CS3230 Prerequisites Revision

CS3230 AY21/22 Sem 2

Christian James Welly
cjwelly@u.nus.edu

My other [materials]

Motivation

- In CS3230, we will need to do quite a bit of proving!
- Important to refresh your Discrete Mathematics (CS1231/S or equivalent)

Motivation

- In CS3230, we will need to do quite a bit of proving!
- Important to refresh your Discrete Mathematics (CS1231/S or equivalent)

- Note: Not exhaustive! I will just give a quick survey of several concepts which we might assume you should have known (based on my personal experience)

Table of Contents

Click on the link to jump to the relevant sections!

Links

Video Recording: https://youtu.be/RdAAknALqjs

Link to these slides (if you have questions, I prefer you comment on the Google Slides directly, in case someone else has the same question!):

https://docs.google.com/presentation/d/1iDOdO803berqZWczEsrRsH15OQUvx5VF94pCvOI_AeI/edit?usp=sharing

Changelog (after the video has been recorded)

- Proof By Induction example: "*positive integers* ***z***" -> "*positive integers* ***n***"
- Equivalent Statement of Trees: "*|V| = |E|* ***-*** *1*" -> "*|V| = |E|* ***+*** *1*"
- Number of edges in complete graph: nC2 should be "n(n**-**1)/2" instead of "n(n**+**1)/2"
- Added links to video/slide and direct links from table of contents
- Added more explanation on Proof by Contradiction vs Contraposition

Last updated on: 10 Jan 2021, 11:31PM

Maths

Sum of Arithmetic Progression (a common one)

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + (n - 1) + n$$

Sum of Arithmetic Progression (a common one)

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + (n-1) + n$$

$$\sum_{i=1}^{n} i = n + (n-1) + \ldots + 2 + 1$$

(By commutativity, rewrite the sum from back to front)

Sum of Arithmetic Progression (a common one)

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + (n - 1) + n$$

$$\sum_{i=1}^{n} i = n + (n - 1) + \ldots + 2 + 1$$

(By commutativity, rewrite the sum from back to front)

$$+$$

$$2\sum_{i=1}^{n} i = (n + 1) + (n + 1) + \ldots + (n + 1) + (n + 1)$$

Sum of Arithmetic Progression (a common one)

$$\sum_{i=1}^{n} i = 1 + 2 + \dots + (n-1) + n$$

$$\sum_{i=1}^{n} i = n + (n-1) + \dots + 2 + 1$$

(By commutativity, rewrite the sum from back to front)

+

$$2\sum_{i=1}^{n} i = \underbrace{(n+1) + (n+1) + \dots + (n+1) + (n+1)}_{n \text{ terms}}$$

Sum of Arithmetic Progression (a common one)

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + (n - 1) + n$$

$$\sum_{i=1}^{n} i = n + (n - 1) + \ldots + 2 + 1$$

(By commutativity, rewrite the sum from back to front)

$$2\sum_{i=1}^{n} i = \underbrace{(n + 1) + (n + 1) + \ldots + (n + 1) + (n + 1)}_{n \text{ terms}}$$

$$= n\,(n + 1)$$

Sum of Arithmetic Progression (a common one)

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + (n - 1) + n$$

$$\sum_{i=1}^{n} i = n + (n - 1) + \ldots + 2 + 1$$

(By commutativity, rewrite the sum from back to front)

$$+$$

$$2\sum_{i=1}^{n} i = \underbrace{(n + 1) + (n + 1) + \ldots + (n + 1) + (n + 1)}_{n \text{ terms}}$$

$$= n\,(n + 1)$$

$$\sum_{i=1}^{n} i = (n\,(n + 1)) / 2$$

(Dividing both sides by 2)

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

Sum of Arithmetic Progression (a common one)

$$\sum_{i=1}^{n} i = 1 + 2 + \ldots + (n - 1) + n$$

$$\sum_{i=1}^{n} i = n + (n - 1) + \ldots + 2 + 1$$

(By commutativity, rewrite the sum from back to front)

$+$

$$2\sum_{i=1}^{n} i = \underbrace{(n + 1) + (n + 1) + \ldots + (n + 1) + (n + 1)}_{n \text{ terms}}$$

$$= n\,(n + 1)$$

$$\boxed{\begin{aligned}\sum_{i=1}^{n} i &= (n\,(n + 1)) / 2 \\ &= (n^2 + n) / 2\end{aligned}}$$

(Dividing both sides by 2)

Sum of (finite) Geometric Progression

a: first term
r: common ratio

$$\sum_{i=0}^{n-1} ar^i = a + ar + ar^2 + \ldots + ar^{(n-1)}$$

Sum of (finite) Geometric Progression

a: first term
r: common ratio

$$\sum_{i=0}^{n-1} ar^i = a + ar + ar^2 + \dots + ar^{(n-1)}$$

Multiply the whole thing by r

$$r\sum_{i=0}^{n-1} ar^i = ar + ar^2 + \dots + ar^{(n-1)} + ar^n$$

Sum of (finite) Geometric Progression

a: first term
r: common ratio

$$\sum_{i=0}^{n-1} ar^i = a + \cancel{ar} + \cancel{ar^2} + \cancel{\ldots} + \cancel{ar^{(n-1)}}$$

$$r\sum_{i=0}^{n-1} ar^i = \cancel{ar} + \cancel{ar^2} + \cancel{\ldots} + \cancel{ar^{(n-1)}} + ar^n$$

$$(1 - r)\sum_{i=0}^{n-1} ar^i = a - ar^n$$

do subtractionsss
the terms are
cancelled!

Sum of (finite) Geometric Progression

a: first term
r: common ratio

$$\sum_{i=0}^{n-1} ar^i = a + ar + ar^2 + \dots + ar^{(n-1)}$$

$$r\sum_{i=0}^{n-1} ar^i = ar + ar^2 + \dots + ar^{(n-1)} + ar^n$$

$$-$$

$$(1-r)\sum_{i=0}^{n-1} ar^i = a - ar^n$$

do divisonssss

$$\sum_{i=0}^{n-1} ar^i = \frac{(a - ar^n)}{(1-r)}$$

Sum of (finite) Geometric Progression

a: first term
r: common ratio

$$\sum_{i=0}^{n-1} ar^i = a + ar + ar^2 + \dots + ar^{(n-1)}$$

$$r\sum_{i=0}^{n-1} ar^i = ar + ar^2 + \dots + ar^{(n-1)} + ar^n$$

$$(1-r)\sum_{i=0}^{n-1} ar^i = a - ar^n$$

$$\sum_{i=0}^{n-1} ar^i = \frac{(a - ar^n)}{(1-r)}$$

$$= \frac{a(1-r^n)}{(1-r)}$$

do factorisationsss

Sum of Geometric Progression (cont)

$$\sum_{i=0}^{n-1} ar^i = \boxed{\frac{a(1 - r^n)}{(1 - r)}} = \frac{-1}{-1}\,\frac{a(1 - r^n)}{(1 - r)} = \boxed{\frac{a(r^n - 1)}{(r - 1)}}$$

Sum of Geometric Progression (cont)

$$\sum_{i=0}^{n-1} ar^i = \frac{a(1 - r^n)}{(1 - r)} = \frac{-1}{-1}\,\frac{a(1 - r^n)}{(1 - r)} = \frac{a(r^n - 1)}{(r - 1)}$$

Special case when $|r| < 1$: As $n \to \infty$, $r^n \to 0$

Sum of Geometric Progression (cont)

$$\sum_{i=0}^{n-1} ar^i = \frac{a(1 - r^n)}{(1 - r)} = \frac{-1}{-1}\frac{a(1 - r^n)}{(1 - r)} = \frac{a(r^n - 1)}{(r - 1)}$$

Special case when $|r| < 1$: As $n \dashrightarrow \infty$, $r^n \dashrightarrow 0$

$$\sum_{i=0}^{n-1} ar^i < \sum_{i=0}^{\infty} ar^i$$

Intuition: put more and more things, so it's definitely larger :D

Sum of Geometric Progression (cont)

$$\sum_{i=0}^{n-1} ar^i = \frac{a(1 - r^n)}{(1 - r)} = \frac{-1}{-1}\,\frac{a(1 - r^n)}{(1 - r)} = \frac{a(r^n - 1)}{(r - 1)}$$

Special case when $|r| < 1$: As $n \to \infty$, $r^n \to 0$

$$\sum_{i=0}^{n-1} ar^i < \sum_{i=0}^{\infty} ar^i$$

$$= \frac{a(1 - 0)}{(1 - r)}$$

Sum of Geometric Progression (cont)

$$\sum_{i=0}^{n-1} ar^i = \frac{a(1 - r^n)}{(1 - r)} = \frac{-1}{-1}\,\frac{a(1 - r^n)}{(1 - r)} = \frac{a(r^n - 1)}{(r - 1)}$$

Special case when $|r| < 1$: As n --> ∞, r^n --> 0

$$\sum_{i=0}^{n-1} ar^i < \sum_{i=0}^{\infty} ar^i$$

$$= \frac{a(1 - 0)}{(1 - r)}$$

$$= \frac{a}{(1 - r)} = c, \text{ where } c \text{ is a constant}$$

Sum of Geometric Progression (cont)

$$\sum_{i=0}^{n-1} ar^i = \frac{a(1 - r^n)}{(1 - r)} = \frac{-1}{-1}\,\frac{a(1 - r^n)}{(1 - r)} = \frac{a(r^n - 1)}{(r - 1)}$$

Graphical intuition of being bounded by a constant!

Special case when $|r| < 1$: As $n \dashrightarrow \infty$, $r^n \dashrightarrow 0$

$$\sum_{i=0}^{n-1} ar^i < \sum_{i=0}^{\infty} ar^i$$

$$= \frac{a(1 - 0)}{(1 - r)}$$

$$= \frac{a}{(1 - r)} = c\text{, where } c \text{ is a constant}$$

Exercise (AP/GP)

- Simplify $1 + 2 + 4 + 8 + \ldots + n$
- Simplify $1 + 2 + 4 + 8 + \ldots + \log_2(n)$

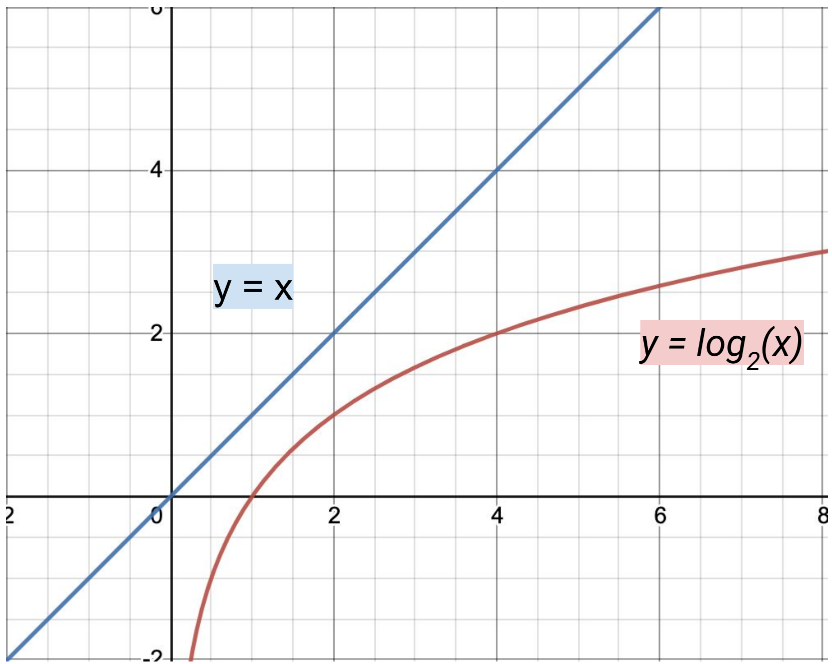
Logarithm

- The inverse function to **exponentiation**

$$2^x = 16$$
$$x = \log_2(16)$$

Logarithm Graph

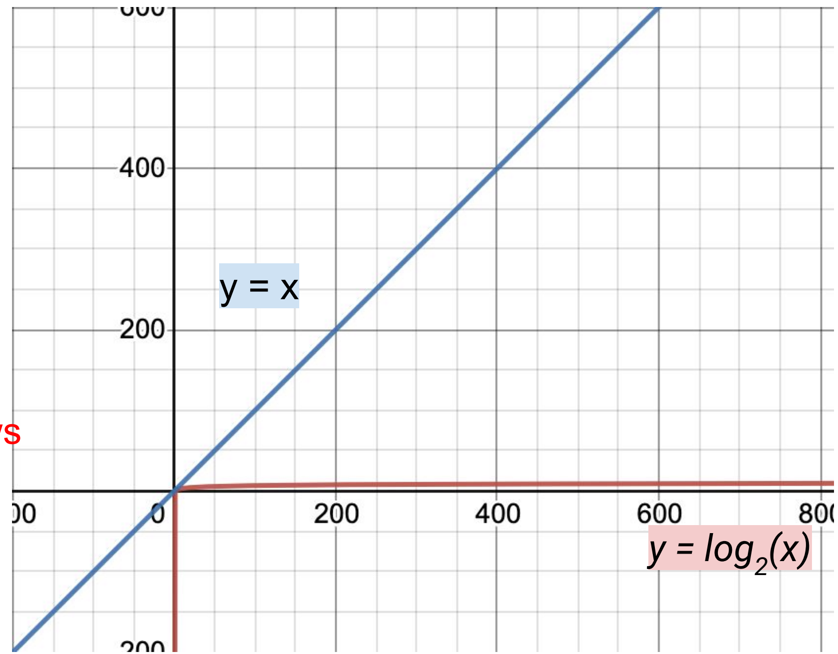
- Comparing $y = x$ and $y = log_2x$



Logarithm Graph

- Comparing $y = x$ and $y = log_2x$

Logarithm grows
very slowly!



Logarithm EZ rules you hopefully already know

- $log_a(xy) = log_a(x) + log_a(y)$
- $log_a(x / y) = log_a(x) - log_a(y)$

Logarithm EZ rules you hopefully already know

- $\log_a(xy) = \log_a(x) + \log_a(y)$
- $\log_a(x / y) = \log_a(x) - \log_a(y)$
- $\log_a(x^n) = n \log_a(x)$

Logarithm EZ rules you hopefully already know

- $log_a(xy) = log_a(x) + log_a(y)$
- $log_a(x / y) = log_a(x) - log_a(y)$
- $log_a(x^n) = n\ log_a(x)$
- $log_a(1) = 0$
- $log_a(a) = 1$

Slightly trickier which I didn’t really know (prior to university)

- $a^{\log_a(n)} = n$

Slightly trickier which I didn't really know (prior to university)

- $a^{log_a(n)} = n$

Intuition: logarithms and exponentials are **inverses** of each other.

Slightly trickier which I didn't really know (prior to university)

- $a^{\log_a(n)} = n$

Intuition: logarithms and exponentials are **inverses** of each other.

So if you exponentiate a logarithm, it cancels off.

Slightly trickier which I didn’t really know (prior to university)

- $a^{\log_a(n)} = n$

Intuition: logarithms and exponentials are **inverses** of each other.

So if you exponentiate a logarithm, it cancels off.

Related: logarithm-fy an exponential

- $\log_a(a^n) = n \log_a(a) = n\ (1) = n$

Slightly trickier which I didn't really know (prior to university)

- $a^{\log_b(c)} = c^{\log_b(a)}$

Slightly trickier which I didn't really know (prior to university)

- $a^{\log_b(c)} = c^{\log_b(a)}$ (Prove it!)

