a cycle in T)
    choose the vertex with min k, v_k, that when added to T
    doesn't create a cycle in T
    $E' = E' \cup \{wv_k\}$, $V' = V' \cup \{v_k\}$, $w = v_k$
    end while
    if $w = v_1$
      return T
    $w =$ parent of w in T // backtrack
end while

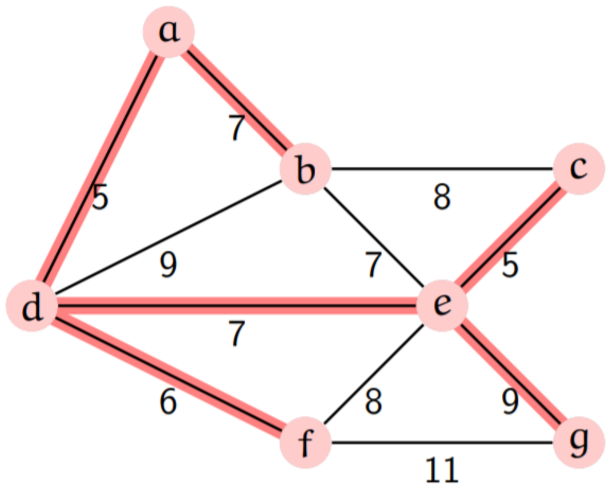
The minimum spanning tree

Definition

A minimum spanning tree of a weighted graph is a spanning tree of least weight (that is, a spanning tree for which the sum of the weights of all its edges is least among all spanning trees).



How to find the minimum spanning tree (Kruskal)



Construct a tree:

Step 1: start with the least edge,

Repeat step 2: add the next least edge, if this won't create a circuit,

Stop: finish when you have 6 edges (Why?)

The length: 39

Kruskal's algorithm formally

Kruskal's algorithm is a greedy algorithm.

- In each phase, a decision is made that appears to be good, without regard for future consequences.

Step 1: Find an edge of least weight and call this e_1. Set $k = 1$.

$e_1 = ad$

Step 2:

while $k < n$

if $\exists e \in E$ such that $\{e, e_1, \ldots, e_k\}$ does not contain a circuit (denote by e_{k+1} such an edge of least weight);

replace k by $k + 1$;

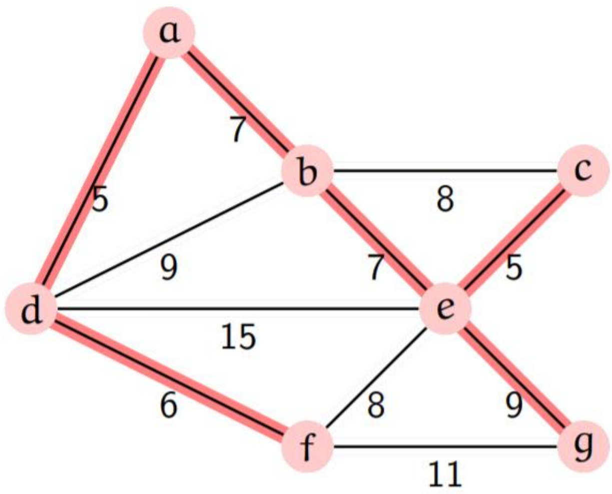
$e_2 = ec, \; e_3 = df, \; e_4 = ab, \; e_5 = de, \; e_6 = eg$

else output $e_1, e_2, \ldots, e_k$ and stop.

end while

How to find the minimum spanning tree (Prim)

It builds a minimum spanning tree T be expanding outward in connected links from some vertex.



Construct a tree T:

Step 0: start with any vertex,

Repeat step 1 six times: add to T an adjacent edge of least weight (blue edges) that connects vertices in T with those not in T

The length: 39

Prim's algorithm formally

Step 0: Initialize $T := \emptyset$

Step 1: Pick any vertex $v \in V$ and set $W = \{v\}$.

