Announcements

- Error in PC assignment D. About what errors ISBN-13 detects.

MATH 1005/6005 Revision Lecture

For ~~today~~: **"Proofs"**

Preliminary: **arguments**

$[p_1, \ldots, p_n \therefore q]$

premises

conclusion

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

What are $p_1, \ldots, p_n, q$?

statements or statement forms

i.e. placeholders for statements

An argument is valid if

$$p_1 \wedge \cdots \wedge p_n \rightarrow q$$

is a tautology (i.e. "$\rightarrow$" is true for any inputs into the statement forms.

What does an argument not do?

Does not tell us that the premises are true.

All it says is something about truth tables of statement forms.

Arguments are just the structure of a proof.

Some valid arguments:

Direct argument

$$\frac{p \qquad p \to q}{q}$$

$$\frac{p \to q \qquad \neg q}{\neg p}$$

Division into cases

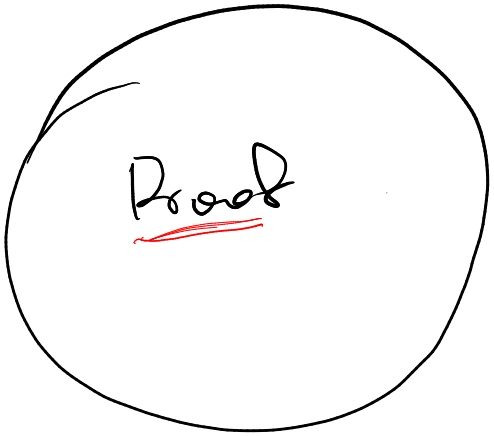
$$\frac{p \vee q \qquad p \to r \qquad q \to r}{r}$$

Proofs:

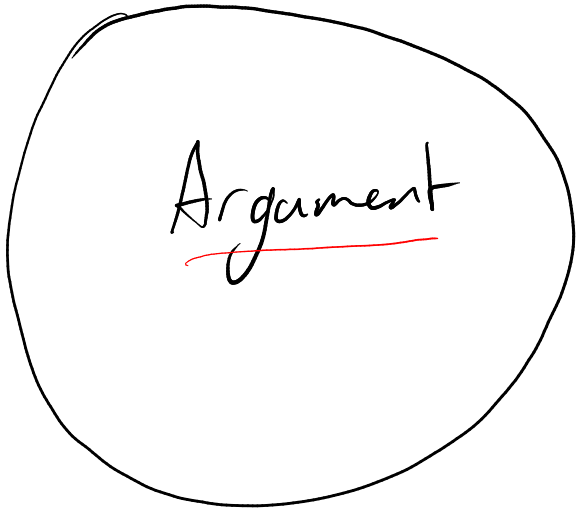
fraction evaluated at $x = 2$

vs.

function at x



vs.



Structure as an argument, but the statement forms are replaced by actual statements

Might need to check each statement is actually true.

Structure built out of statement forms

Valid if premises can be combined to produce the conclusion

Example

Direct argument

$$\frac{p \qquad (p \to q)}{q}$$

Isn't true upon substitution

But what if

p = Is human

q = has seven legs

Note: $p \to q$ is false for this particular p, q.

To prove something, you need to both:

- have a valid argument structure
- Check that all the premises are actually true

How to actually prove things*

*Including $\rightarrow$

1. Figure out if you actually believe the statement
2. Identify the assumptions and conclusion

$[p_1, \ldots, p_n \quad \therefore r]$

3.* Try to decide on a valid argument structure
   Might require new premises

$[p_1, \ldots, p_n, q_1, \ldots, q_m, \therefore]$

new premises

4. Verify any added premises

*: Creative process, might require trial and error, easier with experience

Example:

Theorem: For any $n \in \mathbb{Z}$,

$$f(n) := n^3 + 3n^2 + 2n$$

is divisible by 6.

$\forall n \; p(n)$ where $p(n)$ = "$f(n)$ is divisible by 6"

For this course

$\mathbb{N} = \{1, 2, 3, \ldots\}$

$\mathbb{Z}_{\geq} = \{0, 1, 2, 3, \ldots\}$

$\mathbb{Z} = \{\ldots, -1, 0, 1, \ldots\}$

Do we believe the statement? $f(0) = 0$, $f(1) = 6$, $f(-1) = 0$, $f(2) = 24$ all divisible by 6

Assumption(s): $n \in \mathbb{Z}$ Conclusion: $f(n)$ is divisible by 6

Structure: (Direct) / contrapositive / ~~counter~~example / contradiction

- Direct
- contrapositive: assume $f(n)$ not divisible by 6 ... Find $n \notin \mathbb{Z}$
- ~~counter~~example: since we believe the statement ... $\exists n \; \neg p(n)$
- contradiction: tricky

$\neg(p \to r) \to c$

column of "false" in a truth table

What do the different structures look like?