Slightly trickier which I didn't really know (prior to university)

- $a^{log_b(c)} = c^{log_b(a)}$ (Prove it!)

$$x = a^{\log_b(c)}$$

Slightly trickier which I didn't really know (prior to university)

- $a^{log_b(c)} = c^{log_b(a)}$ (Prove it!)

$$x = a^{\log_b(c)}$$

$$\log_b(x) = \log_b(a^{\log_b(c)})$$

Take log base b in both sides

Slightly trickier which I didn't really know (prior to university)

- $a^{\log_b(c)} = c^{\log_b(a)}$ (Prove it!)

$$x = a^{\log_b(c)}$$

$$\log_b(x) = \log_b(a^{\boxed{\log_b(c)}})$$

$$\log_b(x) = \boxed{\log_b(c)} \times \log_b(a)$$

$[\log_a(x^n) = n \log_a(x)]$

Slightly trickier which I didn't really know (prior to university)

- $a^{\log_b(c)} = c^{\log_b(a)}$ (Prove it!)

$$x = a^{\log_b(c)}$$

$$\log_b(x) = \log_b(a^{\log_b(c)})$$

$$\log_b(x) = \log_b(c) \times \boxed{\log_b(a)}$$

$$\log_b(x) = \log_b(c^{\boxed{\log_b(a)}})$$

$[\log_a(x^n) = n \log_a(x)]$

Slightly trickier which I didn't really know (prior to university)

- $a^{log_b(c)} = c^{log_b(a)}$ (Prove it!)

$$x = a^{\log_b(c)}$$

$$\log_b(x) = \log_b(a^{\log_b(c)})$$

$$\log_b(x) = \log_b(c) \times \log_b(a)$$

$$\log_b(\boxed{x}) = \log_b(\boxed{c^{\log_b(a)}})$$

$$\boxed{x} = \boxed{c^{\log_b(a)}}$$

Logarithm Change of Base

$$\log_{\boxed{b}} \boxed{a} = \frac{\log_{\boxed{d}} \boxed{a}}{\log_{\boxed{d}} \boxed{b}}$$

Logarithm Rules and Tricks summary

1. $log_a(xy) = log_a(x) + log_a(y)$
2. $log_a(x / y) = log_a(x) - log_a(y)$
3. $log_a(x^n) = n\ log_a(x)$
4. $log_a(1) = 0$
5. $log_a(a) = 1$
6. $a^{log_a(n)} = n$
7. $a^{log_b(c)} = c^{log_b(a)}$
8. $log_b(a) = log_d(a) / log_d(b)$

$lg(n!) = O(nlgn)$

$$\lg(n!)$$

lg(n!) = O(nlgn)

Expanding factorial

$$\lg(n!) = \lg(n \times (n-1) \times \ldots \times 2 \times 1)$$

lg(n!) = O(nlgn)

$$\begin{aligned}\lg(n!) &= \lg(n \times (n-1) \times \ldots \times 2 \times 1) \\ &= \lg(n) + \lg(n-1) + \ldots + \lg(2) + \lg(1)\end{aligned}$$

lg(n!) = O(nlgn)

$$\begin{aligned}\lg(n!) &= \lg(n \times (n-1) \times \ldots \times 2 \times 1) \\ &= \lg(n) + \lg(n-1) + \ldots + \lg(2) + \lg(1)\end{aligned}$$

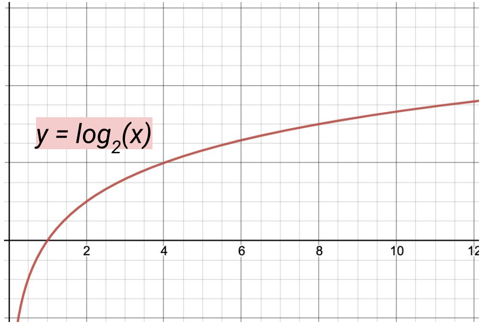
Repeated applications of:
lg(ab) = lg(a) + lg(b)

lg(n!) = O(nlgn)

$$\begin{aligned}
\lg(n!) &= \lg(n \times (n-1) \times \ldots \times 2 \times 1) \\
&= \lg(n) + \lg(n-1) + \ldots + \lg(2) + \lg(1) \\
&\leq \lg(n) + \lg(n) + \ldots + \lg(n) + \lg(n)
\end{aligned}$$

lg(n!) = O(nlgn)

$$\begin{aligned}\lg(n!) &= \lg(n \times (n-1) \times \ldots \times 2 \times 1) \\ &= \lg(n) + \lg(n-1) + \ldots + \lg(2) + \lg(1) \\ &\leq \lg(n) + \lg(n) + \ldots + \lg(n) + \lg(n)\end{aligned}$$



We have:

lg(n-1) <= lg(n)

lg(n-2) <= lg(n)

...

lg(2) <= lg(n)

lg(1) <= lg(n)

lg(n!) = O(nlgn)

$$\begin{aligned}\lg(n!) &= \lg(n \times (n-1) \times \ldots \times 2 \times 1) \\ &= \lg(n) + \lg(n-1) + \ldots + \lg(2) + \lg(1) \\ &\leq \underbrace{\lg(n) + \lg(n) + \ldots + \lg(n) + \lg(n)}_{n \text{ terms}}\end{aligned}$$

$lg(n!) = O(nlgn)$

$$
\begin{aligned}
\lg(n!) &= \lg(n \times (n-1) \times \ldots \times 2 \times 1) \\
&= \lg(n) + \lg(n-1) + \ldots + \lg(2) + \lg(1) \\
&\leq \underbrace{\lg(n) + \lg(n) + \ldots + \lg(n) + \lg(n)}_{n \text{ terms}} \\
&\leq n \lg(n)
\end{aligned}
$$

Exercise (Logarithm)

1. Simplify the $log_2(n)^{th}$ root of n: $\sqrt[\log_2(n)]{n}$

2. Prove that $lg(n!) = \Omega(nlgn)$

Binary to Decimals

If you have *n* bits, you can represent integers in decimals in the range of $[0..2^n-1]$.

Binary to Decimals

If you have *n* bits, you can represent integers in decimals in the range of $[0..2^n-1]$.

(Example) With 4 bits:

0000: 0	1000: 8
0001: 1	1001: 9
0010: 2	1010: 10
0011: 3	1011: 11
0100: 4	1100: 12
0101: 5	1101: 13
0110: 6	1110: 14
0111: 7	1111: 15

$2^4 - 1$

Decimals to Binary

An integer $x >= 1$, needs $n = \lfloor \log_2(x) \rfloor + 1$ bits to represent it

Decimals to Binary

An integer $x >= 1$, needs $n = \lfloor log_2(x) \rfloor + 1$ bits to represent it

x	*Binary Repr.*	*n*
1	*1*	*1*
2	*10*	*2*
3	*11*	*2*
4	*100*	*3*
5	*101*	*3*
10	*1010*	*4*
23	*10111*	*5*
63	*111111*	*6*
64	*1000000*	*7*

Decimals to Binary

$O(log(x))$ bits to represent!

An integer $x >= 1$, needs $n = \lfloor log_2(x) \rfloor + 1$ bits to represent it

x	*Binary Repr.*	*n*
1	1	1
2	10	2
3	11	2
4	100	3
5	101	3
10	1010	4
23	10111	5
63	111111	6
64	1000000	7

Matrix

Let A be a matrix of size $m \times n$, and a_{ij} represent the matrix along the i^{th} row and j^{th} column (using 1-indexing). We also call it the (i, j)-entry of the matrix

Matrix

Let A be a matrix of size m x n, and a_{ij} represent the matrix along the i^{th} row and j^{th} column (using 1-indexing). We also call it the *(i, j)*-entry of the matrix

Example:

This matrix is of size *3x4*

$$
\begin{array}{c|cccc|}
 & 1 & 2 & 3 & 4 \\
\hline
1 & 0 & 23 & 4 & 7 \\
2 & 1 & 0 & \boxed{18} & 1 \\
3 & 6 & 15 & 0 & 8 \\
\hline
\end{array}
$$

The (2, 3)-entry of the matrix is 18

Matrix Transpose

The transpose of matrix $B = A^T$ is such that (i, j)-entry of B is the (j, i)-entry of A

Example:

	1	2	3	4
1	0	23	4	7
2	1	0	18	1
3	6	15	0	8

$\xrightarrow{\text{Transpose}}$

	1	2	3
1	0	1	6
2	23	0	15
3	4	18	0
4	7	1	8

Special Matrices (Part 1)

Note: they have to be **square** -- they are *n x n* matrices!

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Identity Matrix

$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Diagonal Matrix

$$\begin{bmatrix} 0 & 2 & 3 & 4 \\ 2 & 5 & 3 & 5 \\ 3 & 3 & 1 & 0 \\ 4 & 5 & 0 & 1 \end{bmatrix}$$

Symmetric Matrix

Special Matrices (Part 2)

Note: they have to be **square** -- they are *n x n* matrices!

3	2	0	4
0	2	5	2
0	0	3	3
0	0	0	0

Upper Triangular Matrix

3	0	0	0
5	2	0	0
8	9	3	0
30	20	0	0

Lower Triangular Matrix

Matrix Multiplication

If you have a Matrix *A* with size $m \times n$ and Matrix *B* with size $n \times p$, you can multiply them to produce Matrix *C* with size $m \times p$

Matrix Multiplication

If you have a Matrix *A* with size *m* x *n* and Matrix *B* with size *n* x *p*, you can multiply them to produce Matrix *C* with size *m* x *p*

In other words, the number of **columns of *A*** has to be equal number of **rows of *B***

Matrix Multiplication

If you have a Matrix *A* with size *m* x *n* and Matrix *B* with size *n* x *p*, you can multiply them to produce Matrix *C* with size *m* x *p*

In other words, the number of **columns of *A*** has to be equal number of **rows of *B***

Computing c_{ij} :

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Matrix Multiplication (example)

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$\overset{3 \times 4}{\begin{bmatrix} 0 & 23 & 4 & 7 \\ 1 & 0 & 18 & 1 \\ 15 & 6 & 0 & 8 \end{bmatrix}} \times \overset{4 \times 2}{\begin{bmatrix} 0 & 2 \\ 2 & 1 \\ 3 & 3 \\ 4 & 5 \end{bmatrix}} = \overset{3 \times 2}{\begin{bmatrix} & \\ & \\ & \end{bmatrix}}$$

Matrix Multiplication (example)

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$\underset{3 \times 4}{\begin{bmatrix} 0 & 23 & 4 & 7 \\ 1 & 0 & 18 & 1 \\ 15 & 6 & 0 & 8 \end{bmatrix}} \times \underset{4 \times 2}{\begin{bmatrix} 0 & 2 \\ 2 & 1 \\ 3 & 3 \\ 4 & 5 \end{bmatrix}} = \underset{3 \times 2}{\begin{bmatrix} 86 & \\ & \\ & \end{bmatrix}}$$

Matrix Multiplication (example)

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$\overset{3 \times 4}{\begin{bmatrix} 0 & 23 & 4 & 7 \\ 1 & 0 & 18 & 1 \\ 15 & 6 & 0 & 8 \end{bmatrix}} \times \overset{4 \times 2}{\begin{bmatrix} 0 & 2 \\ 2 & 1 \\ 3 & 3 \\ 4 & 5 \end{bmatrix}} = \overset{3 \times 2}{\begin{bmatrix} 86 & 70 \\ & \\ & \end{bmatrix}}$$

Matrix Multiplication (example)

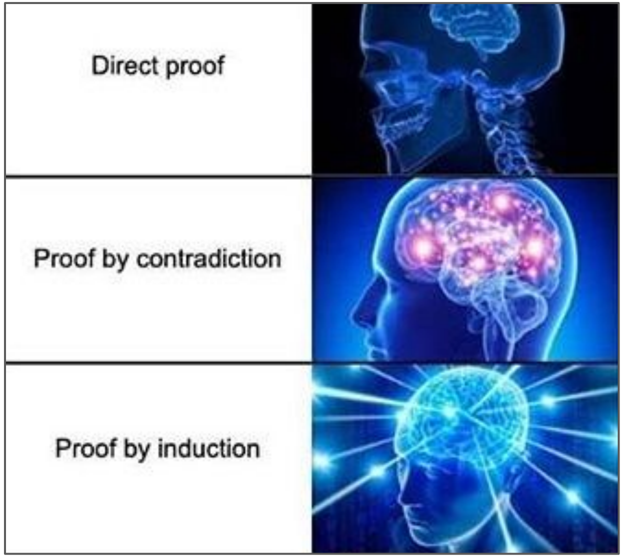
$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

$$\underset{3 \times 4}{\begin{bmatrix} 0 & 23 & 4 & 7 \\ 1 & 0 & 18 & 1 \\ 15 & 6 & 0 & 8 \end{bmatrix}} \times \underset{4 \times 2}{\begin{bmatrix} 0 & 2 \\ 2 & 1 \\ 3 & 3 \\ 4 & 5 \end{bmatrix}} = \underset{3 \times 2}{\begin{bmatrix} 86 & 70 \\ 58 & \\ & \end{bmatrix}}$$

Exercise (Matrix)

1. Given two **square** matrices (of size *n*) *A* and *B*. What's the time complexity of Matrix Multiplication (that was just described)?

2. What if they are not square? i.e. *A* is size *m x n* and *B* is size *n x p*


Direct proof
Proof by contradiction
Proof by induction

Proofs


"The proof is trivial and intuitive"
"The proof is too long and arduous to fit within this margin"
"The proof is left as an exercise to the reader"
The proof is by magic.

Proof by Contradiction

Idea:

Proof by Contradiction

Idea:

1. *Assume* what we want to prove is ***not*** **true**.

Proof by Contradiction

Idea:

1. *Assume* what we want to prove is ***not*** **true**.
2. Using this, we show that the **consequences is not possible**
   a. Either contradicting what we assumed or
   b. Some other fact we already know to be true
   c. (or both)

Proof by Contradiction

Idea:

1. *Assume* what we want to prove is ***not*** **true**.
2. Using this, we show that the **consequences is not possible**
    a. Either contradicting what we assumed or
    b. Some other fact we already know to be true
    c. (or both)
3. Thus, **what we *assumed*** must be **incorrect**. So we can conclude that what we want to prove is true

Proof by Contradiction

Idea:

Proof by Contradiction (Example)

To prove: If p^2 is even, then p is even

Proof (by contradiction):

Proof by Contradiction (Example)

To prove: If p^2 is even, then p is even

Proof (by contradiction):

1. Assume that p is actually odd (can be expressed in the form of $2n + 1$)
   a. What we know currently is two things:
      i. p^2 is even (what we started off with)
      ii. p is odd (by assumption) and

Proof by Contradiction (Example)

To prove: If p^2 is even, then p is even

Proof (by contradiction):

1. Assume that p is actually odd (can be expressed in the form of $2n + 1$)
   a. What we know currently is two things:
      i. p^2 is even (what we started off with)
      ii. p is odd (by assumption) and
2. Expand $p^2 = (2n + 1)^2 = 4n^2 + 4n + 1$

Proof by Contradiction (Example)

To prove: If p^2 is even, then p is even

Proof (by contradiction):

1. Assume that p is actually odd (can be expressed in the form of $2n + 1$)
    a. What we know currently is two things:
        i. p^2 is even (what we started off with)
        ii. p is odd (by assumption) and
2. Expand $p^2 = (2n + 1)^2 = 4n^2 + 4n + 1$ (Form of $2k + 1$)
3. But $p^2 = 4n^2 + 4n + 1$ can be rewritten as $2(2n^2 + 2n) + 1$

Proof by Contradiction (Example)

To prove: If p^2 is even, then p is even

Proof (by contradiction):

1. Assume that p is actually odd (can be expressed in the form of $2n + 1$)
   a. What we know currently is two things:
      i. p^2 is even (what we started off with)
      ii. p is odd (by assumption) and
2. Expand $p^2 = (2n + 1)^2 = 4n^2 + 4n + 1$ (Form of $2k + 1$)
3. But $p^2 = 4n^2 + 4n + 1$ can be rewritten as $2(2n^2 + 2n) + 1$
4. We deduce that p^2 is odd

Proof by Contradiction (Example)

To prove: If p^2 is even, then p is even

Proof (by contradiction):

1. Assume that p is actually odd (can be expressed in the form of $2n + 1$)
   a. What we know currently is two things:
      i. p^2 is even (what we started off with)
      ii. p is odd (by assumption) and
2. Expand $p^2 = (2n + 1)^2 = 4n^2 + 4n + 1$
3. But $p^2 = 4n^2 + 4n + 1$ can be rewritten as $2(2n^2 + 2n) + 1$
4. We deduce that p^2 is odd
5. But this contradicts the fact that p^2 is supposed to be even!

Note on Proof by Contradiction

The previous example can also count as Proof by Contraposition. Please see the following:

> Thank you for the question. Yes, you are right that in this case it **can** count as a Proof by Contraposition as well. I must admit that when I designed the slides, I could not find beginner-friendly examples for Proof by Contradiction that doesn't rely on implications (statements in the form of P -> Q).
>
> In general, a "genuine" Proof by Contradiction would be such that you derive some contradiction that might not even be related. To give a taste of an idea (which again I apologise i do not have an example at hand without going through some other materials), let's say you have:
>
> - assumed the statement X to not be true.
> - Then you make some logical deductions at every step using the assumption that "X is not true"
> - and suddenly derive that "1 = 0" (which is clearly not true!).

Note on Proof by Contradiction (cont'd)

For further reading, you might want to visit this link as well https://math.stackexchange.com/questions/262828/using-proof-by-contradiction-vs-proof-of-the-contrapositive

One of the answers made an observation that contradicting the premise is a proof by contrapositive:

84

> To prove $P \to Q$, you can do the following:
>
> 1. Prove directly, that is assume P and show Q;
> 2. Prove by contradiction, that is assume P and $\neg Q$ and derive a contradiction; or
> 3. Prove the contrapositive, that is assume $\neg Q$ and show $\neg P$.
>
> Sometimes the contradiction one arrives at in (2) is merely contradicting the assumed premise P, and hence, as you note, is essentially a proof by contrapositive (3). However, note that (3) allows us to assume *only* $\neg Q$; if we can then derive $\neg P$, we have a *clean* proof by contrapositive.