$W = \{b\}$

Step 2:

while $W \neq V$

find a min weight edge $\{x, y\}$, where $x \in W$ and $y \in V \setminus W$;

$T := T \cup \{\{x, y\}\}$;

$W := W \cup \{y\}$;

$W = \{b, a\} \to W = \{b, a, d\} \to W = \{b, a, d, f\} \to$

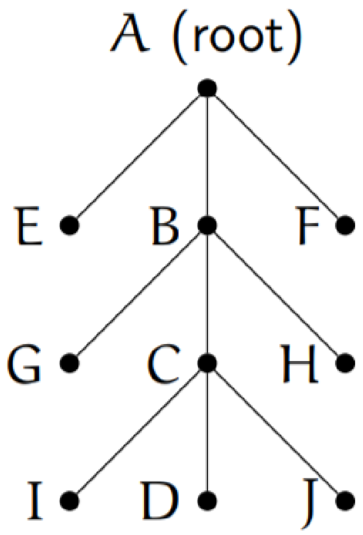
$W = \{b, a, d, f, e\} \to W = \{b, a, d, f, e, c\} \to$

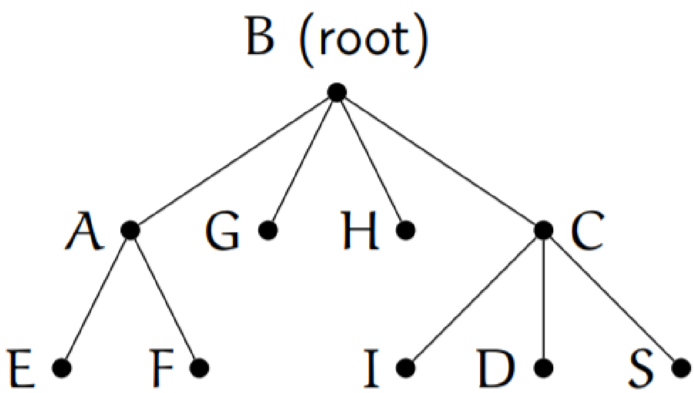
$W = \{b, a, d, f, e, c, g\}$

end while

Root of the tree (data structure)

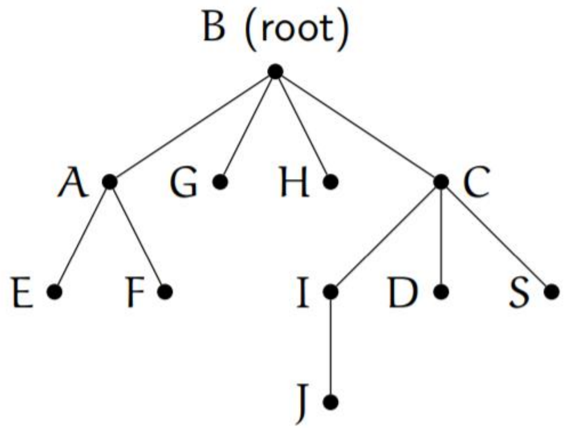
A tree is rooted if it comes with a specified vertex, called the root.





Terminology with rooted tree

- Each vertex in a tree has zero or more children – the vertices "below" it in the tree (our trees are drawn growing downwards).
- A vertex that has a child is called the child's parent vertex.
- If two vertices have the same parent they are called siblings.



Example. The vertex J is a child of I, the vertex I is a child of C. The vertex A is a parent of E and F. The vertices A, G, H and C are siblings.

Network Models and Digraphs

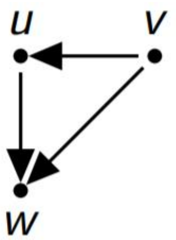
Directed graphs (digraphs)

Graphs are used to model real-life situations; when edges represent roads or pipes then they need to be associated with direction/weights.

Definition

A digraph is a pair (V, E) of sets, V is a nonempty set of vertices, E is a set of ordered pairs of distinct elements of V, called arcs (edges).

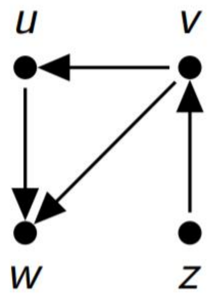
- A digraph can be pictured like a graph with the orientation of an arc indicated by an arrow.
- A digraph is just a graph in which each edge has an orientation or direction assigned to it.