Direct:

→ Suppose ($n \in \mathbb{Z}$) — p

⋮

→ So ($f(n)$ is divisible by 6) — r

Contrapositive:

Suppose ($f(n)$ is not divisible by 6) — $\neg r$

⋮

So ($n \notin \mathbb{Z}$.) — $\neg p$

How do we fill in the gap?

$$\frac{p \quad ??? \quad \cdots}{r}$$

Some (direct) options

"$p \to r$"

"$p \to p_1 \vee \cdots \vee p_n$"

$p_1 \to r$

⋮

$p_n \to r$

e.g. $n \in \mathbb{Z} \to (n \text{ even}) \vee (n \text{ odd})$

n odd $\to \wedge$

n even $\to \wedge$

Let's go with the first option (since I already know it works).

So our proof is

Suppose $n \in \mathbb{Z}$. (p)

Then $f(n) = n^3 + 3n^2 + 2n$

$= n(n^2 + 3n + 2)$

$= n(n+1)(n+2)$

$p \to r$

- So $f(n)$ is the product of three consecutive integers. Atleast one will be even, and at least one will be divisible by 3.

Creative, and difficult to teach

So $f(n)$ is divisible by 6. (r) ✓

Other sorts of statements and proof strategies

For any statement, the first thing to do is

statement form	Prove	Disprove
$\forall x \quad p(x)$	Assume x is fixed but arbitrary, argue $p(x)$ is true & show $\exists x \neg p(x)$ by contradiction	• •
$\exists x \quad p(x)$	• Find an example • Show $\forall x \neg p(x)$ must be false by contradiction	
$p \to q$	already covered	$\neg(p \to q) \equiv p \wedge \neg q$ So come up with an example with p and $\neg q$
$p \leftrightarrow q$	$p \to q$ and $q \to p$	Counter example of $p \to q$ or $q \to p$

$\neg(\exists x \; p(x)) \equiv \forall x \neg p(x)$

$\neg(\forall x \; p(x)) \equiv \exists x \neg p(x)$

$\forall x$ $p(x)$ example

Statement:

$(\forall z \in \mathbb{Z}, \underbrace{z^2 + 3z + 1}_{p} \text{ is odd})$

r

This statement is true

Try ~~$p \to r$~~

Try $\begin{cases} p \to e \vee o \\ e \to r \\ o \to r \end{cases}$

$(e: z \text{ is even}, \quad o: z \text{ is odd})$

Proof:

Suppose $z \in \mathbb{Z}$

Then z is even or odd (by even/odd theorem).

If z is even, $\exists k \in \mathbb{Z}$ with $z = 2k$

So $z^2 + 3z + 1 = 4k^2 + 6k + 1$

$= 2(2k^2 + 3k) + 1$ odd

If z is odd, $\exists k \in \mathbb{Z}$ with $z = 2k+1$

So $z^2 + 3z + 1 = (2k+1)^2 + 3(2k+1) + 1$

$= 4k^2 + 4k + 1 + 6k + 3 + 1$

$= 4k^2 + 10k + 5$

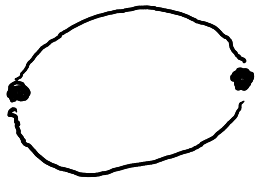
$= 2(2k^2 + 5k + 2) + 1$ odd

So $z^2 + 3z + 1$ is odd.

$\exists x$ $p(x)$ example

Statement:
There exists a graph with at least two vertices, which has a circuit which is both a Hamiltonian circuit and an Euler circuit.

Can prove by giving an example.



Has a circuit which is Euler and Hamiltonian, and it has at least two vertices.

Disproving $\exists$, $\forall$?

Disprove $\exists x \; p(x)$ $\rightsquigarrow$ Prove $\forall x \; \neg p(x)$

$\equiv$ $\forall x \; p(x)$ $\rightsquigarrow$ $\equiv$ $\exists x \; \neg p(x)$

Disprove → example:

$\neg(p \to q) \equiv p \wedge \neg q$

$\neg(p \to (q_1 \vee q_2)) \equiv p \wedge \neg(q_1 \vee q_2)$

$\equiv p \wedge \neg q_1 \wedge \neg q_2$

Statement:

Let $a, b, c \in \mathbb{N}$. (If (p) c divides ab), then (c divides a) (q_1) or (c divides b.) (q_2)

$q = q_1 \vee q_2$

Counter example:

$a = 4, b = 3, c = 6$

2 2 4

$p \wedge$ (c is prime) $\to$ $q_1 \vee q_2$

is actually true.

General tips:

- Check if you think the statement is true or not.
- If there's something about even/odd, then splitting into cases might work
- Think for a while about what method (direct/contrapositive) might work better before you start writing the proof.
- $\forall n \in \mathbb{N}\ P(n)$ — (Directly for each n individually
  — (Induction

  Induction isn't always the answer

  Eg: $\forall n \in \mathbb{N}$, $2n$ is even

- Practice makes it easier, no shortcut / general strategy.

Rational Numbers and IEEE half-precision floating point

What are rational numbers?