Proof by Induction

Proof by Induction

Useful for proving truth of a statement *P(n)* for **all positive integers** *n*

Idea:

Proof by Induction

Useful for proving truth of a statement *P(n)* for **all positive integers** *n*

Idea:

1. **Base Case**: Show that the base case is true (usually checking *n = 0 or 1*, etc)

Proof by Induction

Useful for proving truth of a statement *P(n)* for **all positive integers** *n*

Idea:

1. **Base Case**: Show that the base case is true (usually checking *n = 0 or 1*, etc)
2. **Inductive Step:** Assume *P(x)* is true. Using *P(x),* show that *P(x + 1)* is true

Proof by Induction

Useful for proving truth of a statement *P(n)* for **all positive integers** *n*

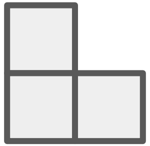
Idea:

1. **Base Case**: Show that the base case is true (usually checking *n = 0 or 1*, etc)
2. **Inductive Step:** Assume *P(x)* is true. Using *P(x)*, show that *P(x + 1)* is true

Proof by Induction (Example)

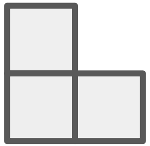
Proof by Induction (Example)

We say that a space is **“L-tileable”** if we can place multiple non-overlapping copies of an L shaped tile (can be flipped or rotated) to cover the entire space

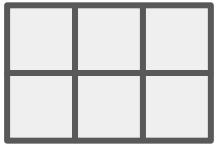


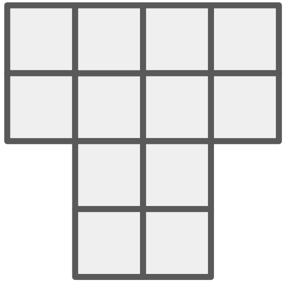
Proof by Induction (Example)

We say that a space is **"L-tileable"** if we can place multiple non-overlapping copies of an L shaped tile (can be flipped or rotated) to cover the entire space



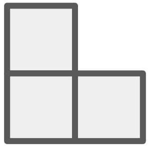
Examples:



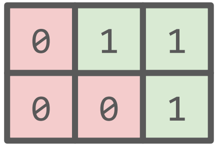


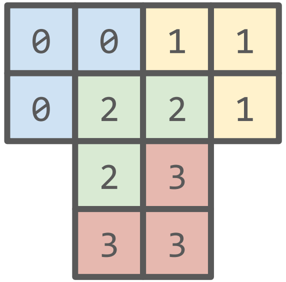
Proof by Induction (Example)

We say that a space is **“L-tileable”** if we can place multiple non-overlapping copies of an L shaped tile (can be flipped or rotated) to cover the entire space



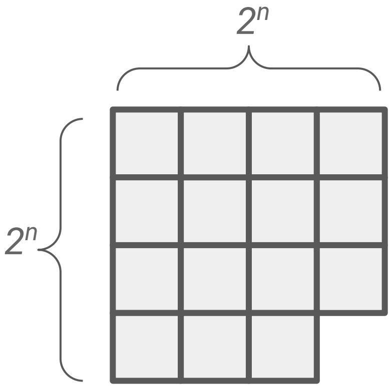
Examples:



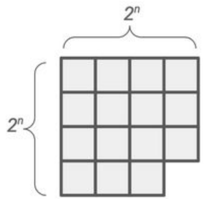


Proof by Induction (Example)

Show that the 2^n x 2^n space, with the bottom right 1 x 1 corner **removed** is L-tileable for all positive integers n



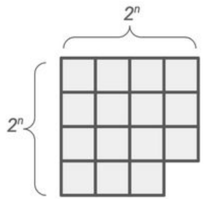
Show that the 2^n x 2^n space, with the bottom right 1 x 1 corner **removed** is L-tileable for all positive integers n



Let *P(k)* be the statement: the 2^k x 2^k space, with the bottom right 1 x 1 corner **removed** is L-tileable

Proof (by induction on *n*):

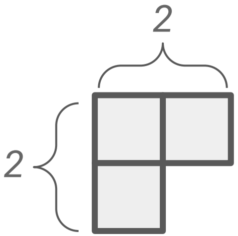
Show that the 2^n x 2^n space, with the bottom right 1 x 1 corner **removed** is L-tileable for all positive integers n



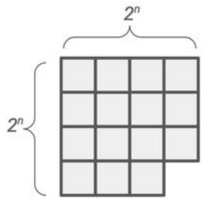
Let *P(k)* be the statement: the 2^k x 2^k space, with the bottom right 1 x 1 corner **removed** is L-tileable

Proof (by induction on *n*):

Base Case (*k = 1*):
$2^n = 2^1 = 2$



Show that the 2^n x 2^n space, with the bottom right 1 x 1 corner **removed** is L-tileable for all positive integers n

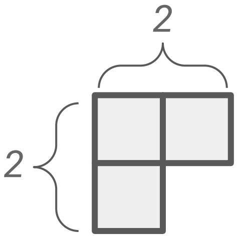


Let *P(k)* be the statement: the 2^k x 2^k space, with the bottom right 1 x 1 corner **removed** is L-tileable

Proof (by induction on *n*):

Base Case ($k = 1$):

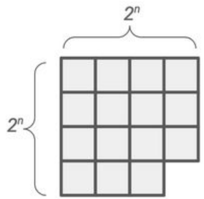
$2^n = 2^1 = 2$



Can simply rotate the base tile!

Thus *P(1)* is true

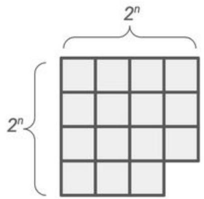
Show that the 2^n x 2^n space, with the bottom right 1 x 1 corner **removed** is L-tileable for all positive integers n



Let *P(k)* be the statement: the 2^k x 2^k space, with the bottom right 1 x 1 corner **removed** is L-tileable

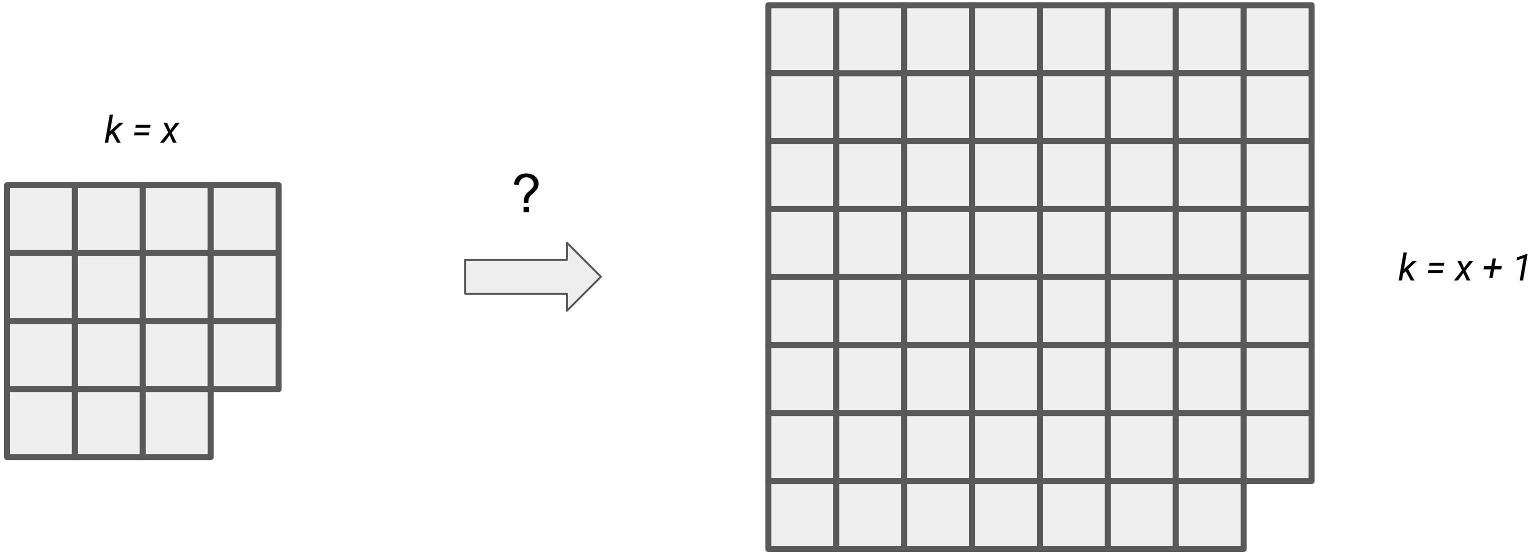
Inductive Step: Assume *P(x)* is true. To show that *P(x + 1)* is true

Show that the 2^n x 2^n space, with the bottom right 1 x 1 corner **removed** is L-tileable for all positive integers n



Let *P(k)* be the statement: the 2^k x 2^k space, with the bottom right 1 x 1 corner **removed** is L-tileable

Inductive Step: Assume *P(x)* is true. To show that *P(x + 1)* is true



Idea

Make 4 copies of the L-tileable space that satisfies P(x). Lay it in this manner (after some rotation)

Idea

Make 4 copies of the L-tileable space that satisfies P(x). Lay it in this manner (after some rotation)

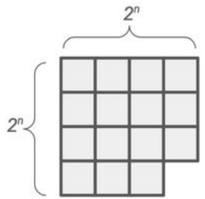
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0			1	1	1
2	2	2		3	3	3	3
2	2	2	2	3	3	3	3
2	2	2	2	3	3	3	3
2	2	2	2	3	3	3	

Idea

Insert the base tile!

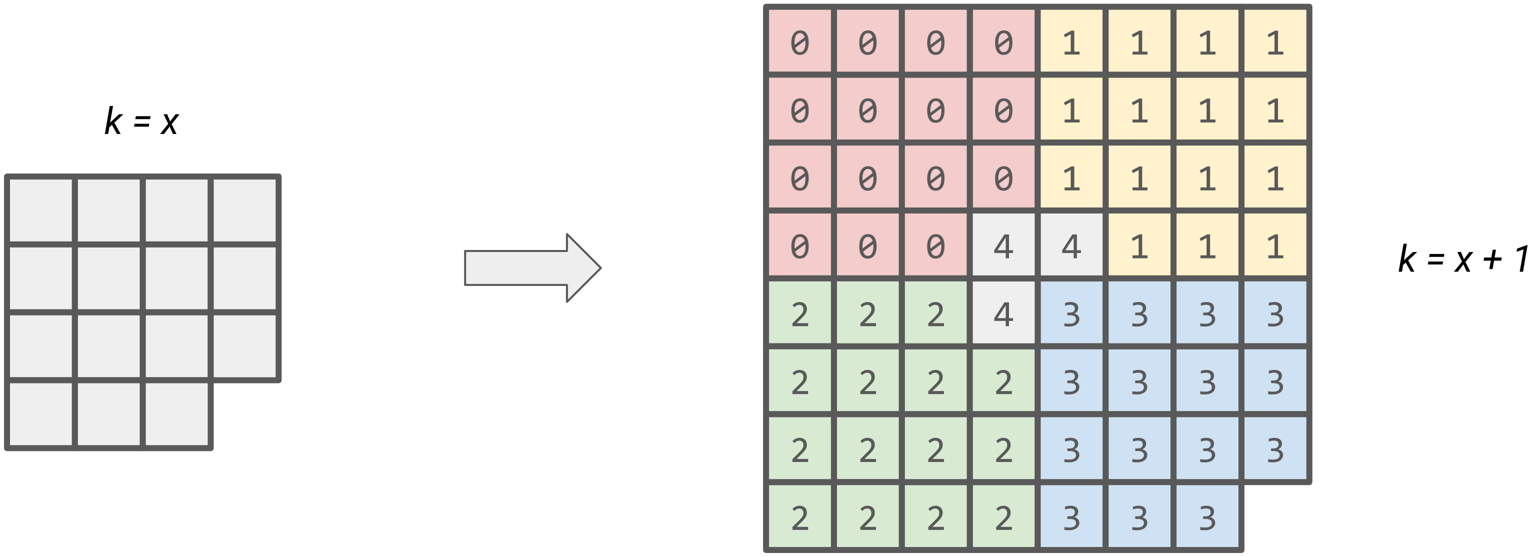
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	4	4	1	1	1
2	2	2	4	3	3	3	3
2	2	2	2	3	3	3	3
2	2	2	2	3	3	3	3
2	2	2	2	3	3	3	

Show that the 2^n x 2^n space, with the bottom right 1 x 1 corner **removed** is L-tileable for all positive integers n

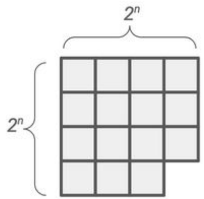


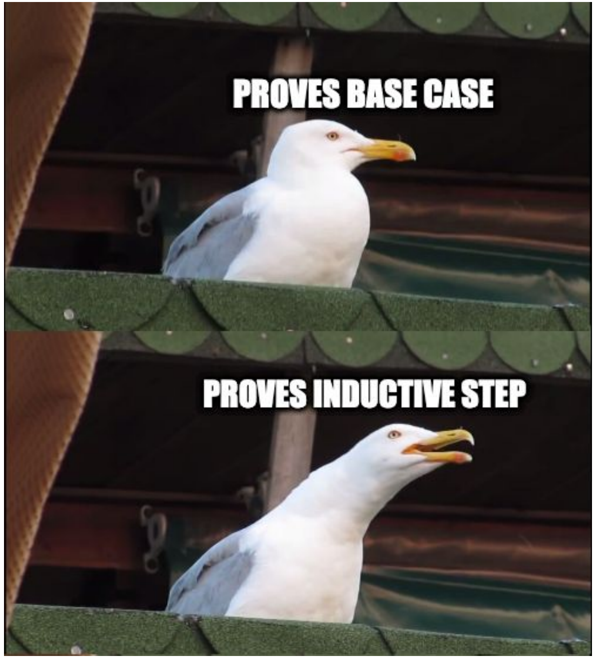
Let *P(k)* be the statement: the 2^k x 2^k space, with the bottom right 1 x 1 corner **removed** is L-tileable

Inductive Step: Assume *P(x)* is true. To show that *P(x + 1)* is true (Proven!)

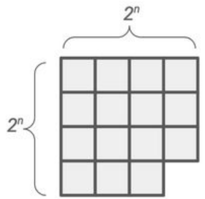


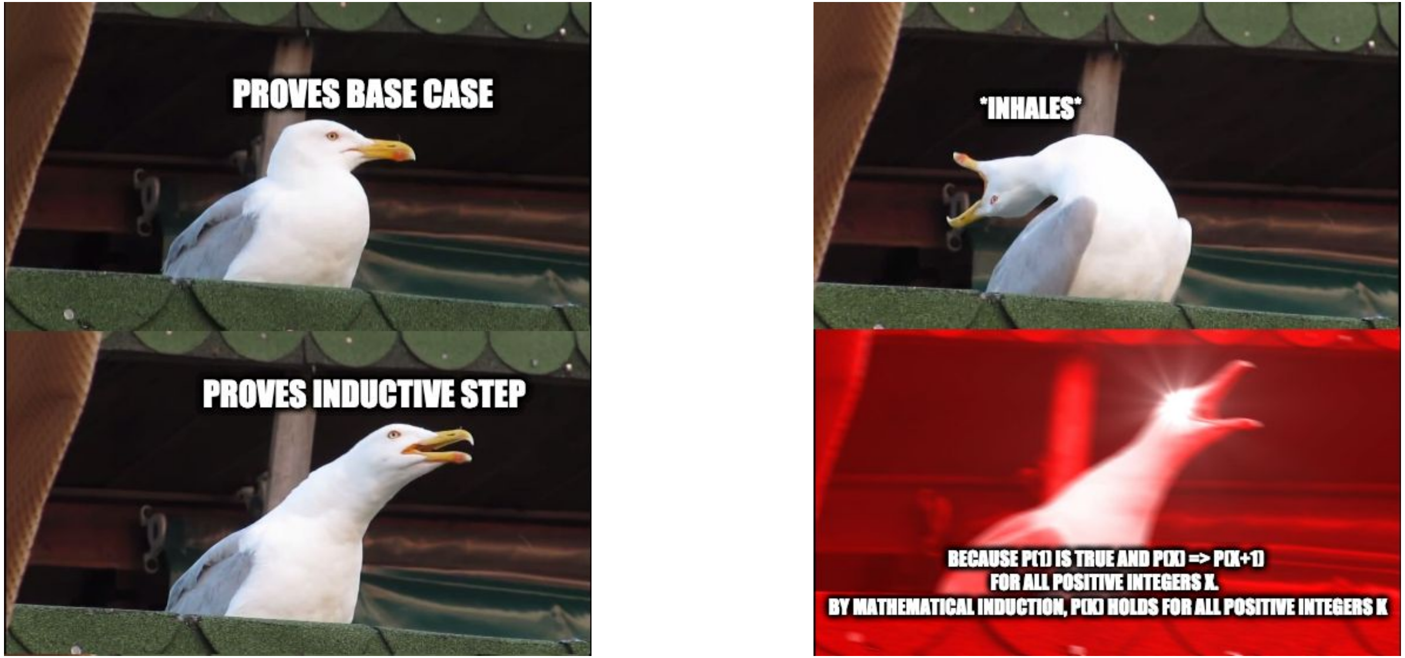
Show that the $2^n \times 2^n$ space, with the bottom right 1 x 1 corner **removed** is L-tileable for all positive integers n





Show that the 2^n x 2^n space, with the bottom right 1 x 1 corner **removed** is L-tileable for all positive integers n





Proof by Strong Induction

Variant of induction!

Idea:

Proof by Strong Induction

Variant of induction!

Idea:

1. **Base Case**: Show that the base case is true (usually checking *n = 0 or 1*, etc. Can be multiple base cases!)
2. **Inductive Step:** Assume ALL OF *P(0), P(1), P(2), … P(x)* is true. Show that *P(x + 1)* is true

Proof by Strong Induction

Variant of induction!

