- Unit I: Basics of Combinatorics
- Unit II: Recurrence Relations and Generative Functions
- Units III-V: Graph Theory
  - → Unit III: Basics of Graphs, Trees
  - → Unit IV: Euler and Hamiltonian Graphs, Planar and Dual Graphs, Graph Colouring
  - → Unit V: NP-Hard and NP-Complete Problems (NP = non-deterministic poly^{l}.) ("Most difficult" problems in computation)

Recommended Books

- Rosen
- Grimaldi
- N Deo
- Cormen

Basics of Combinatorics

Induction

- An inductive proof operates over a subset of the set of positive integers.
- It consists of
  - → Base case: $P(1)$ is true
  - → Inductive hypothesis: $\forall_k (P(k) \rightarrow P(k+1))$
  (i.e. assumption that for any $k \in \mathbb{Z}^+$, if $P(k)$ is true then $P(k+1)$ is true)

  Together, they form the first principle of mathematical induction, which helps construct an inductive proof. (Concl^{n}. reached via rules of inference)
- Real-life examples: Falling of dominoes, climbing a ladder.

Q: Prove that the sum of the first n positive integers is $\frac{1}{2}n(n+1)$.

A: "$P(1)$: The sum of the first 1 positive integer is $\frac{1}{2}(1)(2) = 1$, which is true." (Base Case)

*Assume that "$P(k)$: The sum of the first k positive integers is $\frac{1}{2}k(k+1)$" is true, i.e. $1+2+3+\ldots+k = \frac{1}{2}k(k+1)$. (Inductive hypothesis)

Then, $1+2+3+\ldots+k+(k+1) = \frac{1}{2}k(k+1)+(k+1) = (k+1)(\frac{1}{2}k+1) = \frac{1}{2}(k+1)(k+2)$,

i.e. "$P(k+1)$: The sum of the first $(k+1)$ positive integers is $\frac{1}{2}(k+1)((k+1)+1)$" is true.

$\therefore$ By the principle of mathematical induction, the stmt. is proved.

Q: Prove that the sum of the first n odd positive integers is n^2.

A: Base case: $P(1)$: $1 = 1^2$, which is true.

Inductive hypothesis: $P(k)$: $1+3+\ldots+(2k-1) = k^2$ ($\because k^{th}$ odd +ve int. $= 2k-1$)

$\therefore P(k+1) = 1+3+\ldots+(2k+1) = k^2+2k+1 = (k+1)^2$, i.e. $P(k) \rightarrow P(k+1)$.

$\therefore$ By the principle of mathematical induction, the stmt. is proved.

Extras:

4

Q: Prove that $2^n > n \ \forall n \in \mathbb{Z}^+$.

A: Base case: P(1): $2^1 > 1$, which is true.

Inductive hypothesis: P(k): $2^k > k$

$\therefore 2^k + 1 > k+1$, $2^k + 1 \leq 2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$ $(\because 2^k \geq 1)$

Hence $2^{k+1} \geq 2^k + 1 > k+1 \Rightarrow$ P(k): $2^{k+1} > k+1$ is true, i.e. $P(k) \rightarrow P(k+1)$.

$\therefore$ By the principle of mathematical induction, the statement is proved.

Q: The harmonic series is denoted by $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$. Prove that $H_{2^n} \geq 1 + \frac{n}{2}$, where n is a non-negative integer.

A: Base case: P(0): $H_{2^0} \geq 1 + 0/2 \Rightarrow 1 \geq 1$, which is true.