Ratios / fractions involving whole numbers

eg $\frac{1}{2}$, $\frac{2}{4}$, $\frac{17}{1000}$, $= 0.017$

At a more abstract level:

$$\mathbb{Q} = \left\{ \begin{array}{l} \text{equivalence classes in} \\ \mathbb{Z} \times (\mathbb{Z} \setminus \{0\}) \quad \text{with relation} \\ (m,n) \; R \; (p,q) \quad \text{if and only if} \\ mq = pn \\ \text{"} \frac{m}{n} = \frac{p}{q} \text{"} \end{array} \right\}$$

How can we represent rational numbers?

Idea 1: Encode $\frac{p}{q}$ as $(p,q) \in \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$

Not good! Lots of representations of the same number

$\frac{4031}{4032} \approx 1$, but they have very different representations (except $1 = \frac{4032}{4032}$)

Idea 2: Pick a base $b \in \{2,3,4,\dots\}$, and try write

$$q = a_N b^N + a_{N-1} b^{N-1} + \dots, \quad a_N \in \{0, \dots, b-1\}$$

$$(a_N a_{N-1} \dots a_0 \cdot a_{-1} \dots)_b$$

Eg $b=10$, $\frac{1}{3} = 0 \times 10^0 + 3 \times 10^{-1} + 3 \times 10^{-2} + \dots = (0.33\dots)_{10}$

$b=2$

Idea 3: scientific notation

$$q = (-1)^s \, m \, b^n \,,$$

$s \in \{0, 1\}$, sign bit

$n \in \mathbb{Z}$, exponent

$1 \leq m < b$, mantissa

combined with

$$m = (a_0 . a_{-1} a_{-2} \dots)_b$$

IEEE half-precision:

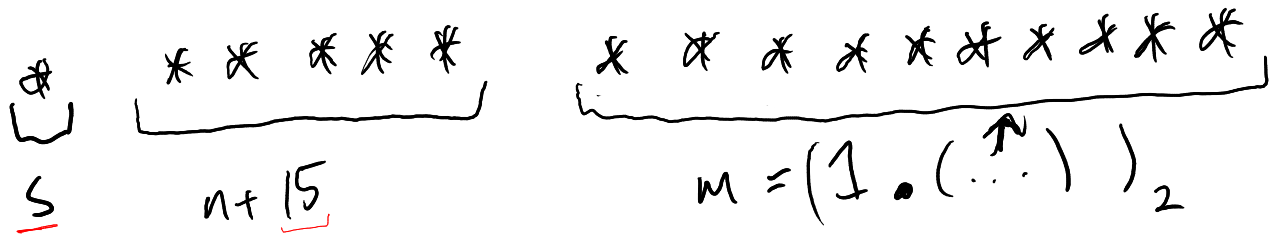
If we take $b = 2$ (i.e. what computers naturally handle)

we can optimise a bit further:

m will always be of the form: $1 \leq m < 2$

So: $m = (1.\ldots)_2$

We can package everything into a bit-string as follows:

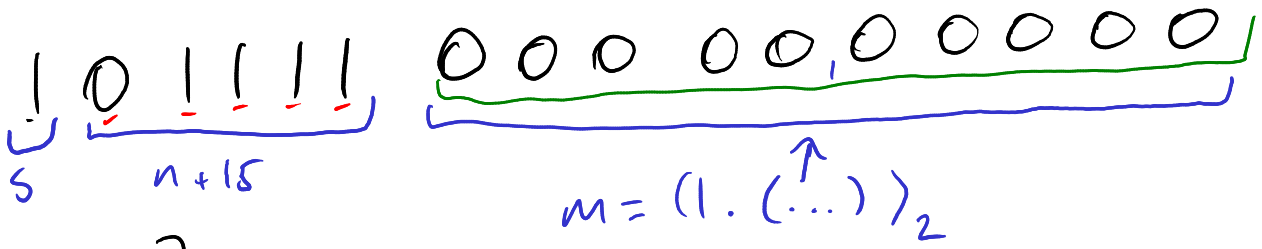


allows for n positive or negative w/o a new sign bit

(or with a different number of bits for exponent / mantissa or different offset)

Example

Suppose x is stored in half-precision floating point as



What is x?

$$n+15 = 0\times 2^4 + 1\times 2^3 + 1\times 2^2 + 1\times 2^1 + 1\times 2^0$$
$$= 8+4+2+1$$
$$= 15$$

So $n=0$

$$x = (-1)^s m\, b^{\wedge} = -(1.00000\ 00000)_2 \times 1 = -1$$

Example

Suppose $y = (10110.0)_2$.

half-precision floating point.

Represent y in

$21 = 16 + 4 + 1$

$= 16 + 4 + 0$

$s = 0$

$n + 15 = 5 + 15 = (21)_{10} = (10100)_2$ (16 8 4 2 1)

$m = 1.01100\ 00000$

So

0 10100 01100 00000

is the representation of y.

Pigeon-hole Principle

If N objects are classified into k categories, at least one category has at least $\lceil \frac{N}{k} \rceil$ objects

↳ smallest whole number bigger than N/k

Example:

A group of n people greet each other, with some shaking others' hands. Then at least two people have shaken the same number of hands.