Idea:

1. **Base Case**: Show that the base case is true (usually checking *n = 0 or 1*, etc. Can be multiple base cases!)
2. **Inductive Step:** Assume ALL OF *P(0), P(1), P(2), … P(x)* is true. Show that *P(x + 1)* is true

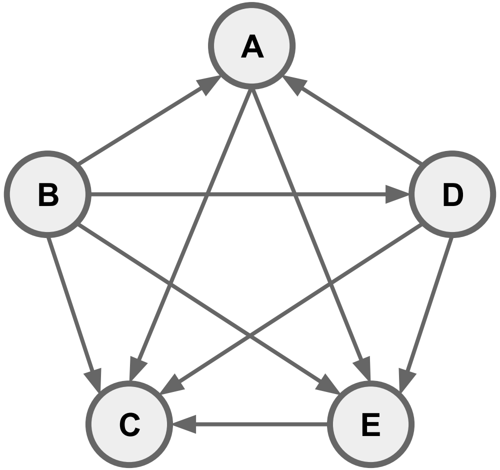
By right, should start from the base case.
Here I just assume that 0 is the base case

Proof by Strong Induction (example)

A country has n cities. Any two cities are connected by a **one-way road**. Show that there is a route that passes through every city (you can start and end anywhere)

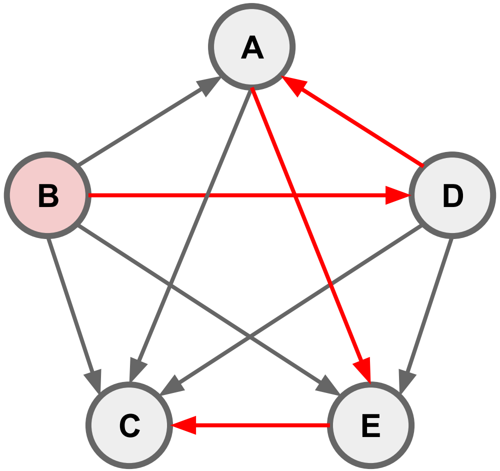
Proof by Strong Induction (example)

A country has n cities. Any two cities are connected by a **one-way road**. Show that there is a route that passes through every city (you can start and end anywhere)

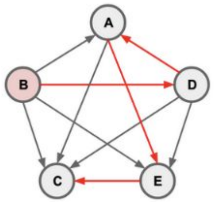


Proof by Strong Induction (example)

A country has n cities. Any two cities are connected by a **one-way road**. Show that there is a route that passes through every city (you can start and end anywhere)



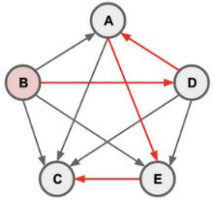
A country has n cities. Any two cities are connected by a **one-way road**. Show that there is a route that passes through every city (you can start and end anywhere)



Let *P(k)* be the statement: given the country of k cities connected by one-way road, there is a route that passes through every city

Proof (by strong induction on *n*):

A country has n cities. Any two cities are connected by a **one-way road**. Show that there is a route that passes through every city (you can start and end anywhere)

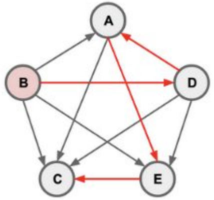


Let *P(k)* be the statement: given the country of k cities connected by one-way road, there is a route that passes through every city

Proof (by strong induction on *n*):

Base Case (k = 1):

A country has n cities. Any two cities are connected by a **one-way road**. Show that there is a route that passes through every city (you can start and end anywhere)



Let *P(k)* be the statement: given the country of k cities connected by one-way road, there is a route that passes through every city

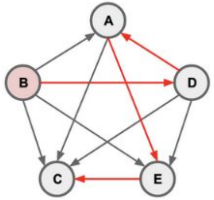
Proof (by strong induction on *n*):

Base Case (k = 1):

Trivial! Just one city. The route is the city itself

Thus *P(1)* is true

A country has n cities. Any two cities are connected by a **one-way road**. Show that there is a route that passes through every city (you can start and end anywhere)



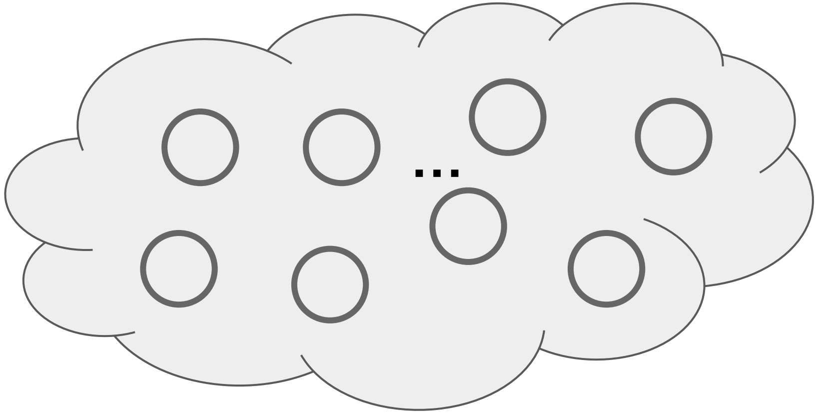
Let *P(k)* be the statement: given the country of k cities connected by one-way road, there is a route that passes through every city

Inductive Step: Assume *P(i)* is true for *1 <= i <= x*. To show that *P(x + 1)* is true

Idea

Start off with country with x + 1 cities

Inductive Step: Assume *P(i)* is true for *1 <= i <= x*. To show that *P(x + 1)* is true

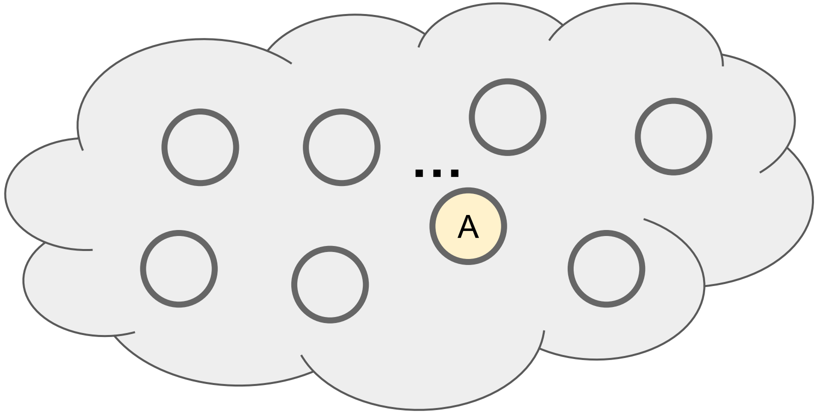


x + 1 cities

Idea

`Choose an *arbitrary* city. Call it city A`

Inductive Step: Assume *P(i)* is true for *1 <= i <= x*. To show that *P(x + 1)* is true



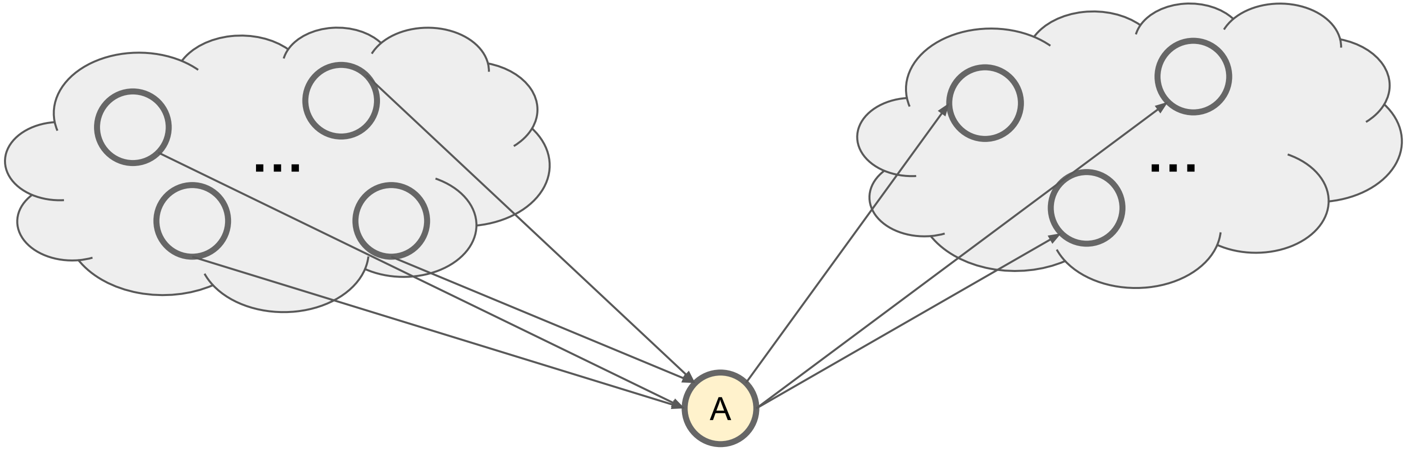
x + 1 cities

Idea

Separate the rest of the cities into two sets:

- Set of cities that lead to A
- Set of cities that A can go to

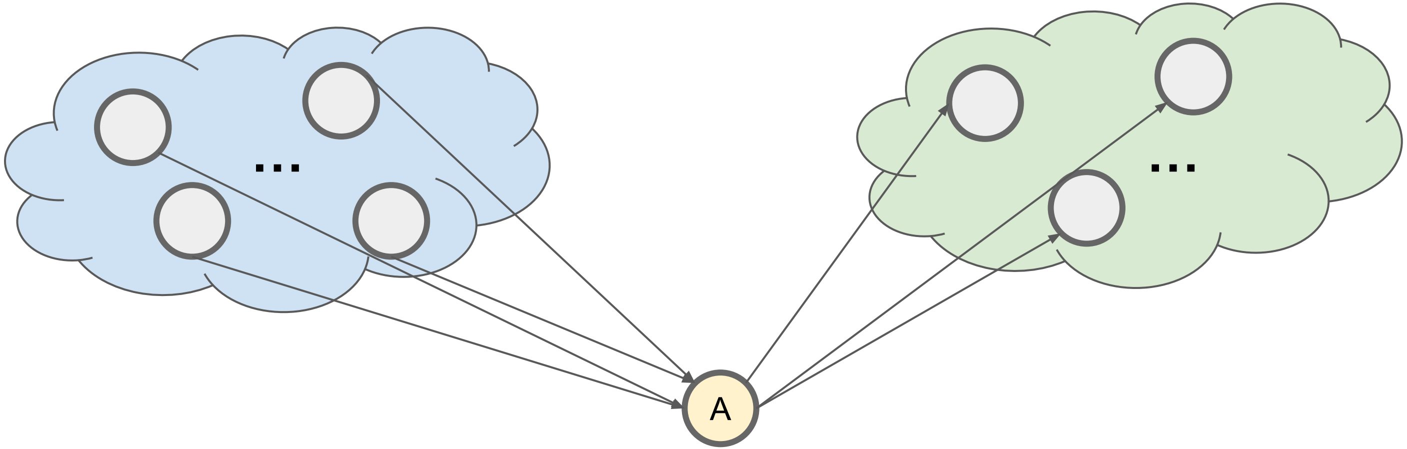
Inductive Step: Assume *P(i)* is true for *1 <= i <= x*. To show that *P(x + 1)* is true



Idea

```
Do we know the exact size of the two sets?
```

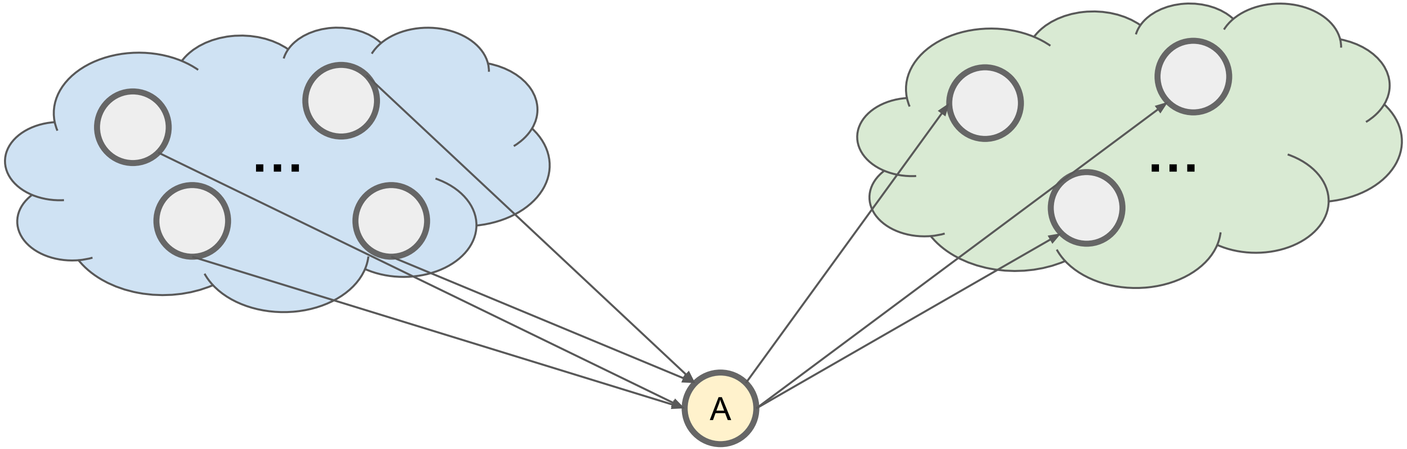
Inductive Step: Assume *P(i)* is true for *1 <= i <= x*. To show that *P(x + 1)* is true



Idea

Do we know the exact size of the two sets?
Not really! But we know that they are less than x + 1 (because we already removed city A)

Inductive Step: Assume *P(i)* is true for *1 <= i <= x*. To show that *P(x + 1)* is true

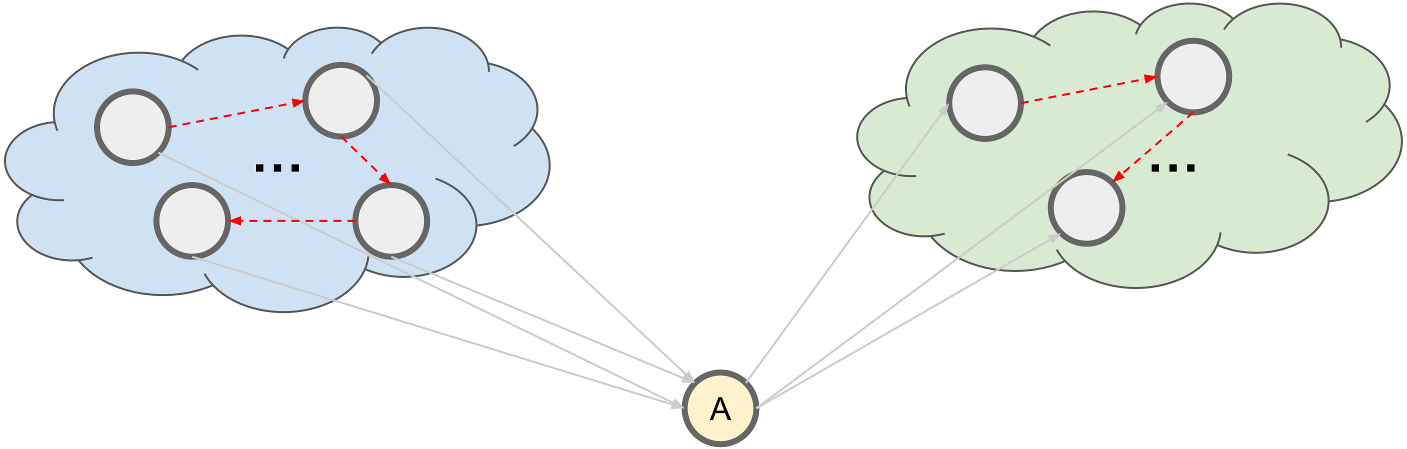


Idea

Invoke the Inductive Hypothesis for both* sets! We know that both of them have some path

**there is a technical detail that we should not consider empty set because it is not covered by the inductive hypothesis. But let's focus on the main idea for this set of slides*

Inductive Step: Assume $P(i)$ is true for $1 <= i <= x$. To show that $P(x + 1)$ is true

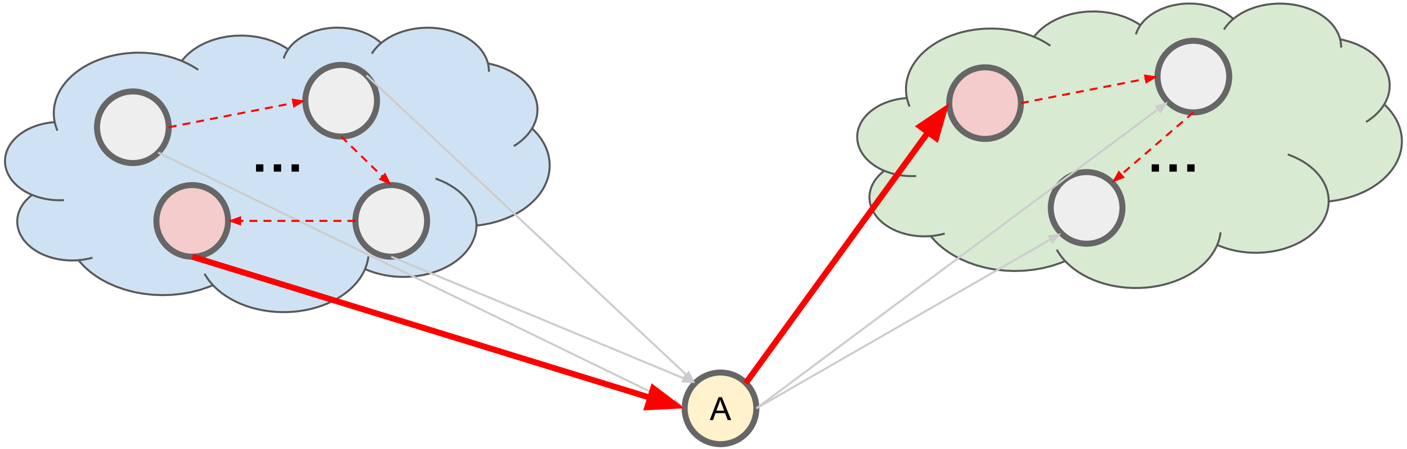


Idea

Take the last city in the set of cities that lead to A and the first city in the set of cities that A can go to