Inductive hypothesis: P(k): $H_{2^k} \geq 1 + \frac{k}{2}$, i.e. $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{2^k} \geq 1 + \frac{k}{2}$.

$$\therefore H_{2^{k+1}} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{2^k} + \frac{1}{2^k+1} + \frac{1}{2^k+2} + \ldots + \frac{1}{2^{k+1}}$$

$$\geq 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{2^k} + \underbrace{\left(\frac{1}{2^{k+1}} + \frac{1}{2^{k+1}} + \ldots + \frac{1}{2^{k+1}}\right)}_{2^k \text{ terms}} = H_{2^k} + \frac{2^k}{2^{k+1}} = H_{2^k} + \frac{1}{2}$$

$$\geq 1 + \frac{k}{2} + \frac{1}{2} = 1 + \frac{k+1}{2}.$$

i.e. P(k+1): $H_{2^{k+1}} \geq 1 + \frac{k+1}{2}$ is true. $\therefore P(k) \rightarrow P(k+1)$, hence by the principle of mathematical induction, the statement is proved.

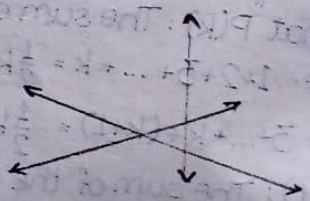
- Note: The harmonic series is divergent, i.e. $H_n \rightarrow \infty$ as $n \rightarrow \infty$.

Q: "Prove" that for every set of lines in a plane, no two of which are parallel, there exists a single point of concurrence.

A: Base case: P(2): ~~Dc~~ 2 non-parallel lines intersect, which is true.

Inductive hypothesis: We can't assume P(k) is true, because $P(2) \rightarrow P(3)$ is false:

P(2) is true



P(3) is false

$\therefore$ We can't make a hypothesis because an intermediate step is false.

$\therefore$ The principle of mathematical induction can't be used to prove this stmt.

Turns out this statement is actually false.

Q: Prove that the sum of the squares of the first $(n+1)$ odd positive integers is $\frac{1}{3}(n+1)(2n+1)(2n+3)$.

A: Base case: P(0): $1^2 = \frac{1}{3}(1)(1)(3)$, which is true.

Extras:

Inductive hypothesis: Let $P(k)$ be true, i.e.

let $1^2+3^2+5^2+\ldots+(2k+1)^2=\frac{1}{3}(k+1)(2k+1)(2k+3)$.

Adding the square of the $(k+2)^{th}$ odd positive integer, i.e. $(2k+3)^2$, we get

$1^2+3^2+5^2+\ldots+(2k+1)^2+(2k+3)^2 = \frac{1}{3}(k+1)(2k+1)(2k+3)+(2k+3)(2k+3)$

$=(2k+3)(\frac{1}{3}(k+1)(2k+1)+\frac{1}{3}(6k+9)) = (2k+3)(\frac{1}{3}(2k^2+3k+1+6k+9))$

$=(2k+3)(\frac{1}{3}(2k^2+9k+10)) = \frac{1}{3}(2k+3)(k+2)(2k+5) = \frac{1}{3}((k+1)+1)(2(k+1)+1)(2(k+1)+3)$,

i.e. $1^2+3^2+5^2+\ldots+(2k+3)^2 = \frac{1}{3}((k+1)+1)(2(k+1)+1)(2(k+1)+3)$, i.e. $P(k+1)$ is true.

$\therefore P(k) \rightarrow P(k+1)$, hence the stmt. is proved by mathematical induction.

Q: Prove that for all non-negative integers n, $\sum_{i=0}^{n} 3\cdot 5^i = \frac{3}{4}(5^{n+1}-1)$.

A: Base case: $P(0)$: $3\cdot 5^0 = \frac{3}{4}(5^1-1)$, which is true.

Inductive hypothesis: Let $P(k)$ be true, i.e.

$3\cdot 5^0+3\cdot 5^1+3\cdot 5^2+\ldots+3\cdot 5^k = \frac{3}{4}(5^{k+1}-1)$. Adding $3\cdot 5^{k+1}$ to both sides,

Then, $3\cdot 5^0+3\cdot 5^1+3\cdot 5^2+\ldots+3\cdot 5^{k+1} = \frac{3}{4}(5^{k+1}-1)+3\cdot 5^{k+1} = \frac{3}{4}(5^{k+1}-1+4\cdot 5^{k+1})$

$=\frac{3}{4}(5\cdot 5^{k+1}-1) = \frac{3}{4}(5^{(k+1)+1}-1)$, i.e. $P(k+1)$ is true. $\therefore P(k) \rightarrow P(k+1)$

and the stmt. is proved by mathematical induction.