Equiv.: In any simple graph with n vertices, at least two vertices have the same degree

Pigeons: n people

Holes: labelled by # of handshakes

Actually: $|\{0, \ldots, n-2\}| = n-1$

or $|\{1, \ldots, n-1\}| = n-1$

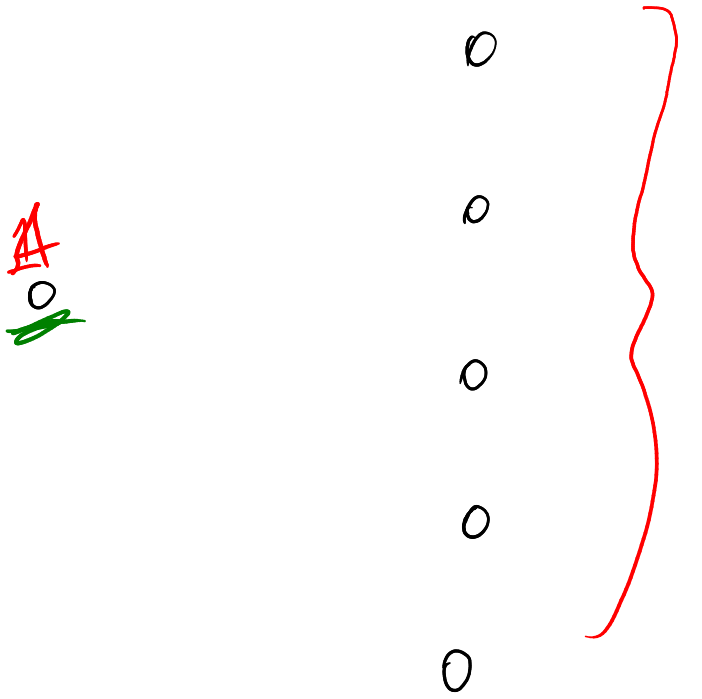
$|\{0, 1, \ldots, n-1\}| = n$

If someone shook no hands, nobody could have shaken all the hands

$n-1 < n$, so pigeonhole principle means two people shook the same number of hands

Example:

Show that in any group of 6 people, there are three who have either all met before or all not met before.



Suppose 3 have met A. Amongst the three, either all three haven't met or two have, together with A make 3.

Permutations and Combinations

Permutations: r out of n care about order

$$n(n-1)(n-2)\dots(n-r+1) = \frac{n!}{(n-r)!}$$

Combinations: r out of n don't care about order

Same, but you can reorder in $r!$ ways

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

DIY Examples of Permutations

Award ceremony

Counting letter rearrangements

Seating arrangements.

DIY Examples of combinations

Any of the previous but you've forgotten everybody's names.

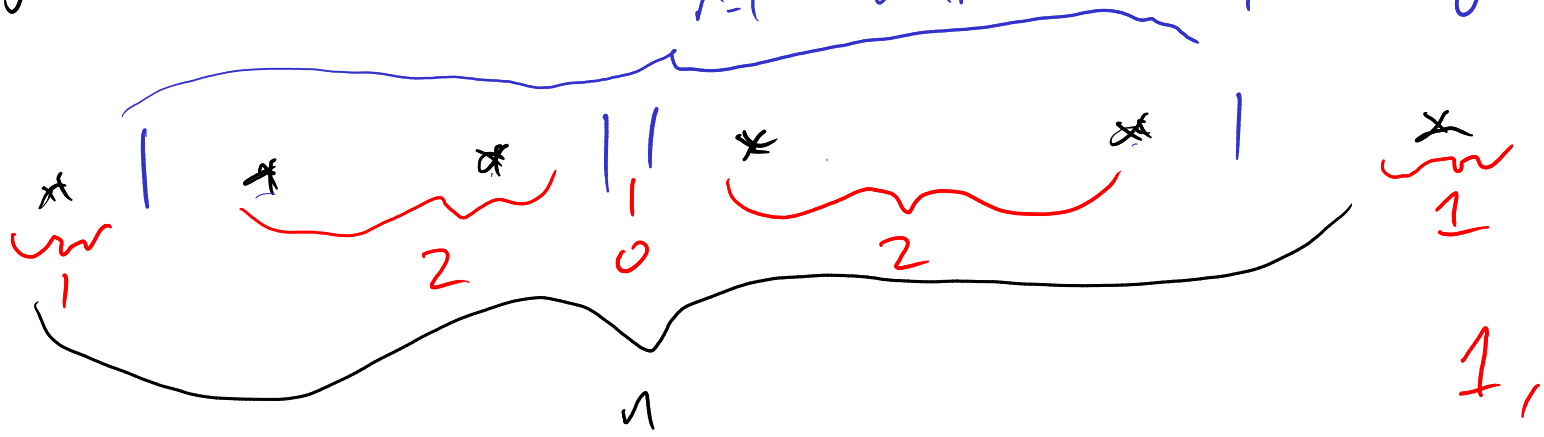
Classes for next semester

Announcements

- SELTs, please submit constructive
- PC assignment D correction
- Consultation hour will continue until exam.
  Possibly extra sessions

Stars and Bars

Idea: want to count the number of ways to split n objects into r groups (possibly empty).