Connect A with those cities

Inductive Step: Assume *P(i)* is true for *1 <= i <= x*. To show that *P(x + 1)* is true

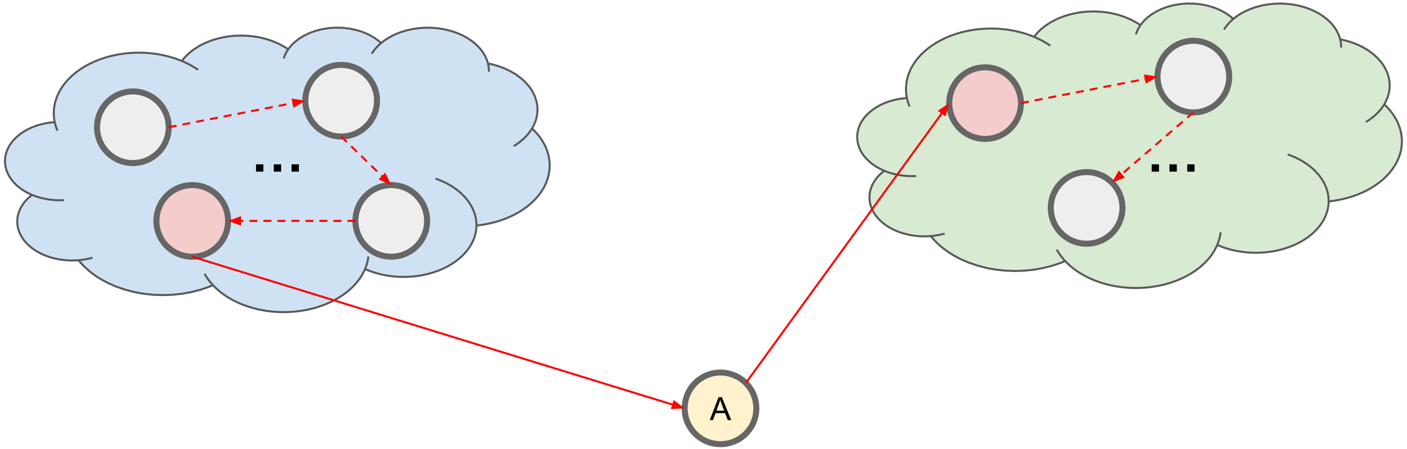


Idea

And we are done*!

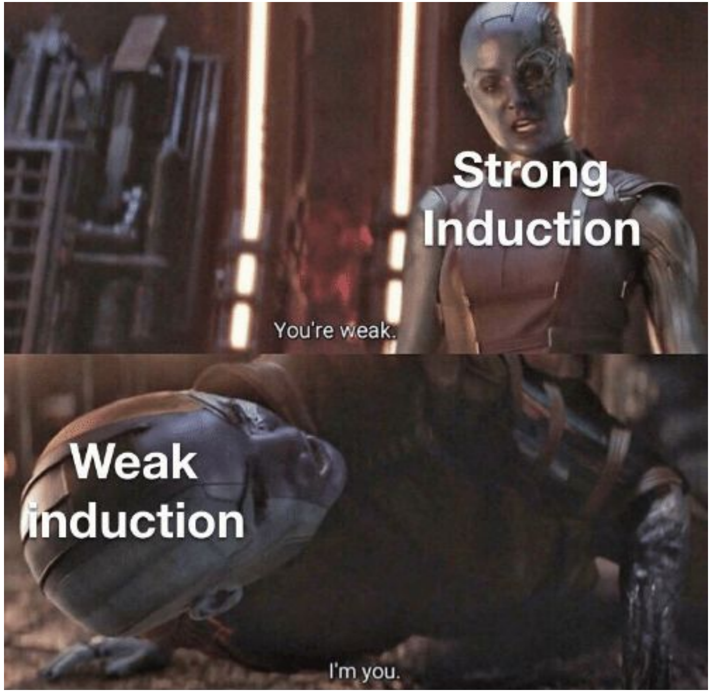
*probably needs to mention that we ignore the empty sets. This happens when city A is the start/end city of the entire country

Inductive Step: Assume *P(i)* is true for *1 <= i <= x*. To show that *P(x + 1)* is true



Induction or Strong Induction?

They're actually *equivalent*!



Induction or Strong Induction?

- Sometimes, it is easier to prove using one rather than the other

Induction or Strong Induction?

- Sometimes, it is easier to prove using one rather than the other
- The Strong Induction example can also be proven using Induction (see the reference link later)!

Induction or Strong Induction?

- Sometimes, it is easier to prove using one rather than the other
- The Strong Induction example can also be proven using Induction (see the reference link later)!
- Personal experience: I have experienced a few examples involving **trees and graphs** where **strong induction** makes it easier

Further Reading (Proofs)

- https://brilliant.org/wiki/contradiction/
- https://brilliant.org/wiki/induction/
- https://brilliant.org/wiki/strong-induction/

Combinatorics

Permutations

- "All possible arrangements"
- Eg, all permutations of "abc":

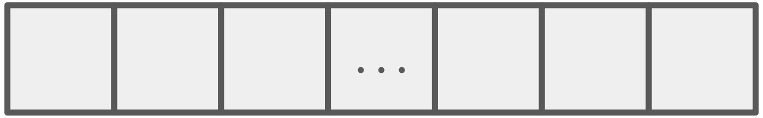
abc acb cba bac bca cab

Permutations

- If you have a set of n objects, how many permutations does the set have?

Permutations

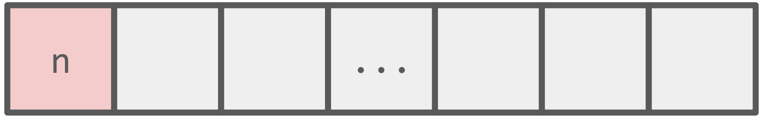
- If you have a set of *n* objects, how many permutations does the set have?
- Each slot represents the number of ways to choose an object for that slot



Permutations

- If you have a set of *n* objects, how many permutations does the set have?
- Each slot represents the number of ways to choose an object for that slot

Choosing first element...



Permutations

- If you have a set of *n* objects, how many permutations does the set have?
- Each slot represents the number of ways to choose an object for that slot

Choosing second element...

n	(n-1)		...			

Permutations

- If you have a set of *n* objects, how many permutations does the set have?
- Each slot represents the number of ways to choose an object for that slot

n	(n-1)	(n-2)	...	3	2	1

Permutations

- If you have a set of *n* objects, how many permutations does the set have?
- Each slot represents the number of ways to choose an object for that slot

n	(n-1)	(n-2)	...	3	2	1

Number of permutations:

$n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1 = \mathbf{n!}$

Permutations of Selected Elements

- *r-permutation* of n objects is all the permutations when you only take r objects

Permutations of Selected Elements

- *r-permutation* of n objects is all the permutations when you only take r objects
- Also denoted as *nPr*

Permutations of Selected Elements

- *r-permutation* of n objects is all the permutations when you only take r objects
- Also denoted as *nPr*
- Eg 2-permutation for “abcd”:

  ab, ba

  ac, ca

  ad, da

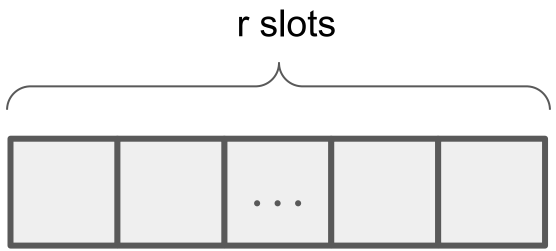
  bc, cb

  bd, db

  cd, dc

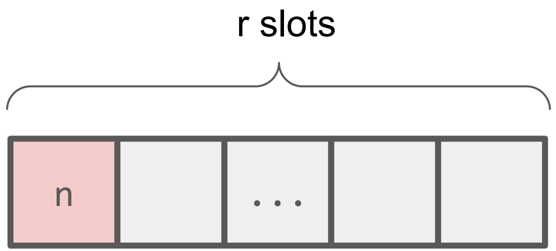
Permutations of Selected Elements

- Each slot represents the number of ways to choose an object for that slot



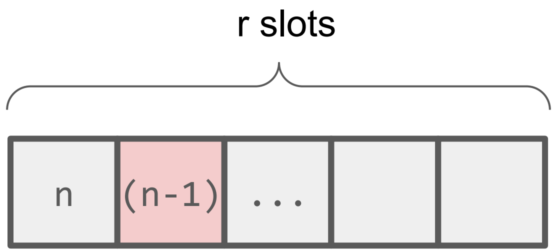
Permutations of Selected Elements

- Each slot represents the number of ways to choose an object for that slot
  Choosing first element...



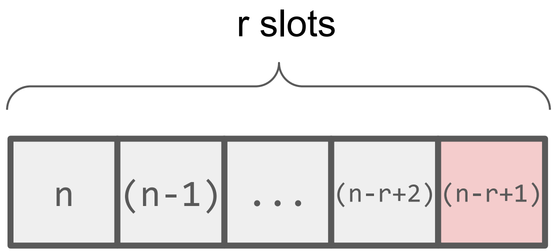
Permutations of Selected Elements

- Each slot represents the number of ways to choose an object for that slot
  Choosing second element..



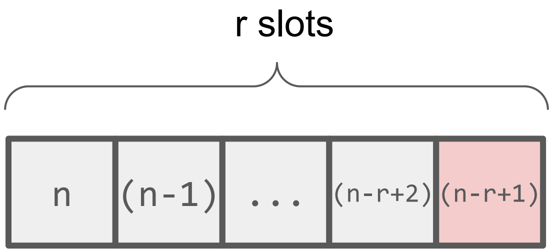
Permutations of Selected Elements

- Each slot represents the number of ways to choose an object for that slot
  All the way till r slots are filled up!



Permutations of Selected Elements

- Each slot represents the number of ways to choose an object for that slot
  All the way till r slots are filled up!



nPr

= (n)(n-1)...(n-r+2)(n-r+1) *(first version)*

Permutations of Selected Elements

- Each slot represents the number of ways to choose an object for that slot
  All the way till r slots are filled up!

$$\overbrace{\boxed{n}\boxed{(n-1)}\boxed{\ldots}\boxed{(n-r+2)}\boxed{(n-r+1)}}^{\text{r slots}}$$

nPr

= (n)(n-1)...(n-r+2)(n-r+1) *(first version)*

= n! / (n-r)! *(second version)*

Permutation of selected elements

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

Permutation of selected elements

$$\frac{n!}{(n-r)!} = \frac{n \times (n-1) \times \ldots \times (n-r+1) \times (n-r) \times (n-r-1) \times \ldots \times 1}{(n-r) \times (n-r-1) \times \ldots \times 1}$$

Permutation of selected elements

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

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

Permutation of selected elements

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

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

$$= n \times (n-1) \times \dots \times (n-r+1)$$

Combinations

- An *r-combination* of a set of n elements is a subset of r of the n elements

Combinations

- An *r-combination* of a set of n elements is a subset of r of the n elements
- Also denoted as nCr

Combinations

- An *r-combination* of a set of *n* elements is a subset of *r* of the *n* elements
- Also denoted as *nCr*
- Eg 2-combination for *{a, b, c, d}*:
  {a, b}, {a, c}, {a, d}, {b, c}, {b, d}, {c, d}

Connection between nCr and nPr

- *Consider {a, b, c, d}*

Connection between nCr and nPr

- *Consider {a, b, c, d}*

2-combination

Unordered

{a, b}

{a, c}

{a, d}

{b, c}

{b, d}

{c, d}

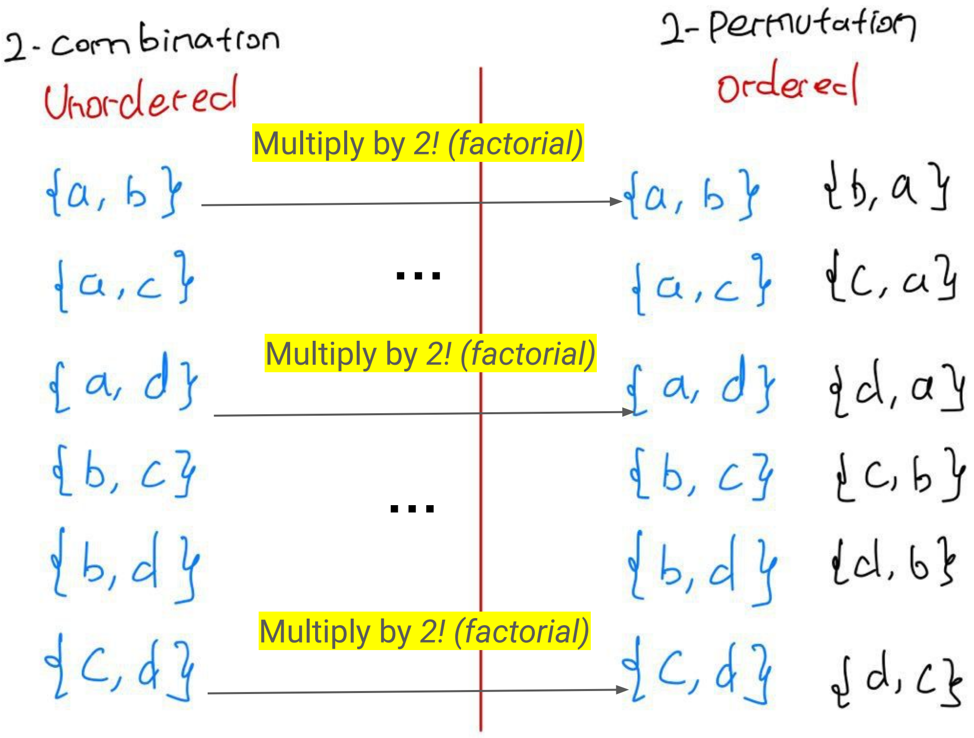
Connection between nCr and nPr

- *Consider {a, b, c, d}*

2-combination	2-permutation	
Unordered	Ordered	
{a, b}	{a, b}	{b, a}
{a, c}	{a, c}	{c, a}
{a, d}	{a, d}	{d, a}
{b, c}	{b, c}	{c, b}
{b, d}	{b, d}	{d, b}
{c, d}	{c, d}	{d, c}

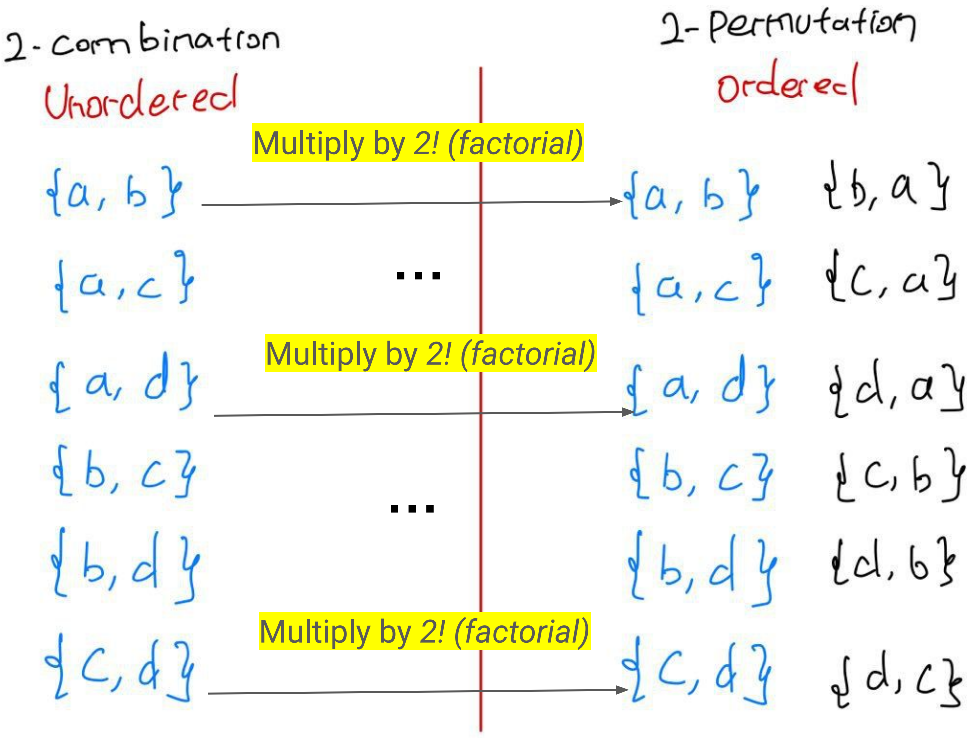
Connection between nCr and nPr

- *Consider {a, b, c, d}*



Connection between nCr and nPr

- *Consider {a, b, c, d}*



Therefore:

*nCr * r! = nPr*

Formula for nCr

From the previous slide, we have *nCr * r! = nPr:*

Formula for nCr

From the previous slide, we have *nCr * r! = nPr:*

$$\binom{n}{r} = \frac{n!}{(n-r)!} \cdot \frac{1}{r!}$$

Formula for nCr

From the previous slide, we have *nCr * $\boxed{r!}$ = nPr:*

$$\boxed{\binom{n}{r}} = \boxed{\frac{n!}{(n-r)!}} \cdot \frac{1}{\boxed{r!}}$$

Another way to write
nCr

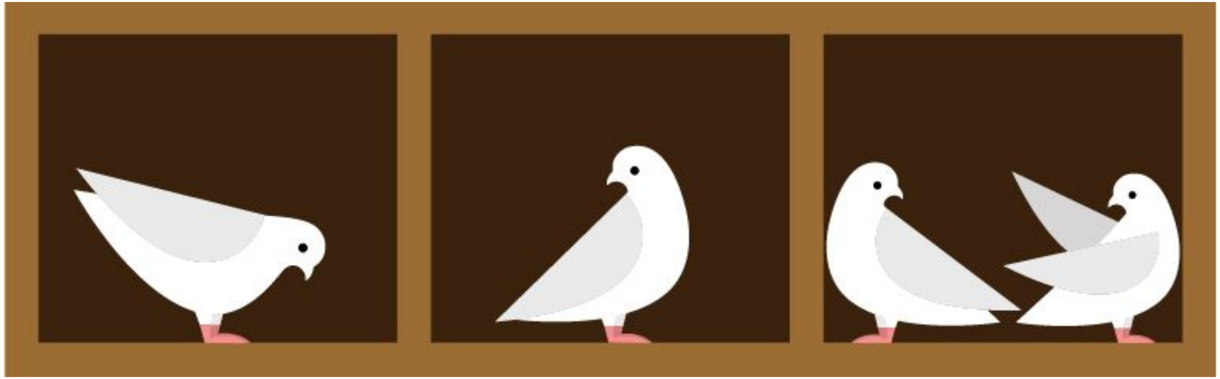
nPr

Pigeonhole Principle

- Simplest form: If we have **n pigeonholes**, if you have at least **n + 1 pigeons**, then, one pigeonhole must contain more than one pigeon

Pigeonhole Principle

- Simplest form: If we have **n pigeonholes**, if you have at least **n + 1 pigeons**, then, one pigeonhole must contain more than one pigeon



- Application: Usually a matter of of identifying the pigeon and pigeonholes!