- Another method is the second principle of mathematical induction (strong ind^n.)
  - → We verify the base case ($P(1)$).
  - → We prove the truth of all the statements $P(2), P(3), \ldots P(k)$, $k \in \mathbb{N}$.
  - → Using this we prove the truth of $P(k+1)$, i.e. we attempt to prove $[P(2) \wedge P(3) \wedge \ldots P(k)] \rightarrow P(k+1)$.
  - → Once proved, we can conclude that $P(n)$ is true for all n.

Q: Prove that any element in the set $\{n : n \in \mathbb{N} \text{ and } n>1\}$ can be either written as the product of one or more primes, or is a prime itself.

A: Base case: $P(2)$ is true. $(2=2^1)$ (2 is prime)

Inductive step: Assume that for some $k \in \mathbb{N}$, $[P(2) \wedge P(3) \wedge \ldots \wedge P(k)]$ is true.

In trying to prove $P(k+1)$,

- → if $k+1$ is prime, then $P(k+1)$ is true (directly).
- → if $k+1$ is composite, then we can write (by def^n.) $k+1 = a\cdot b$, $1<a,b<j+1$. But according to the inductive step, $P(a)$ and $P(b)$ are true. Hence $k+1$ is a product of some number of primes, thus $P(k+1)$ is true.

$\therefore$ By mathematical induction, the stmt. is proved.

Note: Strong induction is a better, more rigorous form of proof than the first principle of mathematical induction.

Extras:

Axioms for the Set of Positive Integers

- The number 1 is a positive integer.
- If n is a positive integer, its successor, i.e. n+1, is also a positive integer.
- Every positive integer greater than 1 is the successor of exactly one +ve integer.
- Well-Ordering Property: Every non-empty subset of the set of positive integers has a least/minimum element.

Q: Prove that if $a \in \mathbb{Z}$ and $d \in \mathbb{Z}^+$, there exist unique $q, r \in \mathbb{Z}^*$ such that $0 \leq r < d$ and $a = dq + r$.

A: Consider a set $S \subseteq \{x : x \in \mathbb{Z} \text{ and } x \geq 0\}$ such that $S = \{a - dq, q \in \mathbb{Z}\}$.
(Note: S is a subset of the non-negative integers)

∴ By the well-ordering property, S has a least element. Let this element be r. Clearly, $r \geq 0$.

If $r \geq d$, we can write $r = d + r'$, where $r' \geq 0$.
Since $r \in S$, $a = dq_0 + r$ (for some q_0) $= dq_0 + d + r' = d(q_0 + 1) + r'$. $\Rightarrow r' \in S$.
Thus $r - r' = d > 0$ ∴ $r' < r$, contradicting the assumption that r is the smallest element. ∴ $r \geq d$ is false, thus $r < d$.

Thus the uniqueness of r and q is proved.

Recursive Definition (note: dealing only w. +ve integers and 0)

- Factorial of a non-negative integer: $fact(n) = n \cdot fact(n-1)$ and $fact(0) = 1$
  ($fact(n) = n \cdot fact(n-1)$ is the recursive step and $fact(0) = 1$ is the base case.)
- Raising a number to a non-negative integer power:
  - → Base case: $a^0 = 1$
  - → Recursive step: $a^n = a \cdot a^{n-1}$
- The n^{th} Fibonacci number:
  - → Base case: $Fibo(1) = 0$, $Fibo(2) = 1$
  - → Recursive step: $Fibo(n) = Fibo(n-1) + Fibo(n-2)$