6 things into 5 groups

1, 2, 0, 2, 1

Pick $r-1$ spots to put bars, filling all others with stars.

$$\binom{n+r-1}{n} = \binom{n+r-1}{r-1}$$

ways of doing this

$n+r-1$ places for n stars and $r-1$ bars

Example:

A 4-tuple of numbers $(x_1, \dots, x_4)$ with each $x_i \in \{0, 1, 2, \dots\}$ is called <u>valid</u> if $x_1 + x_2 + x_3 + x_4 = 10$.

How many valid 4-tuples are there?

10 stars, 4 groups $\sim$ 3 bars

$$\binom{10+4-1}{4-1} = \binom{13}{3} = \binom{13}{10}$$

What if $x_i \in \{1, \dots\}$, set $y_i := x_i - 1 \in \{0, \dots\}$

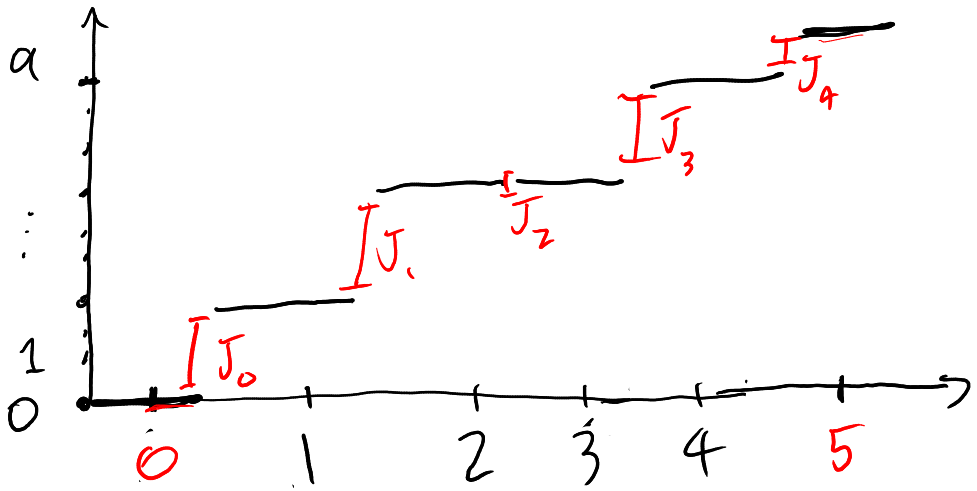
$$y_1 + \dots + y_4 = x_1 + \dots + x_4 - 4 = 6$$

6 stars, 4 groups.

Difficult Example:

How many functions $f: \{1, 2, 3, 4\} \to \{0, 1, \ldots, 9\}$ are non-decreasing? (Non-decreasing means $(x \le y) \Rightarrow (f(x) \le f(y))$)

Hint:



$J_0 + \ldots + J_4 = 9$

$J_i \in \{0, 1, \ldots\}$

If strictly increasing

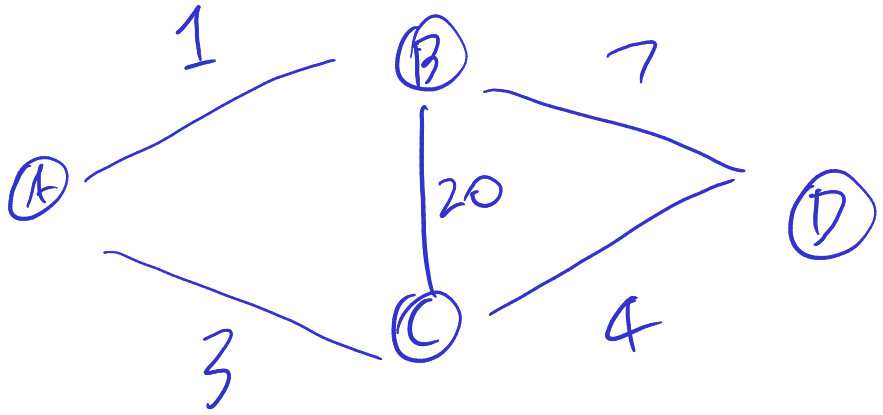
$J_i \in \{1, \ldots\}$

9 stars, 5 groups,

$$\binom{9+5-1}{5-1} = \binom{13}{4}$$

Shortest Path, Dijkstras Algorithm:

Idea:



length of edges.

Want: shortest path between two fixed vertices.

Dijkstra's Algorithm — A Formal Description

Input: (a) Connected simple graph G. Vertices A, Z from G.
(b) Distance function $\text{dist}: E(G) \to \mathbb{Q}^+$.