$V = \{u, v, w\}$,
$E = \{vu, vw, uw\}$,
e.g. for vu – the direction is from v to u, uv and vu are different

Indegree and outdegree

- Similarly to graphs, digraphs for us do not contain multiple arcs/loops.
- Each vertex of a digraph has:
  - – an indegree: the number of arcs directed into that vertex, and
  - – an outdegree: the number of arcs directed out of that vertex

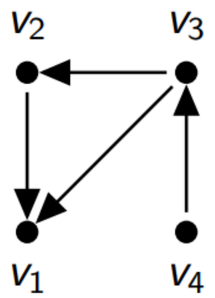


vertex u has indegree 1 and outdegree 1,
vertex v has indegree 1 and outdegree 2,
vertex w has indegree 2 and outdegree 0,
vertex z has indegree 0 and outdegree 1.

or e.g. v has one incoming arc and two outgoing arcs

Terminology and adjacency matrix

- Similar definitions to those for graphs exist for walks, paths . . . . However it is necessary to follow the direction of the arcs.
- The adjacency matrix A of G with vertices v1, v2, . . . , vn is defined by setting aij = 1 if there is an arc from vi to vj (0 otherwise) – A generally is not symmetric.



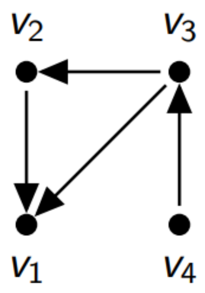
The adjacency matrix $A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

e.g. the element $(3,1)$ is 1 iff $v_3 v_1$ is an arc

Properties of adjacency matrix

Most assertions made about the adjacency matrix for graphs apply with appropriate changes for digraphs as well.

- The outdegree of the vertex vi is the number of 1's in row i; the indegree of the vertex vi is the number of 1's in column i.
- The (i, j) entry of A k is the number of different walks of length k from vi to vj respecting the orientation of arcs, k > 1.

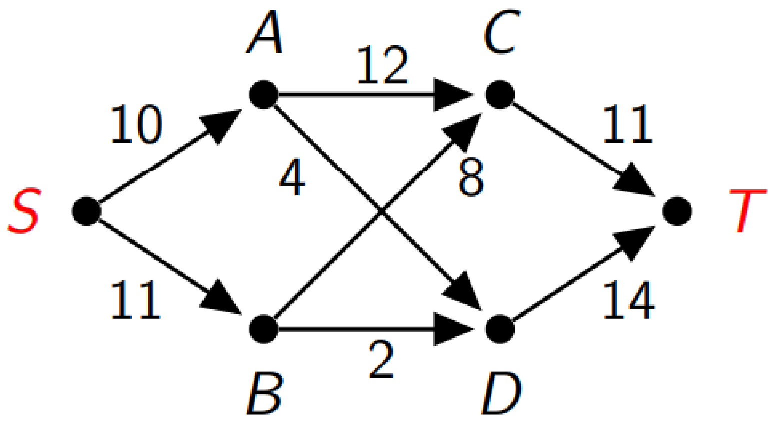


$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad A^2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

e.g. the element $(4,1)$ is 1 iff there is one walk of length 2 from v_4 to v_1

Introduction to network models

Directed graphs are useful for modelling networks problem.



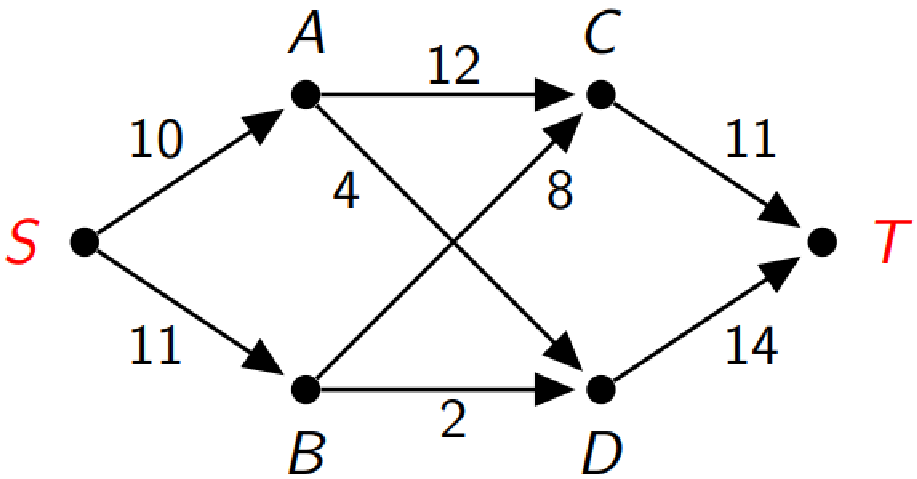
- The network might be:
  - – a transportation network through which commodities flow,
  - – a pipeline network through which oil/gas/. . . flows,
  - – a computer network through which data flows,

In each case the problem is to find a maximum flow.