Pigeonhole Principle (Example)

Assume there are 365 days in a year. How many people do you need in a room to guarantee that there is at least one pair of people who have the same birthdays?

Pigeonhole Principle (Example)

Assume there are 365 days in a year. How many people do you need in a room to guarantee that there is at least one pair of people who have the same birthdays?

Identify the Pigeonhole and Pigeons!

Pigeonhole = Birthdays
Pigeon = People

Pigeonhole Principle (Example)

Assume there are 365 days in a year. How many people do you need in a room to guarantee that there is at least one pair of people who have the same birthdays?

Identify the Pigeonhole and Pigeons!

Pigeonhole = Birthdays
Pigeon = People

By Pigeonhole Principle, since we have 365 pigeonholes, we need **366** pigeons (people) so that we are guaranteed one pigeonhole (birthday) has 2 pigeons!

Exercise (Combinatorics)

1. Show that if we have a set of size *n*, then the total number of subsets is 2^n (the set of all subsets is also called the powerset)
2. Prove:

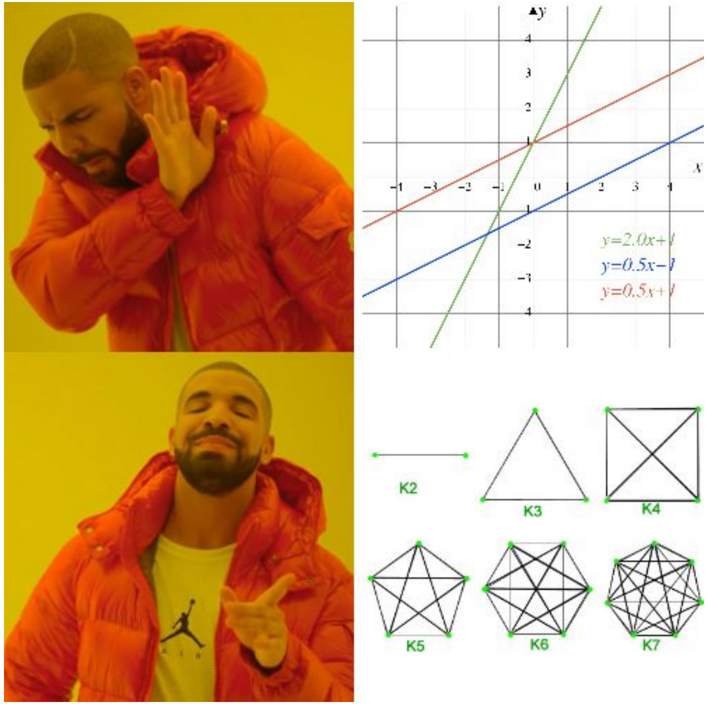
$$\binom{n+1}{r} = \binom{n}{r} + \binom{n}{r-1}$$

3. Prove:

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

Hint: You might want to refresh on Combinatorial arguments to work on questions 2 and 3. Although these can still be proven algebraically / through other proof methods, I think it's good to revise combinatorial arguments as well!

Graph Theory

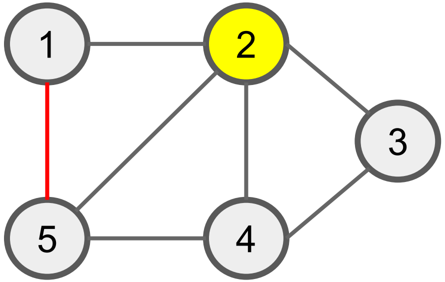


Graph Definitions

- Graph consists of:
  - V, a set of nodes/vertices
  - E, a set of edges

Graph Definitions

- Graph consists of:
  - *V*, a set of nodes/vertices
  - *E*, a set of edges



$V = \{1, 2, 3, 4, 5\}$
$E = \{(1, 2), (1, 5), (2, 3), (2, 4), (2, 5), (3, 4), (4, 5)\}$

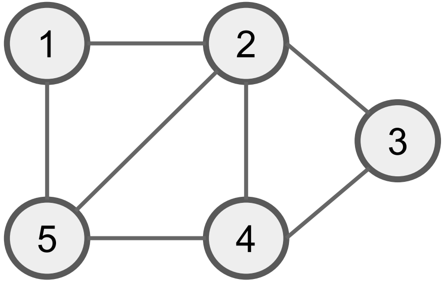
Graph Representation

- How do you represent Graphs in a program?
  - Adjacency List
  - Adjacency Matrix
  - Edge List (not covered in these slides)

Adapted from CLRS

Adjacency List (Undirected Graph)

How to represent this?

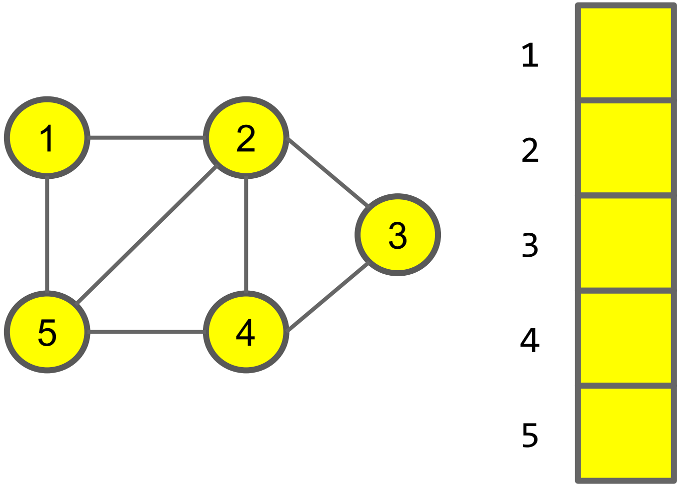


Adapted from CLRS

Adjacency List (Undirected Graph)

How to represent this?

Array: The nodes



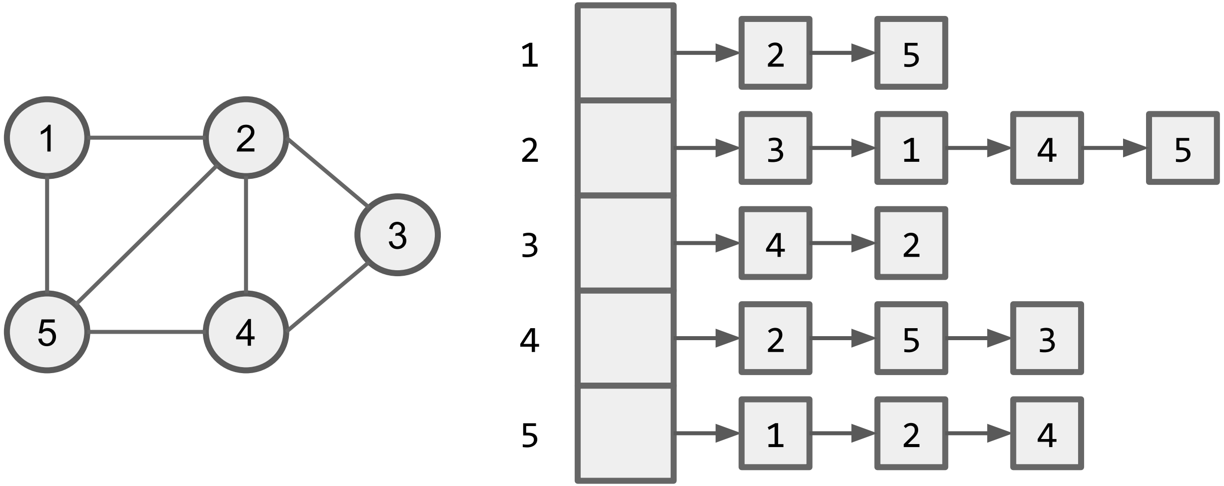
Adapted from CLRS

Adjacency List (Undirected Graph)

How to represent this?

Array: The nodes

Linked List: The edges representing the other nodes it is connected to



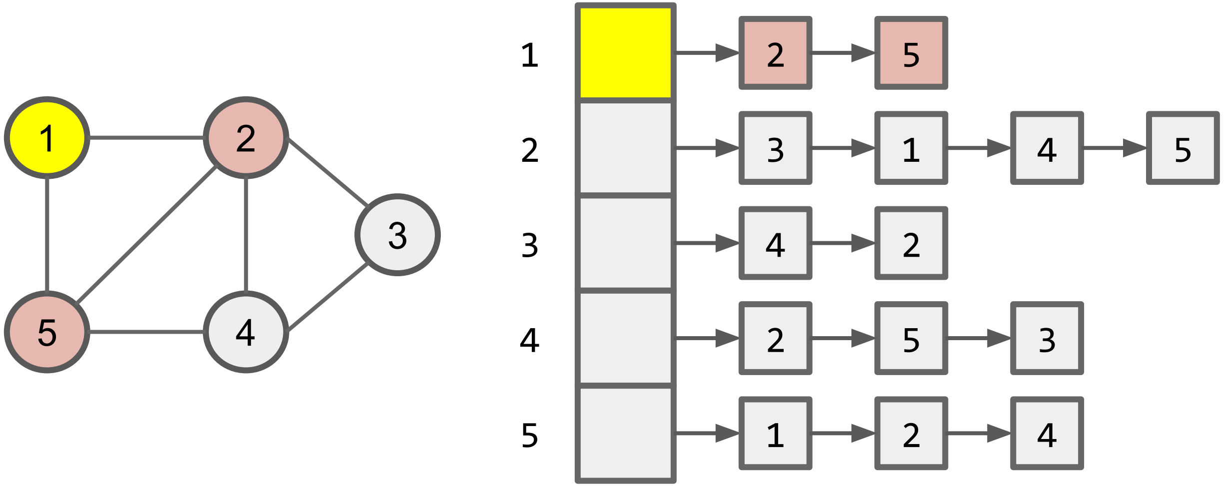
Adapted from CLRS

Adjacency List (Undirected Graph)

How to represent this?

Array: The nodes

Linked List: The edges representing the other nodes it is connected to

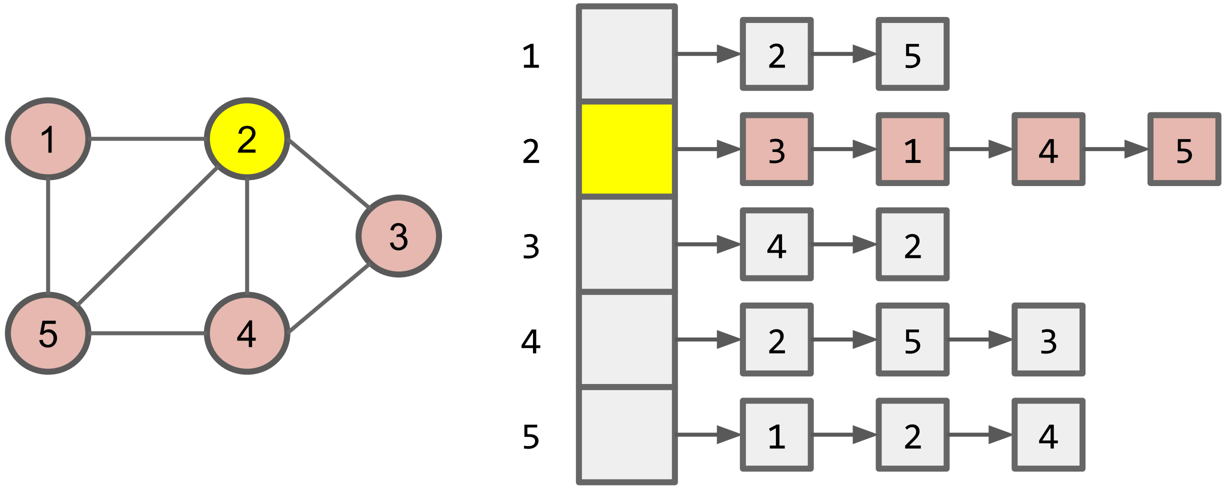


Adjacency List (Undirected Graph)

How to represent this?

Array: The nodes

Linked List: The edges representing the other nodes it is connected to



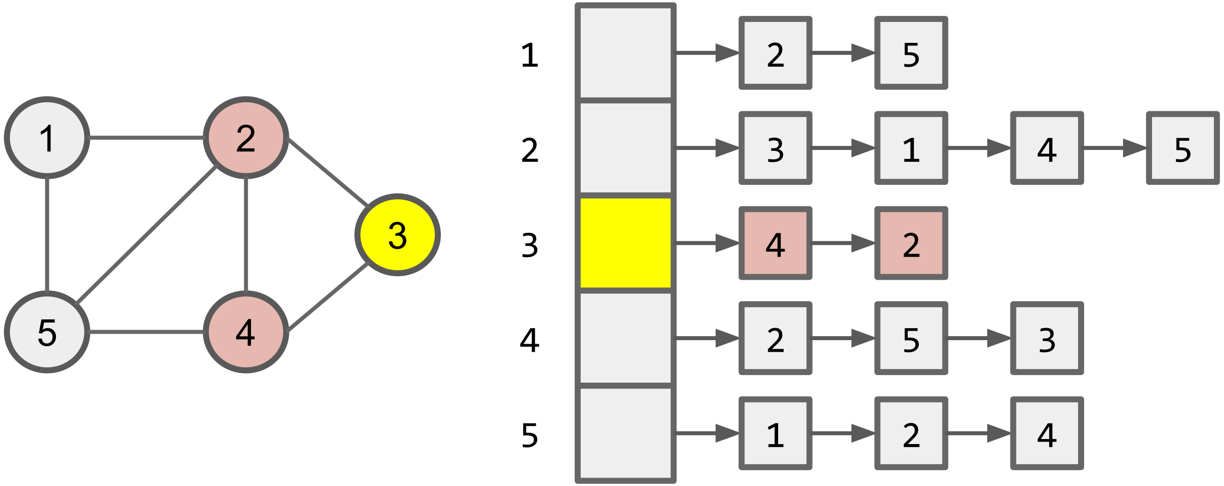
Adapted from CLRS

Adjacency List (Undirected Graph)

How to represent this?

Array: The nodes

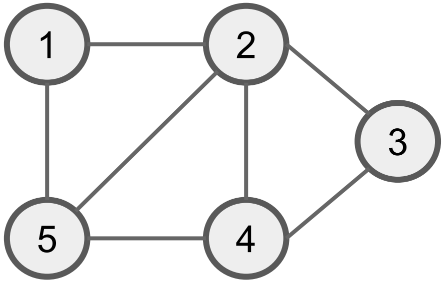
Linked List: The edges representing the other nodes it is connected to



Adapted from CLRS

Adjacency Matrix (Undirected Graph)

How to represent this?

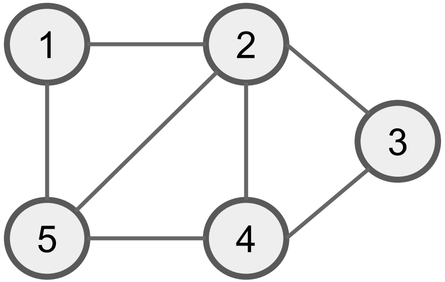


Adapted from CLRS

Adjacency Matrix (Undirected Graph)

How to represent this?