Q: Recursively define the set of positive multiples of 3.
A: Base case - $3 \in S$
Recursive step - If $x \in S$ and $y \in S$ then $x + y \in S$

Q: Recursively define the Kleene star (Σ^*), which is the set of all strings over a given alphabet Σ.
A: Base case - $\lambda \in \Sigma^*$ (λ is the empty string)
Recursive step - If $w \in \Sigma^*$ and $x \in \Sigma$ then $wx \in \Sigma^*$ (string concatenation)

Q: Recursively define (i) string concatenation (ii) string length
A: (i) Base case - If $w \in \Sigma^*$ then $w \cdot \lambda = w$
(ii) Recursive step - If $w_1, w_2 \in \Sigma^*$ and $x \in \Sigma$ then $w_1 \cdot (w_2 \cdot x) = (w_1 \cdot w_2) \cdot x$

Extras:

(ii) Base case - $len(\lambda)=0$
Inductive step - If $w \in \Sigma^*$, $x \in \Sigma$ then $len(wx) = len(w)+1$.

Q: Recursive definition of a well-formed formula is as follows:

Base Case - Every atomic formula (a single proposition or variable like p, q, r) is a well-formed formula (WFF).

Recursive Step - If A and B are WFFs, then $\sim A$ (negation), $A \wedge B$ (conjunction), $A \vee B$ (disjunction), $A \to B$ (implication) and $A \leftrightarrow B$ (biconditional) are also WFFs. Parentheses must be used properly, for clarity.

Closure - Nothing else is a WFF except those obtained by following the above rules.

Show that the following are not WFFs: (i) $\sim$ (ii) $p \wedge$ (iii) $p \wedge \sim q$ (iv) $p \to \sim \forall q$ (v) $p \vee q \wedge r$

A: (i) Incomplete negation, No formula after $\sim$.
(ii) Incomplete conjunction, missing second operand.
(iii) & (iv) Invalid combination of operands, no clear structure.
(v) Missing parentheses for clarity.

Structural Induction

- It is very similar to the first principle of induction, but it is applied in different areas.

Q: A set S is defined as follows: $3 \in S$
If $x \in S$ and $y \in S$ then $x+y \in S$

and $A = \{3n : n \in \mathbb{N}\}$. Show that $S=A$. ($\mathbb{N} = \{1, 2, 3, \ldots\}$)

A: To prove that $S=A$, we need to prove that A and S are subsets of each other.

$A \subseteq S$: Base case: $3 \in A$ and $3 \in S$ (By definition).
~~Inductive hypothesis: $x, y \in A \to x, y \in S$~~
~~Since $x, y \in A$, let $x = 3k_1$, $y = 3k_2$ for some $k_1, k_2 \in \mathbb{N}$.~~
~~As $x, y \in S$, $x+y \in S \Rightarrow 3(k_1+k_2) \in S$~~

Inductive hypothesis: $x \in A \to x \in S$. $\therefore$ Let $x = 3k$, $k \in \mathbb{N}$.
By the definition, if $x \in S$ then $x+3 \in S$ ($\because 3 \in S$).

Note that if A and S are defined exactly the same (i.e. have the same def^n.), then they are equal.
From the definition of S, we can write that if $x \in S$, then $x+3 \in S$. ($\because 3 \in S$). Now,

$\because$ $3 \times 1 \in S$, $3 \in S \wedge (3 \in S \to 3+3 \in S) \Rightarrow 3+3 \in S$ $\therefore 6 \in S$ ($3 \times 2 \in S$)

$\because$ $3 \times 2 \in S$, $6 \in S \wedge (6 \in S \to 6+3 \in S) \Rightarrow 6+3 \in S$ $\therefore 9 \in S$ ($3 \times 3 \in S$)

⋮ we can use modus ponen as many times as reqd.