Output: (a) Tree T containing A and Z as vertices.
T is a subgraph of G.
The unique path A$\to$Z in T has minimal total distance of all paths A$\to$Z in G.
(b) 'Labelling' $L: V(T) \to \mathbb{Q}_+$; $L(v) = \text{min.dist}(\text{A} \to v)$.

Method:

1. Initialize the tree T: Set $V(T) = \{\text{A}\}$, $E(T) = \emptyset$.
2. Initialize a 'Marking' function $M: V(G) \to V(G) \cup \{\text{blank}\}$:
   Set $M(v) = \text{blank}$ for all $v \in G$.
3. Set $L(\text{A}) = 0$. 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 $M(v) = c$, $L(v) = L(c) + \text{dist}(\{c, v\})$.

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

Example:

1. Initialize the tree T: Set $V(T) = \{A\}$, $E(T) = \emptyset$.
2. Initialize a 'Marking' function $M : V(G) \to V(G) \cup \{\text{blank}\}$: Set $M(v) = \text{blank}$ for all $v \in G$.
3. Set $L(A) = 0$. Set 'current vertex' c to A.

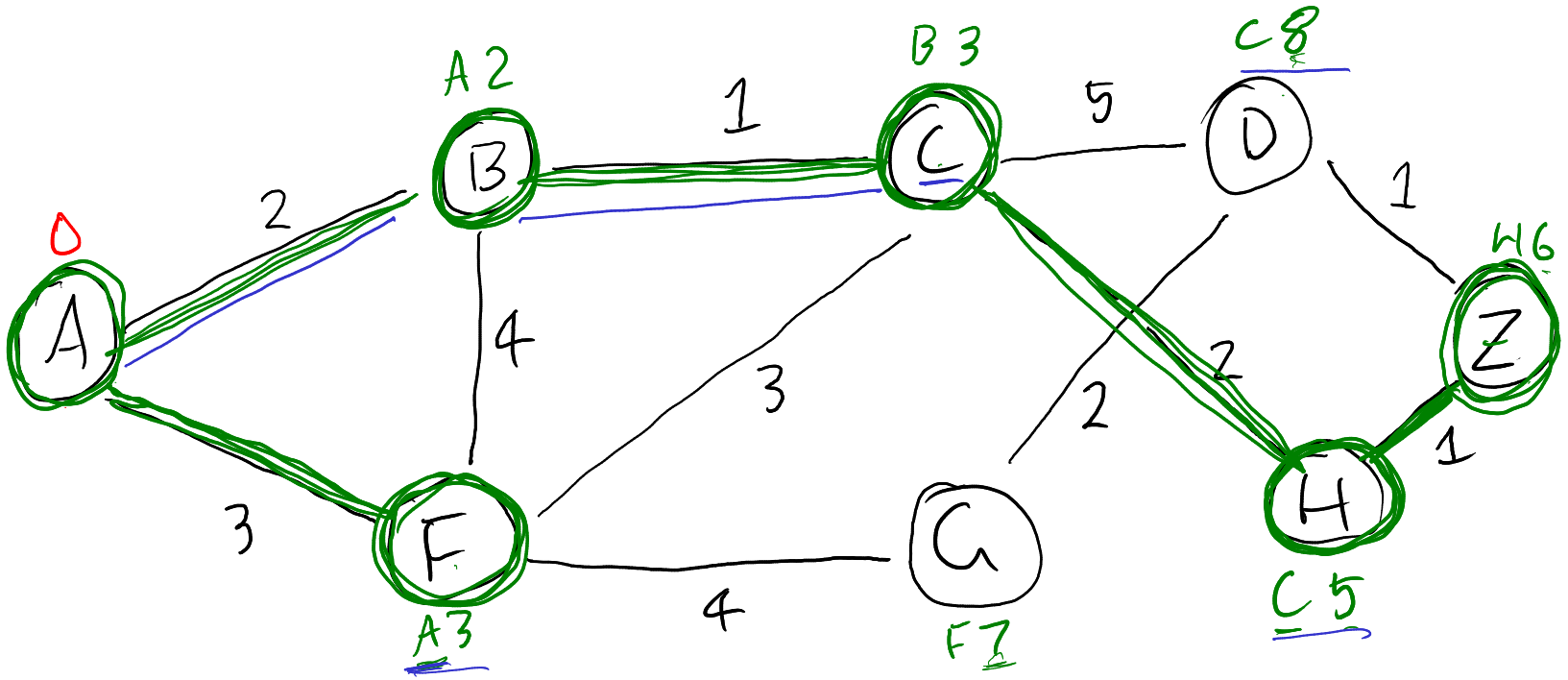
While $c \neq Z$:

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 $M(v) = c$, $L(v) = L(c) + \text{dist}(\{c, v\})$.
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.)