Maximising the flow in a network is a problem that belongs both to graph theory and to operations research.

An example of a network model

The arcs of a digraph (a network) can represent an oil pipeline network and show the direction the oil can flow



Oil is unloaded at the dock S (the source) and pumped through the network to the refinery T (the sink).

- The weight on the arcs shows the capacities of the pipelines.

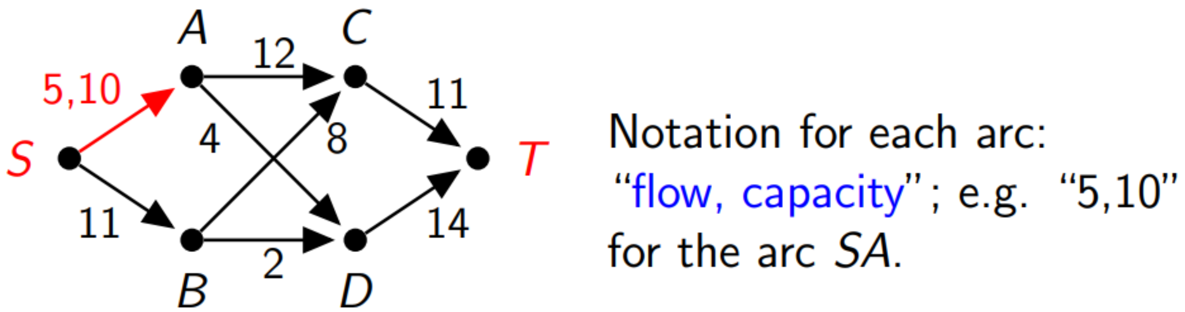
Our goal is to pump as much oil as possible from S to T. To formalise such a concept, we use the term a flow.

What is a flow?

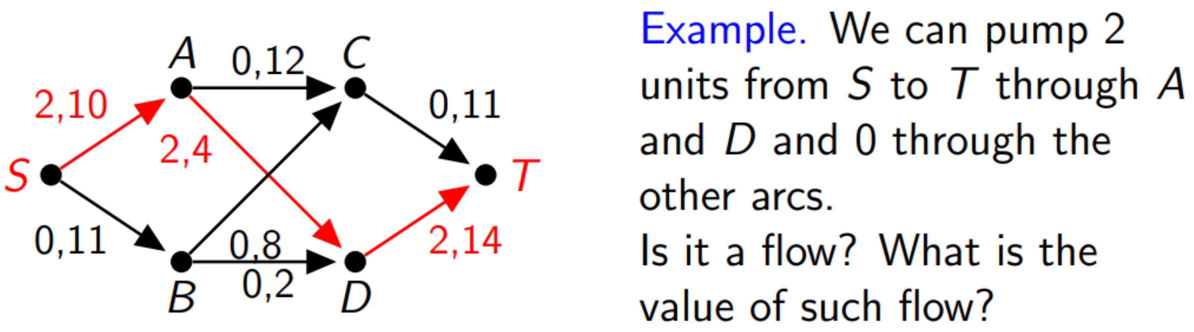
A flow in a network is a description of the amount of commodity that can flow along the network (in unit time).

- No pipe must receive more than it can cope with (“flow 6 capacity” for each arc).
- No commodity must be lost along the way (“flow in = flow out” for all vertices except S, T).

A flow assigns to each arc e a nonnegative number, f (e), subject to the previous two constraints.