$3k \in S \wedge (3k \in S \to 3(k+1) \in S) \Rightarrow 3k+3 \in S \therefore 3(k+1) \in S$ (For some k)

But, by the definition of $\mathbb{N}$ ($1 \in \mathbb{N} \wedge (n \in \mathbb{N} \to n+1 \in \mathbb{N})$), we can conclude that the definition of S ($3 \times 1 \in S \wedge (3k \in S \to 3(k+1) \in S)$) includes every number which is thrice of a natural number, and no other number.
$\therefore$ The definition of S may be stated as $\{3k : k \in \mathbb{N}\}$, which is the definition of A.
$\therefore$ $S=A$, Hence proved.

Q: Write the recursive definition of a rooted tree.

A: Base Case: If r is a single node, it is a rooted tree.
Recursive Step: If $T_1, T_2 \dots T_k$ are rooted trees with roots $r_1, r_2, \dots r_k$, then a new graph, where a new node r is connected to all trees T_i by their roots r_i, is also a rooted tree.

Q: Write the recursive definition of a full binary tree.

A: Base Case: A single vertex is a full binary tree.
Recursive Step: If T_1 and T_2 are full binary trees, then a new graph where T_1 and T_2 are children trees to a new root r is also a binary tree.

Q: Recursive def^n^s. of the height $h(T)$ and number of nodes $n(T)$ of a full binary tree are as follows:

Base Case: $n(T) = 1$ and $h(T) = 0$ for a single vertex.
Recursive Step: If T_1 and T_2 have respectively $n(T_1)$ and $n(T_2)$ nodes, and heights $h(T_1)$ and $h(T_2)$, then $n(T) = 1 + n(T_1) + n(T_2)$ and $h(T) = 1 + \max(h(T_1), h(T_2))$ (T defined as above)

Prove by induction that $n(T) \leq 2^{h(T)+1} - 1$.

A: Base case: For a single vertex, $n(T) = 1$ and $h(T) = 0$, so $1 \leq 1$, which is true.
Inductive step: Let this be true for two full binary trees T_1 and T_2, i.e.

$$n(T_1) \leq 2 \cdot 2^{h(T_1)} - 1, \quad n(T_2) \leq 2 \cdot 2^{h(T_2)} - 1.$$

Now, for a new tree T', where a root node r has children T_1 and T_2,

$$n(T') = 1 + n(T_1) + n(T_2), \quad h(T') = 1 + \max(h(T_1), h(T_2))$$

$$\therefore n(T') = 1 + n(T_1) + n(T_2) \leq 1 + 2 \cdot 2^{h(T_1)} + 2 \cdot 2^{h(T_2)} - 2$$
$$= 2 \cdot 2^{h(T_2)} + 2 \cdot 2^{h(T_1)} - 1$$
$$\leq 2 \cdot 2^{\max(h(T_1), h(T_2))} + 2 \cdot 2^{\max(h(T_1), h(T_2))} - 1$$
$$= 2^{2 + \max(h(T_1), h(T_2))} - 1 = 2^{1 + h(T')} - 1$$

hence the statement is proved by mathematical induction.

Extras:

Combinatorics

- **Product Rule** - If a task is broken down into k subtasks $T_1, T_2, \dots T_k$, where T_1 can be done in n_1 ways, T_2 can be done in n_2 ways, and so on, then the number of ways to perform the k subtasks in order, and complete the task, is $\prod_{i=1}^{k} n_i$. (Multiplication is represented by the keyword AND.)

- **Sum Rule** - If a task can be done in n_1 ways or n_2 ways or... n_k ways, then the number of ways to perform the task is $\sum_{i=1}^{k} n_i$. (Addition is represented by 'OR').

Q (i) How many bit strings of length k exist?
(ii) From a domain containing m elements to a codomain containing n elements, how many functions can be made? What about injective functions?
(iii) How many subsets does a set containing n elements have?

A: (i) Each bit can be either 0 or 1, and we have to fill in k such bits. $\therefore (1+1)^k = 2^k$.