Matrix A: `A[u][v] == 1` iff `(u, v) ∈ E`



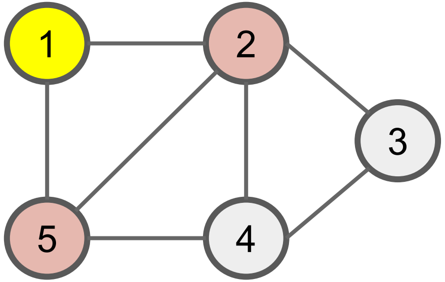
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Note: Notice that this is a symmetric matrix!

Adjacency Matrix (Undirected Graph)

How to represent this?

Matrix A: `A[u][v] == 1` iff `(u, v) ∈ E`



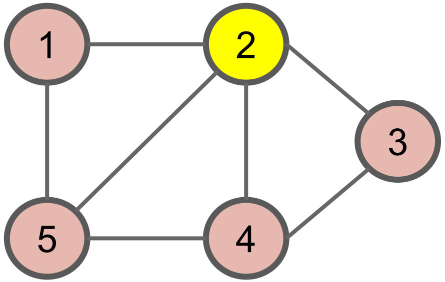
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Adapted from CLRS

Adjacency Matrix (Undirected Graph)

How to represent this?

Matrix A: `A[u][v] == 1` iff `(u, v) ∈ E`



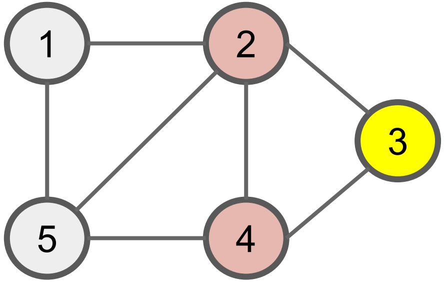
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Adapted from CLRS

Adjacency Matrix (Undirected Graph)

How to represent this?

Matrix A: `A[u][v] == 1` iff `(u, v) ∈ E`



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

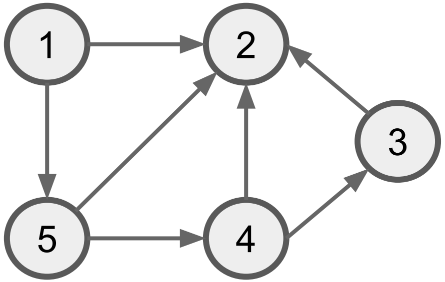
Adapted from CLRS

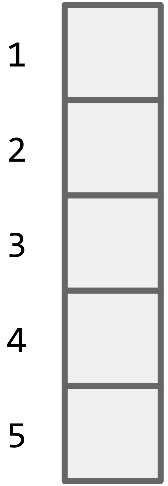
Adjacency List (Directed Graph)

How to represent this?

Array: The nodes

Linked List: The edges representing the other nodes it is connected to





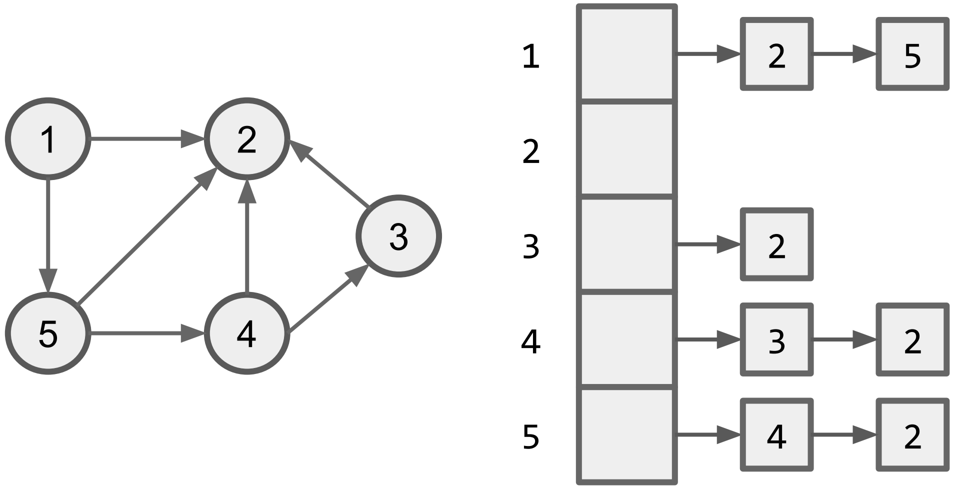
Adapted from CLRS

Adjacency List (Directed Graph)

How to represent this?

Array: The nodes

Linked List: The edges representing the other nodes it is connected to



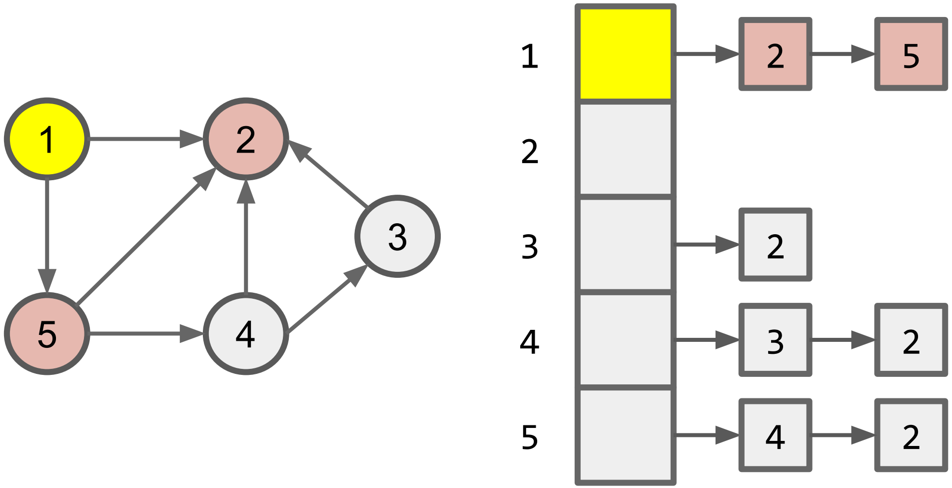
Adapted from CLRS

Adjacency List (Directed Graph)

How to represent this?

Array: The nodes

Linked List: The edges representing the other nodes it is connected to



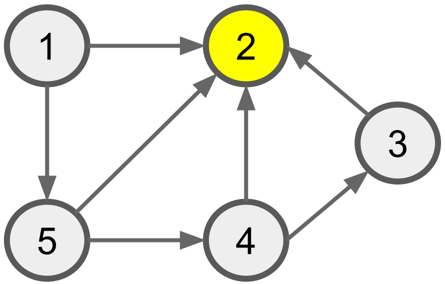
Adapted from CLRS

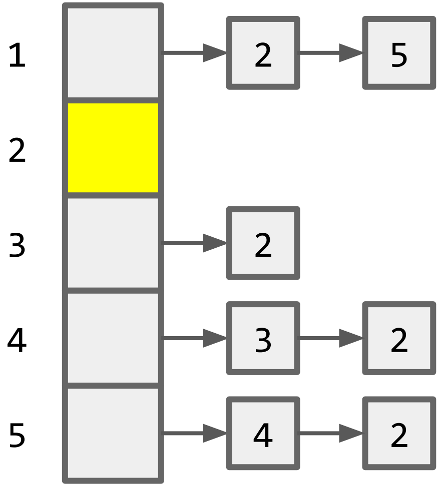
Adjacency List (Directed Graph)

How to represent this?

Array: The nodes

Linked List: The edges representing the other nodes it is connected to





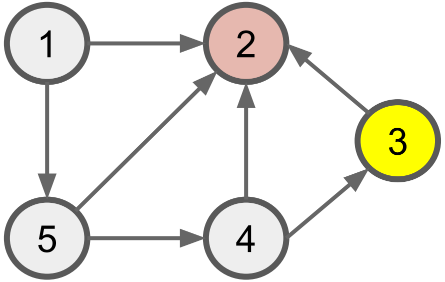
Adapted from CLRS

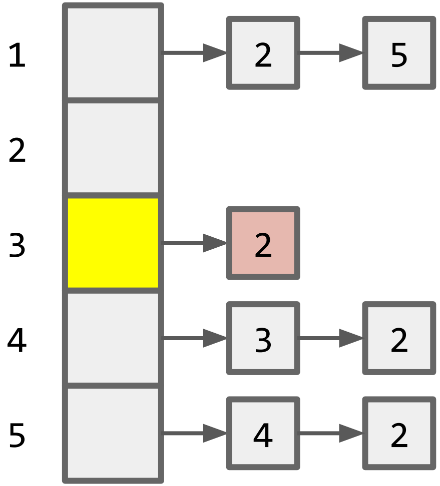
Adjacency List (Directed Graph)

How to represent this?

Array: The nodes

Linked List: The edges representing the other nodes it is connected to

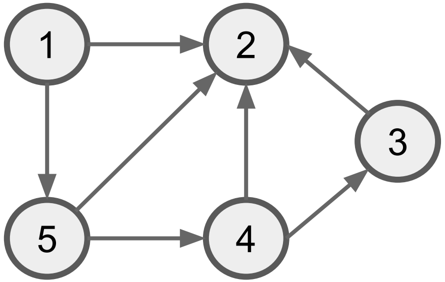




Adapted from CLRS

Adjacency Matrix (Directed Graph)

How to represent this?



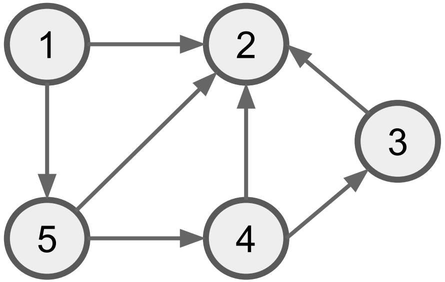
Adapted from CLRS

Adjacency Matrix (Directed Graph)

How to represent this?

Matrix A: `A[u][v] == 1` iff `(u, v) ∈ E`

Note: (u, v) is different from (v, u) if edges are directed



	1	2	3	4	5
1	0	1	0	0	1
2	0	0	0	0	0
3	0	1	0	0	0
4	0	1	1	0	0
5	0	1	0	1	0

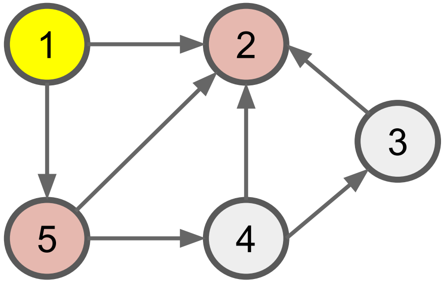
Adapted from CLRS

Adjacency Matrix (Directed Graph)

How to represent this?

Matrix A: `A[u][v] == 1` iff `(u, v) ∈ E`

Note: (u, v) is different from (v, u) if edges are directed



	1	2	3	4	5
1	0	1	0	0	1
2	0	0	0	0	0
3	0	1	0	0	0
4	0	1	1	0	0
5	0	1	0	1	0

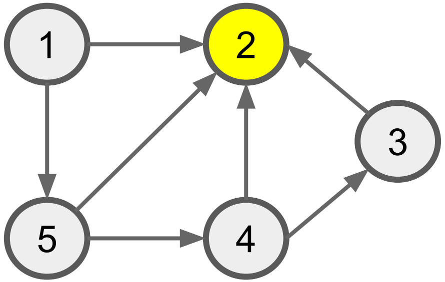
Adapted from CLRS

Adjacency Matrix (Directed Graph)

How to represent this?

Matrix A: `A[u][v] == 1` iff `(u, v) ∈ E`

Note: (u, v) is different from (v, u) if edges are directed



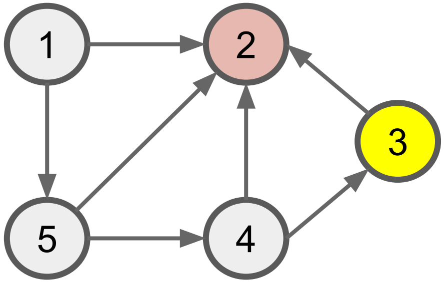
	1	2	3	4	5
1	0	1	0	0	1
2	0	0	0	0	0
3	0	1	0	0	0
4	0	1	1	0	0
5	0	1	0	1	0

Adjacency Matrix (Directed Graph)

How to represent this?

Matrix A: `A[u][v] == 1` iff `(u, v) ∈ E`

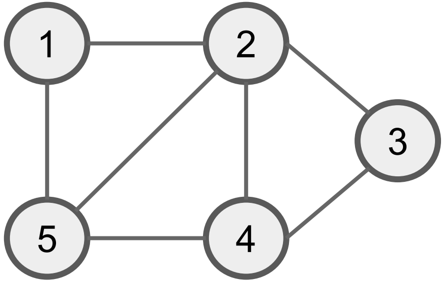
Note: (u, v) is different from (v, u) if edges are directed

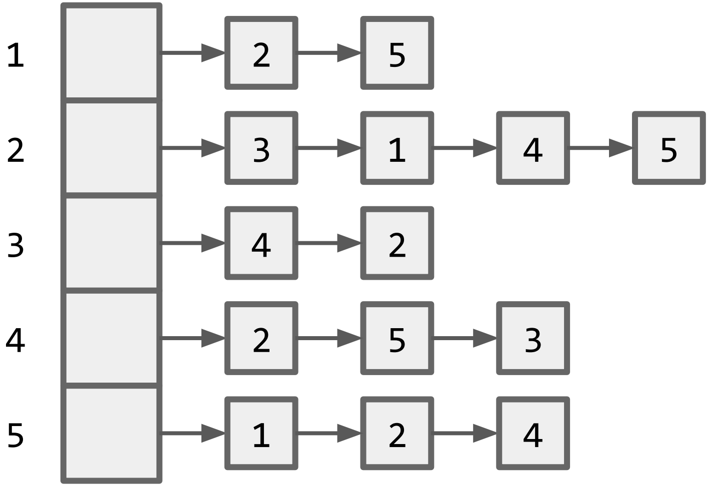


	1	2	3	4	5
1	0	1	0	0	1
2	0	0	0	0	0
3	0	1	0	0	0
4	0	1	1	0	0
5	0	1	0	1	0

Graph Representations

Space complexity?

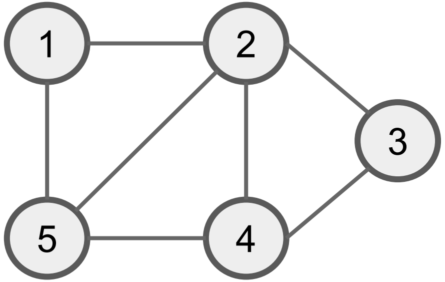




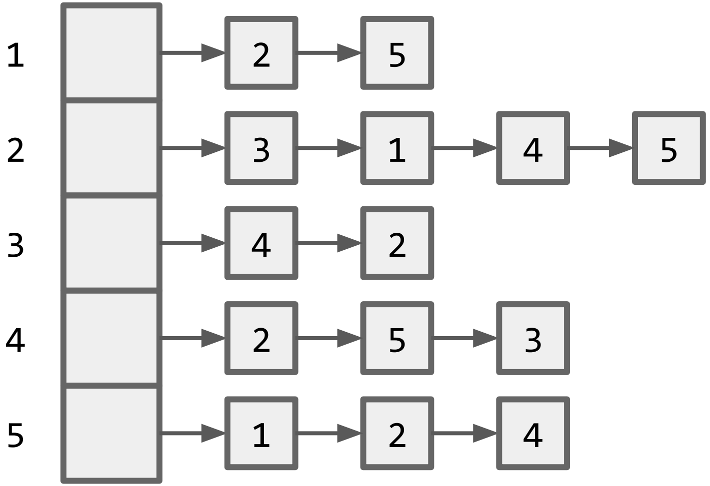
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Graph Representations

Space complexity?



$O(V + E)$



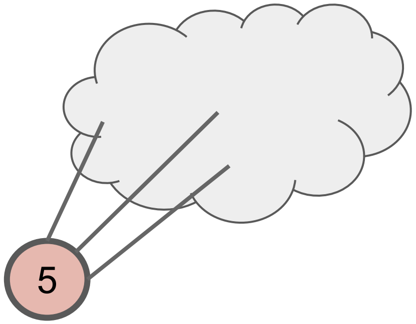
$O(V^2)$

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Degree of a Graph

Let *v* be a vertex in graph *G*, the **degree** of the *v*, denoted by *deg(v)* is the *number of edges incident on v*.

E.g. this vertex has degree 3:

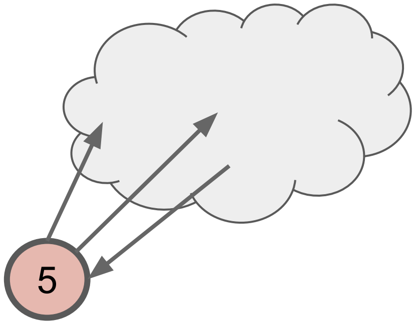


Degree of a Graph

Indegree = Incident edges that are **incoming**
Outdegree = Incident edges that are **outgoing**

e.g.

Indegree of this vertex = 1
Outdegree of this vertex = 2



Handshaking Lemma

The **total degree of graph G** is the sum of the degrees of all vertices of G.

Handshaking Lemma

The **total degree of graph G** is the sum of the degrees of all vertices of G.

In an undirected graph G, the total degree of G is **twice** the number of **edges:**

$$\sum_{v \in V} deg(v) = 2|E|$$

Handshaking Lemma

The **total degree of graph G** is the sum of the degrees of all vertices of G.