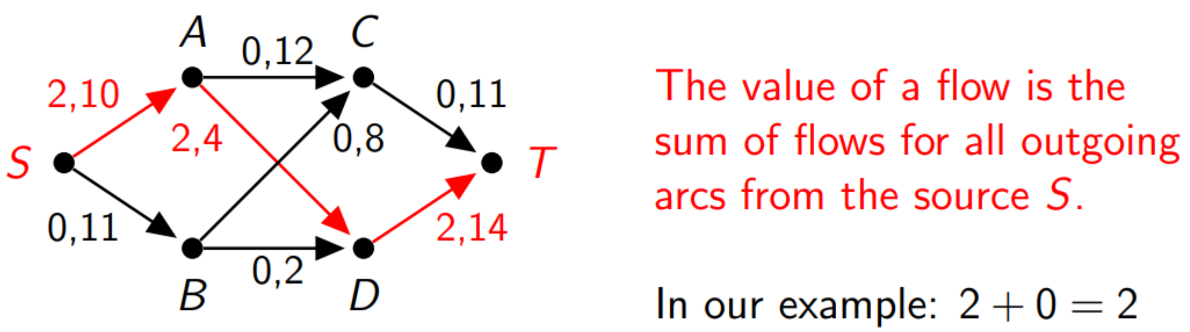
An example of a flow



- This assignment has the properties:
  - – for each arc “flow(e) 6 capacity(e)”,
  - – for each vertex A, B, C, D (all internal vertices) “the flow into each one is equal to the flow out of it”.

Hence, it is a flow.

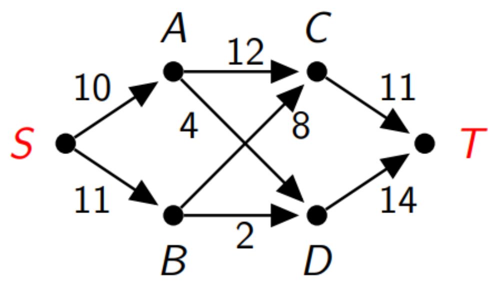
What is a the value of the flow?



The value of a flow must be the same as the sum of flows for all incoming arcs to T. So, the value of this flow is 2.

Formal definition of a network

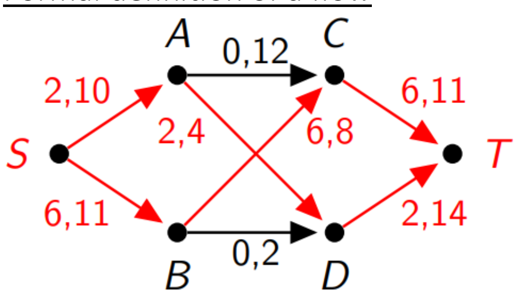
Let G = (V, E) be a directed weighted graph with the following properties:



- a vertex S (source) has no incoming arcs – the start of the flow
- a vertex T (sink) has no outgoing arcs – the end of the flow

- the (nonnegative) weight on each arc is the capacity of the edge e, denoted c(e), c(e) > 0 – the maximum amount of some commodity that can flow through it in unit of time (liters of oil, kW of electricity, # of people, # of messages, . . .)

Formal definition of a flow



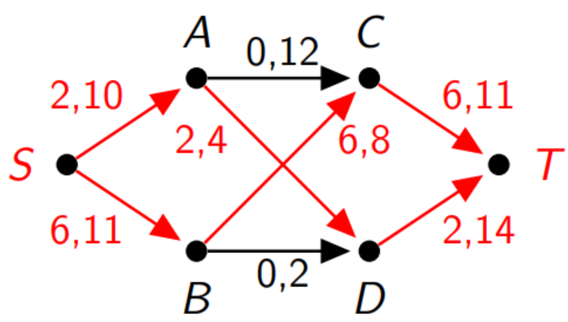
A flow is a mapping that assigns to each edge e a number $f(e)$ satisfying two conditions:

Feasibility condition: 0 6 f (e) 6 c(e) for each edge e ∈ E, in words: the flow along each arc must be less than or equal to the capacity of that arc.

Conservation of flow: $\sum_{v \in V} f(uv) = \sum_{v \in V} f(vu)$ for all $u \in V - \{S, T\}$,

in words: for each internal vertex u (i.e. not S or T) the sum of flows along the arcs into u is equal to the sum of the flows along the arcs out of u

The value of the flow formally



The value of the flow:
$\sum_{v \in V} f(Sv)$
(sum of S-outgoing arcs)
In our example $2 + 6 = 8$

But by the flow conservation law, none of the flow is lost at any vertex.