(ii) For functions, each element in the domain has n choices, and there are m elements in the domain. $\therefore$ No. of functions is n^m.
For one-one functions, $n \geq m$. So it's 0 if $m > n$.
If $m \leq n$, the first element has n choices, the second has (n-1) choices, the third has (n-2) choices, ... the m^{th} has (n-m+1) choices.
$\therefore$ No. of one-one (injective) functions $= n(n-1)(n-2)\dots(n-m+1) = \frac{n!}{(n-m)!} = {}^nP_m$

(iii) Each of the n elements can be either in the subset or out of it.
$\therefore$ Total no. of subsets $= (1+1)^n = 2^n$.

Q: How many passwords can be created such that
→ every character is either an uppercase letter, a lowercase letter or a digit
→ its length is atleast 6 and atmost 8
→ there should be atleast 1 digit and 1 uppercase letter?

A: Idea The total number of ways is $P_6 + P_7 + P_8$, where P_i denotes the number of permissible passwords of length i.
Idea 1: Select one position for an uppercase letter, another one for a digit and fill the remaining places arbitrarily.
$\therefore P_6 = (6 \times 26) \times (5 \times 10) \times 62^4$, $P_7 = (7 \times 26) \times (6 \times 10) \times 62^5$,
$P_8 = (8 \times 26) \times (7 \times 10) \times 62^6$.

Idea 2 (Principle of Mathematical Inclusion-Exclusion): From all possibilities, exclude all the invalid ones, then add back the ones subtracted twice by mistake.

$\therefore P_6 = 62^6 - 10^6 - 52^6 - 36^6 + 10^6 + 26^6 = 62^6 - 52^6 - 36^6 + 26^6$
= Total pwds. - Pwds. with only digits - Pwds. with only letters - Pwds. with only digits or lowercase letters + Pwds. with only digits (subtracted twice) + Pwds. with only lowercase letters (subtracted twice).

$P_7 = 62^7 - 52^7 - 36^7 + 26^7$
$P_8 = 62^8 - 52^8 - 36^8 + 26^8$

Remark: The addition and multiplication rules of counting may be represented as a series of for loops, as follows:

Addition Rule

```
n=0
for i in range(k1):
    n=n+1
for i in range(k2):
    n=n+1
    ⋮
for i in range(km):
    n=n+1
# At the end of this program, the
# value of n will be k1+k2+... +km.
# Only one for loop runs at a time.
```

Multiplication Rule

```
n=0
for i in range(k1):
    for i in range(k2):
        ⋮
            for i in range(km):
                n=n+1
# At the end of this program, the value of
# n will be k1*k2*... *km. All for loops
# run simultaneously (they are nested in layers).
```

Q: Classes A, B and C are three classes of IPv4 addresses, which are represented by a string of 32 bits.

- → Class A: The first bit is 0, followed by a 7-bit network part and a 24-bit host part.
- → Class B: The first two bits are 10, followed by a 14-bit network part and a 16-bit host part.
- → Class C: The first three bits are 110, followed by a 21-bit network part and an 8-bit host part.

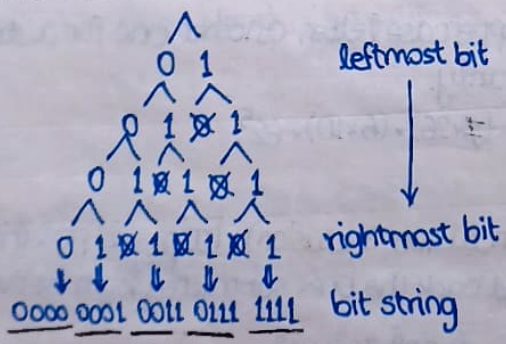
None of the host parts may contain all 0s or all 1s. For class A, the network part cannot be all 1s. How many IPv4 address can be made belonging to these classes?

A: $(2^7-1)(2^{24}-2)+(2^{14})(2^{16}-2)+(2^{21})(2^8-2)$.
(class A, class B, class C)

. Tree Diagram: It is a representation used to list out all possibilities. This can be used to obtain a count of the possibilities.