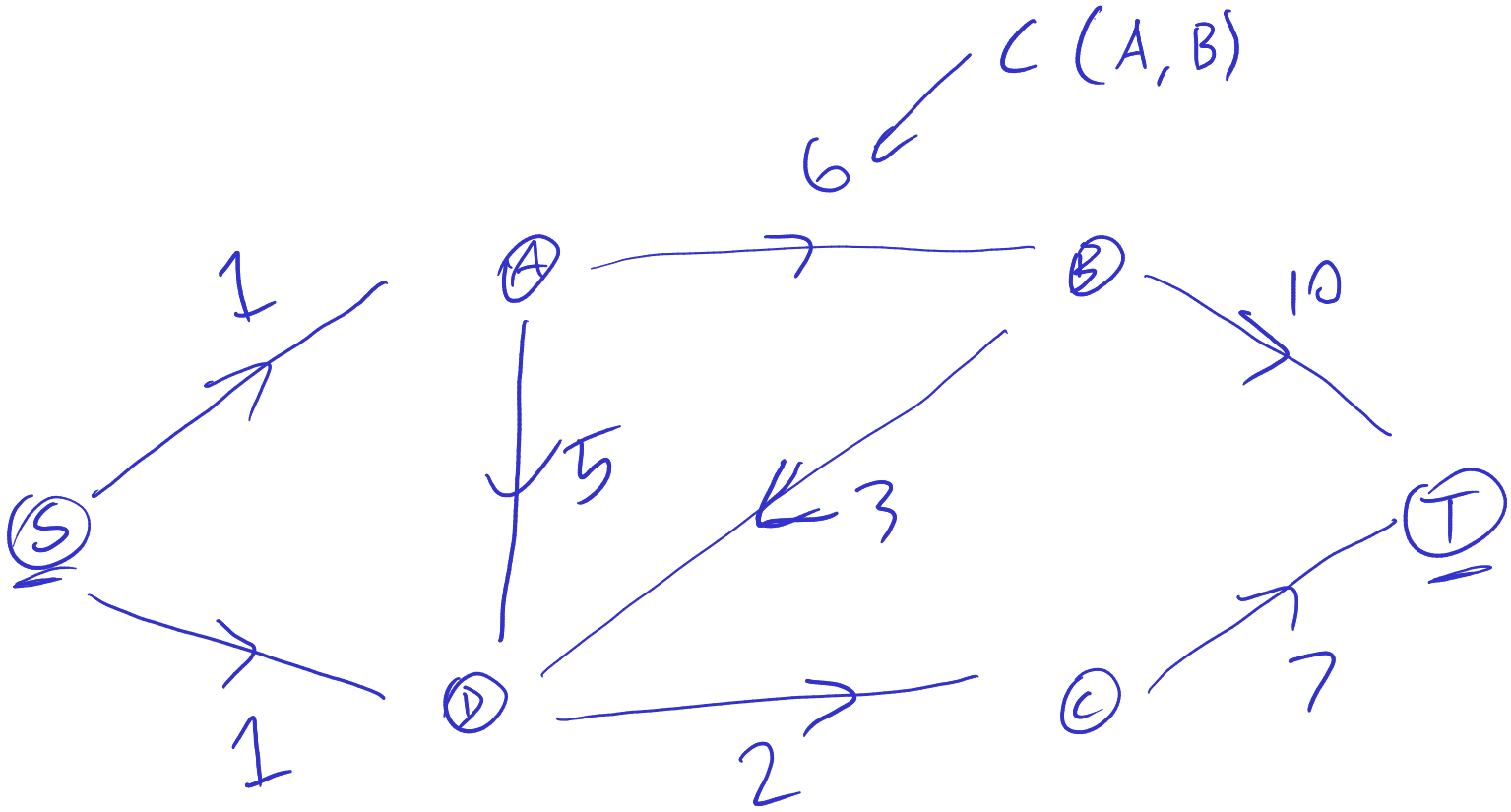
End of While Loop

$T = \{A, B, C, F, H, Z\}$

T gives shortest paths from A to any vertex on the tree.



Transport Networks, Vertex labelling algorithm, max. flow / min. cut, virtual flow



Transport Networks, Vertex labelling algorithm, max. flow / min. cut, virtual flow

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 $i = 1, 2, \ldots$ 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.

$$S(u,v) = \begin{cases} C(u,v) - F(u,v), & (u,v) \in E \\ F(v,u), & (v,u) \in E \\ 0, & \text{otherwise} \end{cases}$$

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. 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 $u_0 u_1 \ldots u_n$ 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

Transport Networks, Vertex labelling algorithm, max. flow / min. cut, virtual flow