This means, the value of the flow is also equal the sum of flows of all arcs going into T: $\sum_{v \in V} f(vT)$ ($\sum$ of T-incoming arcs)
$= \sum_{v \in V} f(Sv)$ ($\sum$ of S-outgoing arcs)

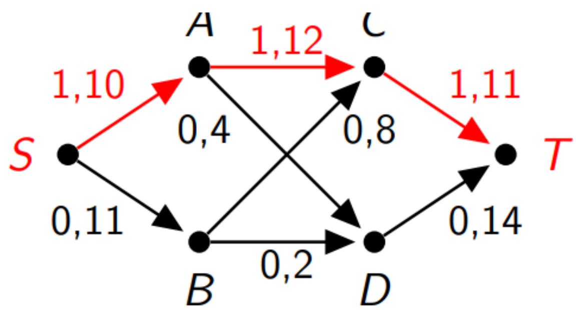
A maximum flow is a flow of maximum value.

The construction of flows

There is a simple way of finding a maximum flow:

- Locate a path P from S to T (which follows the direction specified by the arrows on arcs) and define a flow by setting:

$$f(e) = \begin{cases} 1, & \text{if } e \in P \\ 0, & \text{if } e \notin P \end{cases}$$

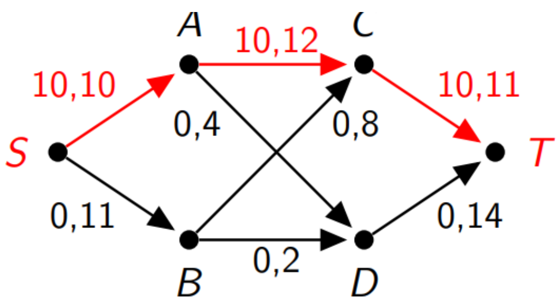


E.g. $SACT$ is such a path.

The value of the flow is 1.

Conservation of flow is guaranteed.

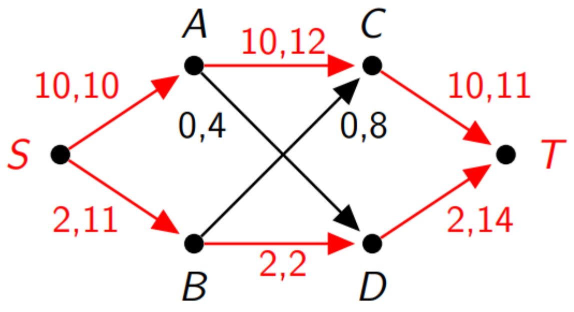
- • Continue to increment by 1 the flow on the arcs of SACT until you reach the smallest capacity on the arcs of SACT.



The path $SACT$ has a saturated arc SA, a flow equal to its capacity. A flow along the path $SACT$ can't be improved.
The value of the flow is 10.

Can we find another path from S to T with an unsaturated edge?

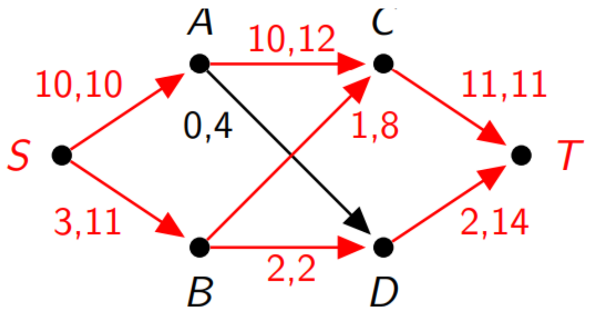
- Yes, e.g. SBDT. And again a flow can be incremented by 1, until an arc on SBDT is saturated.



The path *SBDT* has a saturated arc *BD* (the flow along the path *SBDT* can't be improved).
The value of the flow is 12.

Can we find another path from S to T with an unsaturated edge?

- Yes, e.g. SBCT. And again a flow can be incremented by 1, until an arc on SBCT is saturated.



The path *SBCT* has a saturated arc *CT*.
The value of the flow is 13.

At this stage, every path between S and T contains a saturated arc!

However, this is not the end of the story, the flow can still be increased . . . up to 17 :-) But how?