Q: How many 4-bit strings can be made, such that a bit can be 0 only if any bits to its left are also 0s?

A:



By making a tree diagram as shown, we can count 5 possible bit strings satisfying the given cond^n.

Q: How many 8-bit strings either begin with a 1 or end with a 00?

A: Using the principle of mathematical inclusion & exclusion. (see prev. pg.)

No. of strings = Strings that begin with a 1 + Strings that end with a 00
− Strings that both begin with a 1 and end with a 00 (counted 2x)
$= 2^7+2^6-2^5 = 160.$

Pigeonhole Principle

- This principle states that if $k+1$ pigeons fly into k pigeonholes, there must necessarily be a pigeonhole containing two or more pigeons.
- It is also known as Dirichlet's drawer principle.

Q: Show that for every positive integer n, one can find a positive multiple of n made up wholly of 0s and 1s.

A: Consider the set of $(n+1)$ numbers $\{1, 11, 111, \ldots 111\cdots1\}$; where the k^{th} element $(1 \le k \le n+1)$ is made up of k 1s. Let's call this set A.

Recall that any integer when divided by n leaves a remainder belonging to the set $\{0, 1, 2, \ldots n-1\}$ (see pg. 6). Let's call this set B.

∴ By the pigeonhole principle, there exist atleast 2 distinct numbers belonging to A that leave the same remainder (which belongs to B) when divided by n. Let's call this remainder r, $(0 \le r < n)$ and the two numbers x and y $(x > y)$.

∴ By Euclid's division algorithm, $x = k_1 n + r$

$y = k_2 n + r$ (where $k_1 > k_2$)

∴ $x - y = (k_1 - k_2)n$ is exactly divisible by n, and by the nature of x and y being solely composed of 1s, the subtraction happens without borrowing, and $x-y$ is entirely made up of 1s and 0s.

Hence proved.

```
 1 1 1 . . 1 . . 1 1 1 1   (x)
-       . . 1 . . 1 1 1 1   (y)
-------------------------
 1 1 1 . . 0 . . 0 0 0 0   (x-y)
```

Extras:

- The pigeonhole principle can alternatively be stated as: if N objects are placed into K boxes, there exists atleast one box containing atleast $\text{ceil}(N/K)$ objects. (ceil(▢) = the smallest integer ≥ ▢)

Proof: Assume, on the contrary, that every box has fewer than $\text{ceil}(\frac{N}{K})$ objects. Now, observe that

$$\text{if } \frac{N}{K} \in \mathbb{Z} \Rightarrow \text{ceil}(\tfrac{N}{K}) = \tfrac{N}{K} \Rightarrow \text{ceil}(\tfrac{N}{K}) < \tfrac{N}{K} + 1$$

$$\text{if } \frac{N}{K} \notin \mathbb{Z} \Rightarrow \text{ceil}(\tfrac{N}{K}) = \tfrac{N}{K} + f \ (0<f<1) \Rightarrow \text{ceil}(\tfrac{N}{K}) < \tfrac{N}{K} + 1$$

Hence, the total number of objects in all K boxes is less than or equal to $K(\text{ceil}(\frac{N}{K})-1) < K(\frac{N}{K}) = N$ (i.e. always less than N), a contradiction, since the total number of objects must be equal to N. ∴ The given statement is proved.

Q: What is the minimum value of N, such that $\text{ceil}(\frac{N}{K}) \geq r$? Assume $r \in \mathbb{Z}$.

A: $\text{ceil}(\frac{N}{K}) \geq r \Rightarrow r-1 < \frac{N}{K} \Rightarrow N > K(r-1)$ ∴ Minimum value = $K(r-1)+1$.

Q: A standard deck of 52 cards has 4 suits (♡, ♠, ◇, ♣). How many cards should be chosen such that there are atleast 3 cards of the same suit?