In an undirected graph G, the total degree of G is **twice** the number of **edges:**

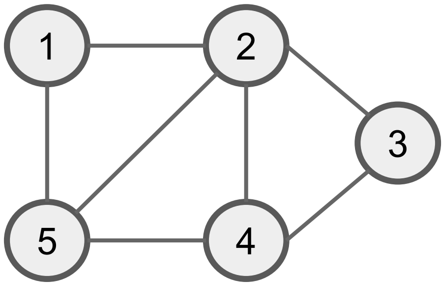
$$\sum_{v \in V} deg(v) = 2|E|$$

Note: Keep this lemma in mind! I have had to use this lemma to prove some graph properties or to analyse runtime of algorithms!

Handshaking Lemma

$$\sum_{v \in V} deg(v) = 2|E|$$

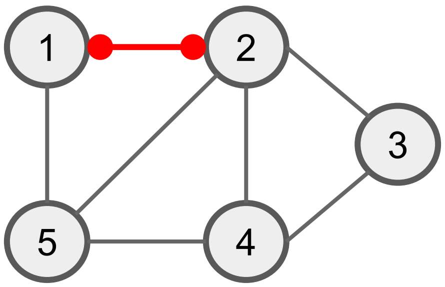
Idea: Consider every edge - each one contributes 2 to the total degree!



Handshaking Lemma

$$\sum_{v \in V} deg(v) = 2|E|$$

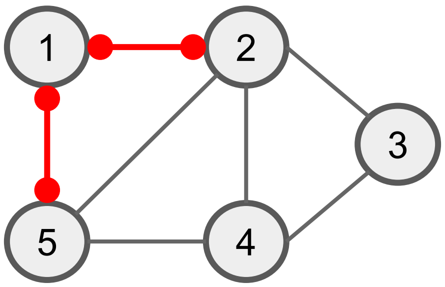
Idea: Consider every edge - each one contributes 2 to the total degree!



Handshaking Lemma

$$\sum_{v \in V} deg(v) = 2|E|$$

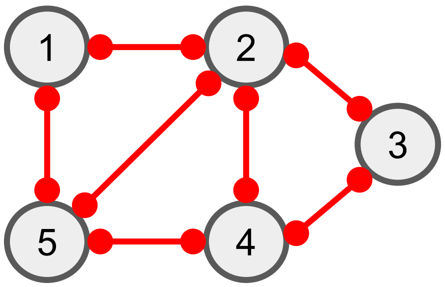
Idea: Consider every edge - each one contributes 2 to the total degree!



Handshaking Lemma

$$\sum_{v \in V} deg(v) = 2|E|$$

Idea: Consider every edge - each one contributes 2 to the total degree!

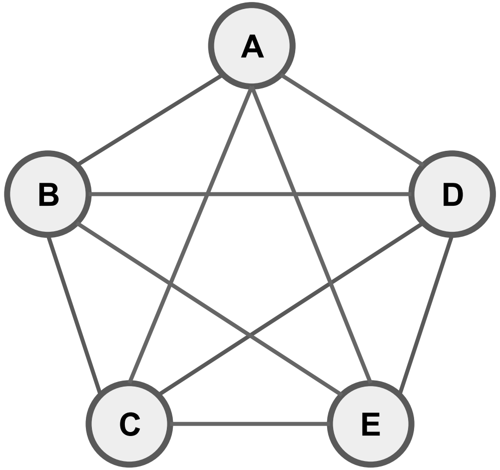


Complete Graph

Clique / Complete Graph: All pairs connected by edges

Complete Graph

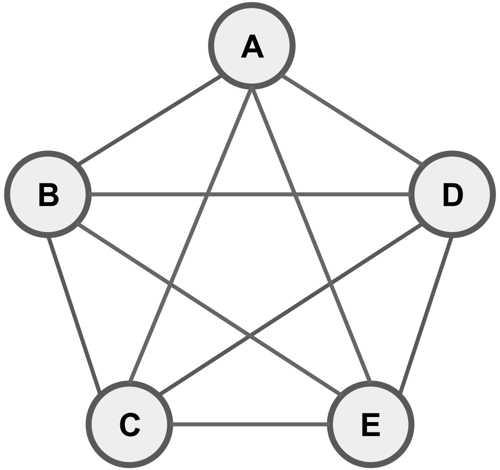
Clique / Complete Graph: All pairs connected by edges



Complete Graph

Clique / Complete Graph: All pairs connected by edges

How many edges are there if there are *n* nodes?

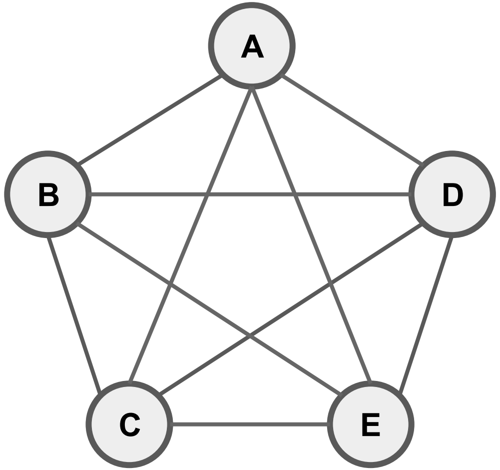


Complete Graph

Clique / Complete Graph: All pairs connected by edges

How many edges are there if there are *n* nodes?

$$\binom{n}{2} = \frac{n(n-1)}{2} = O(n^2)$$



Bipartite Graph

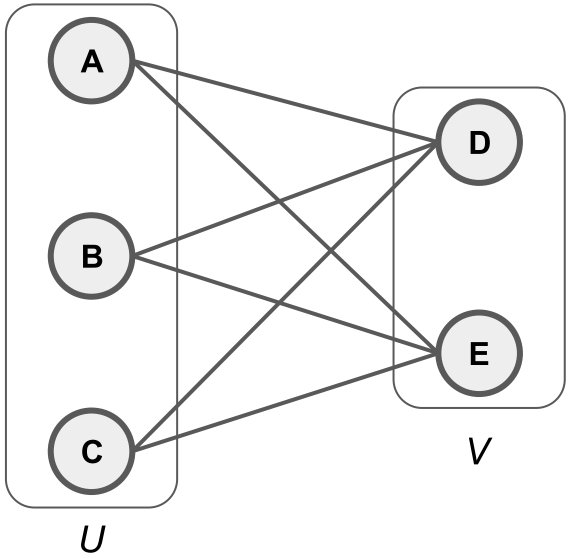
Graph whose vertices can be divided into two disjoint sets ***U*** and ***V***, such that every edge connects one vertex in ***U*** and one vertex in ***V***

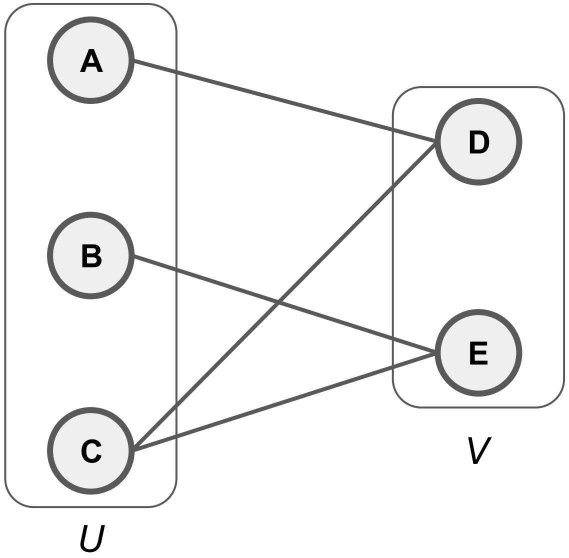
(Within the set U or V, none of the vertices are connected)

Bipartite Graph

Graph whose vertices can be divided into two disjoint sets ***U*** and ***V***, such that every edge connects one vertex in ***U*** and one vertex in ***V***

(Within the set U or V, none of the vertices are connected)





DAG

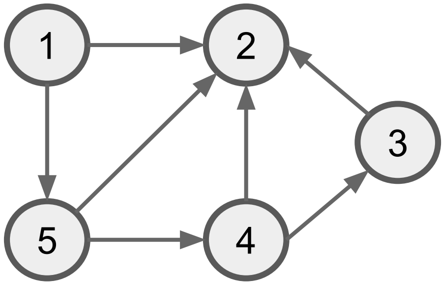

Don't Spread Germs
This Winter
Germs
Destroy. All. ~~Bacteria.~~
When You Sneeze

Directed Acyclic Graph

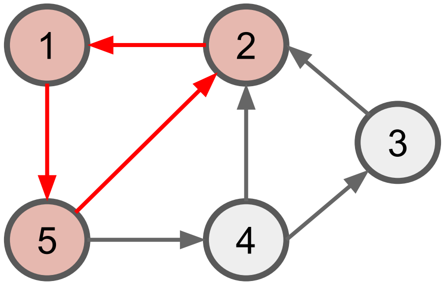
- Directed:
- Acyclic:

Directed Acyclic Graph

- Directed: It has directions :D
- Acyclic: It has no cycles



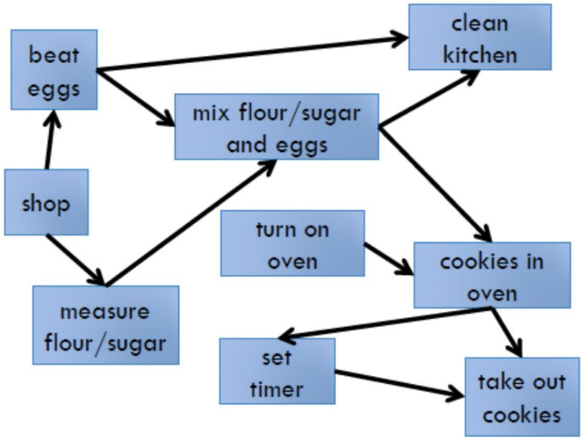
NOT a Directed Acyclic Graph



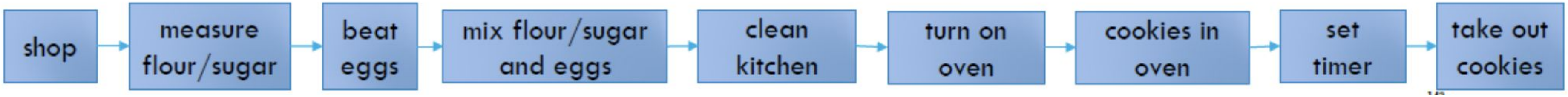
Taken from Prof Harold’s AY19/20 S1 CS2040S slides (with permission)

Topological Ordering

Given a DAG:



Come up with a sequential ordering that only “points forward”:



Graph Algorithms you should know

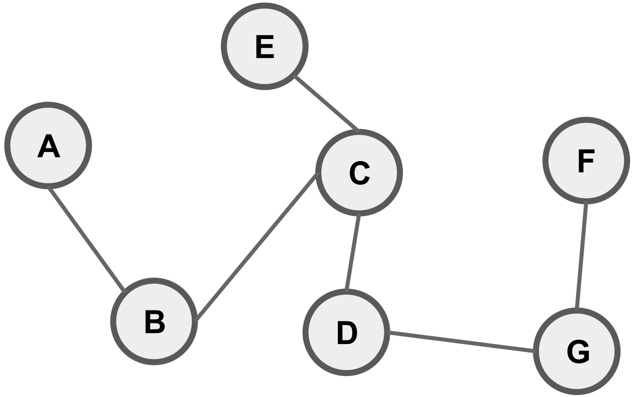
- Depth-First Search
- Breadth-First Search
- Topological Sort (Postorder DFS or BFS with Kahn's Algorithm)
- Dijkstra's Algorithm (for Single Source Shortest Path)
- Bellman-Ford (for Single Source Shortest Path)

Maybe (depending on which class you took):

- Floyd-Warshall (for All-Pairs Shortest Path)

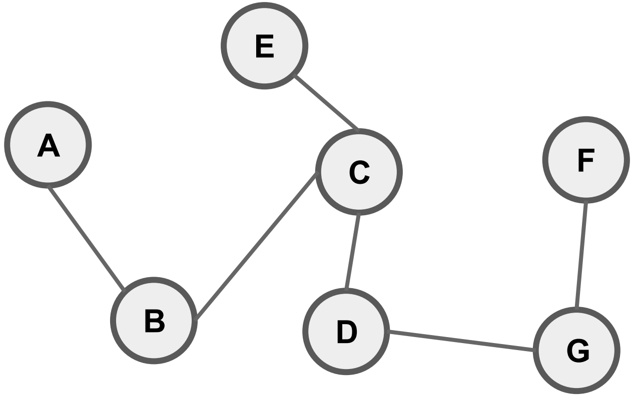
Trees

Tree is a connected graph with **no cycles**



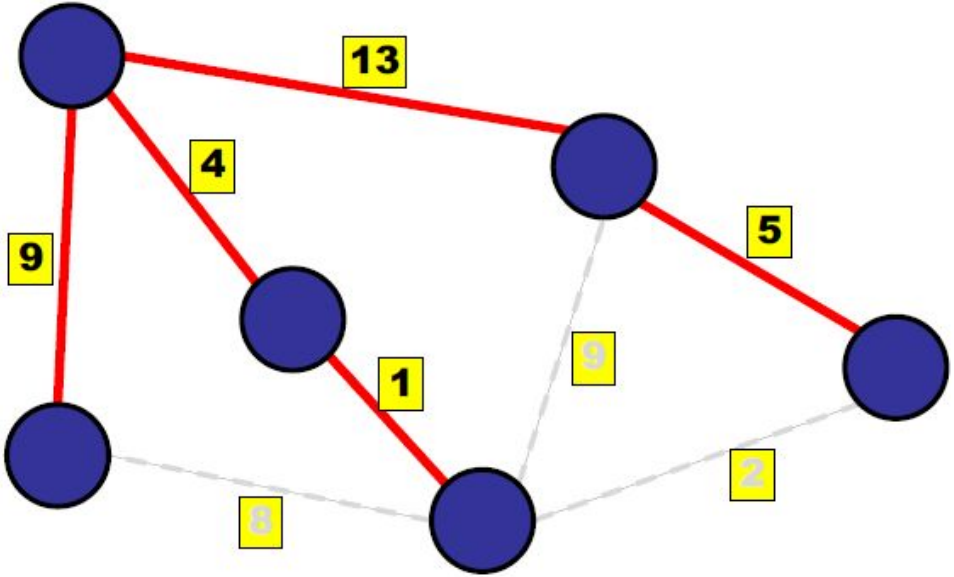
Equivalent Statement on Trees

1. Tree *T* is a graph with no cycles
2. Every two distinct vertices in *T* are joined by a **unique path**
3. $|V| = |E| + 1$



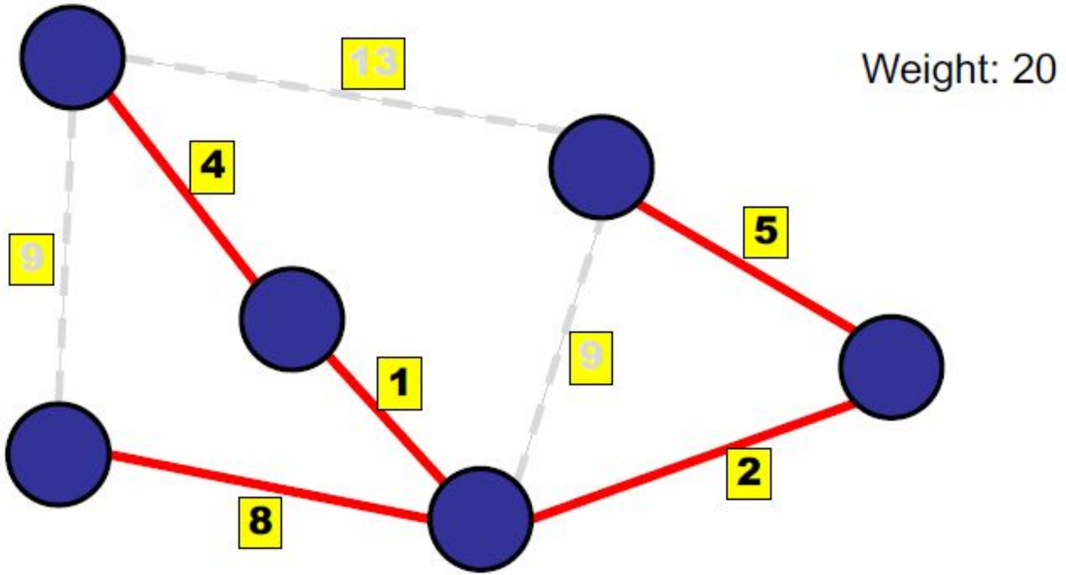
Spanning Tree

Definition: a spanning tree is an acyclic subset of the edges that connects all nodes



Minimum Spanning Tree

Definition: a spanning tree with minimum weight



Algorithms for Minimum Spanning Trees

You may or may not know (depending on which algorithms class you took):

- Prim’s Algorithm
- Kruskal’s Algorithm

But do learn them if you haven’t :)

Another MST algorithm that is rarely discussed:

- Boruvka’s Algorithm

That was a lot!

- If that was easy stuff for you -- great!
- If you are uncomfortable with most of the material I shared / unfamiliar with them -- do spend some time revising!

That was a lot!

- If that was easy stuff for you -- great!
- If you are uncomfortable with most of the material I shared / unfamiliar with them -- do spend some time revising!

- I couldn't have covered everything! But I tried to cover the tools and techniques you might often need in CS3230

All the best for CS3230!