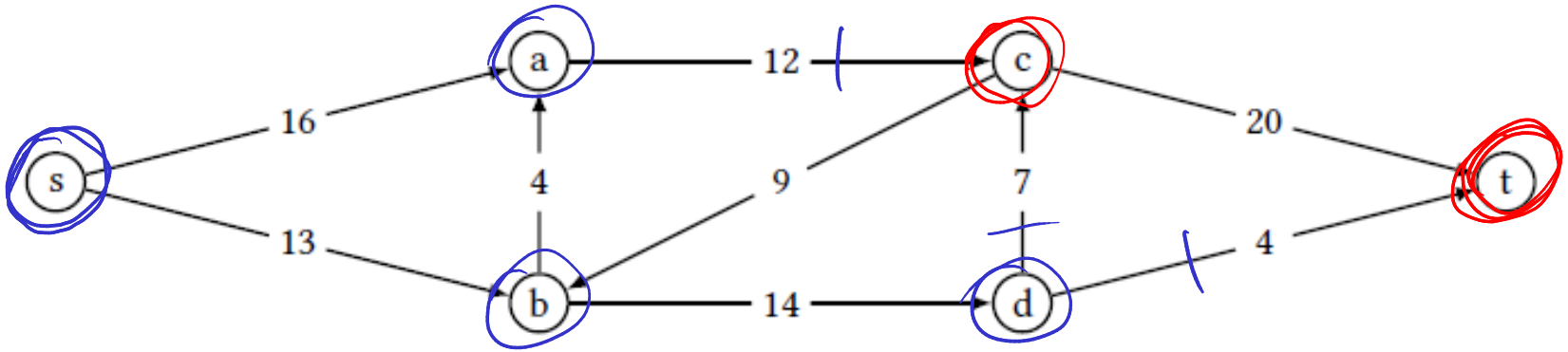
A set K of edges in a transport network D is called a **cut** when there is a partition $\{S, T\}$ of $V(D)$ with $s \in S$ and $t \in T$ such that K comprises all edges of D that start in S and finish in T; *i.e.* $K = E(D) \cap (S \times T)$.

The **capacity of a cut** K is $\sum_{e \in K} C(e)$.

Max flow min cut theorem: For any transport network:

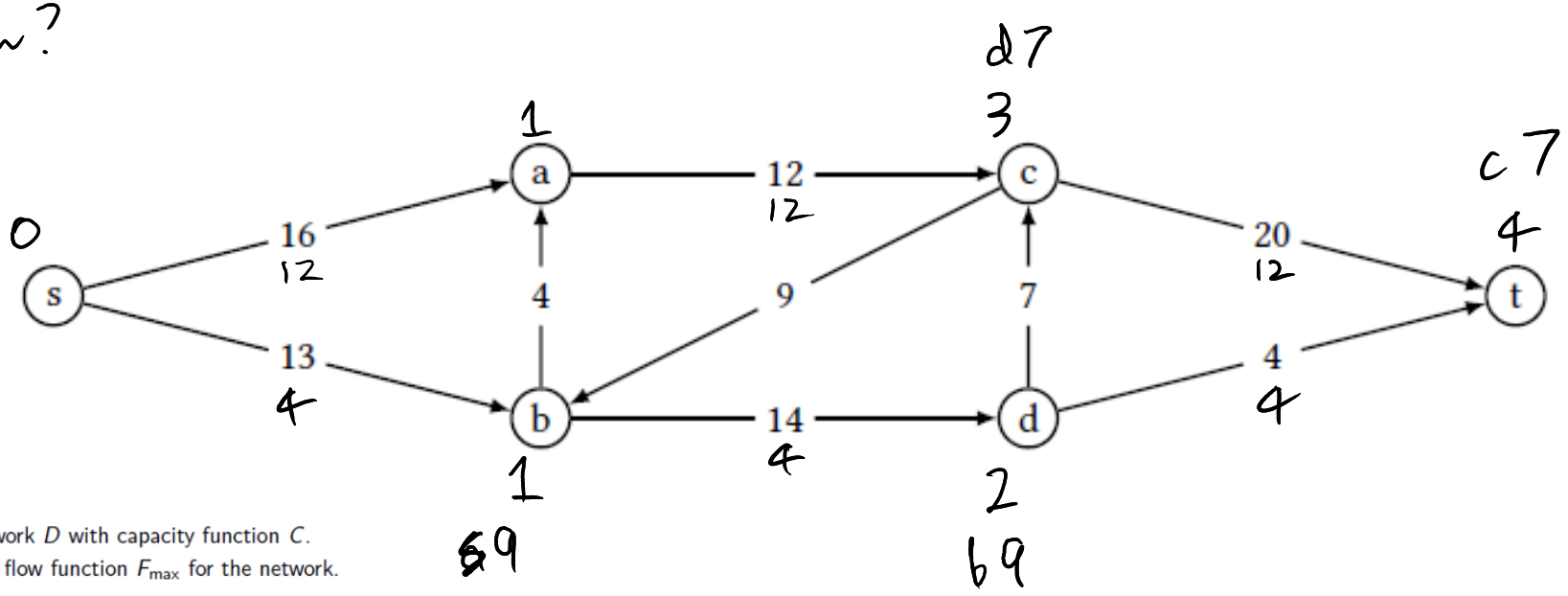
maximum flow value = minimum cut capacity

Example:



Min cut? 23

Max flow?



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 $i = 1, 2, \ldots$ 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. 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 $u_0u_1 \ldots u_n$ 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

$f_1 = 12$, s a c t

$f_2 = 4$ s b d t

$f_3 = 7$ s b d c t

↑

add to 23, max flow

Transport Networks, Vertex labelling algorithm, max. flow / min. cut, virtual flow