A: $\text{ceil}(\frac{N}{4}) \geq 3 \Rightarrow N \geq 4(3-1)+1$ ∴ atleast 9 cards should be chosen.

Q: An Aadhaar card number has 12 digits, and the leading number(digit) is nonzero. Assume the population of India to be 1.5 billion.

(i) How many Aadhaar card numbers are possible?

(ii) NXXX XXXX XXXX $\quad N, X \in \{0, \ldots 9\}, N \neq 0$

Suppose the GoI decides to permit only certain values of NXXX (leading 4 digits), out of the available 9000. How many such values should they permit, such that each Indian citizen still gets a unique Aadhaar number? (at min.)

A: (i) 9×10^{11} (ii) ~~ceil(1.5B/9000)~~ For 1 value of NXXX, there are 10^8 nos.

∴ The least value of p such that $p \cdot 10^8 \geq 1.5\,B$ is $p = 15$.

Q: Show that among any n+1 positive integers, none of which are greater than 2n, one must be a multiple of another.

A: ~~We note that any positive integer may be written as $2^k \cdot q$, where k is non-negative and q is odd.~~

~~∴ Consider n numbers $\{2^{k_1}q_1, 2^{k_2}q_2, \ldots 2^{k_{n+1}}q_{n+1}\}$ (There are n+1 of them)~~

Let us try to select n+1 positive integers such that none of them is a multiple of another.

We note that it is possible to select n such integers, viz. $\{n, n+1, \ldots 2n-1\}$.

Now we can't select the $(n+1)^{th}$ number to be 2n, because 2n is a multiple of n. We can't select it to be from $\{1, 2, \ldots n-1\}$ either, because ∃ a multiple of it in the n integers chosen earlier.

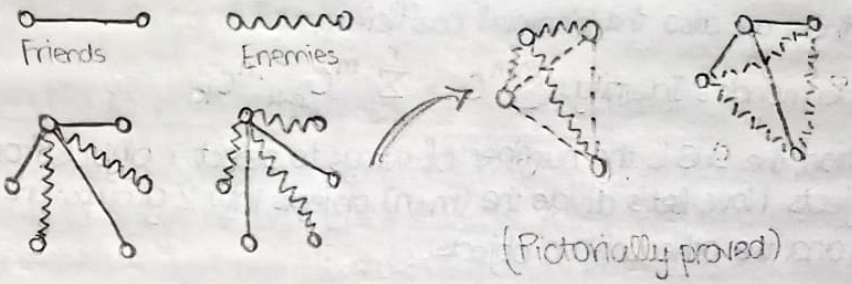
> Proof (by contradiction): Let ∃ $k \in [1, n)$ such that none of $\{n, n+1, \ldots 2n-1\}$ are divisible by k. Consider the largest multiple of k (<n) to be m_1 and the smallest multiple of k (≥2n) to be m_2. They clearly must be consecutive multiples of k. But $m_2 - m_1 > 2n - n = n > k$, a contradiction. So no such k exists.

This completes the proof.

Theorem: Any sequence of n^2+1 real numbers must have a subsequence of atleast $n+1$ reals, which is either strictly increasing or strictly decreasing.

Q: Assume that there is a group of 6 people, where every pair of them are either mutual friends or mutual enemies. Show that there are three people, who are either a group of friends or a group of enemies.

A:



Permutations and Combinations

- ${}^nP_r$ (r-permutation): An ordered arrangement of r out of n objects can be made in these many ways. It equals $\frac{n!}{(n-r)!}$.
- ${}^nC_r$ (r-combination): Also represented by $\binom{n}{r}$. it is the number of ways to select (unordered) r out of n objects, which equals $\frac{n!}{(n-r)!\,r!}$. (aka Binomial Coefficient)
- ${}^nP_r = {}^nC_r \cdot r!$
- ${}^nC_r = {}^nC_{n-r}$
- Note: Defined only if $0 \le r \le n$. Obv. $n \ge 0$

Q: How many n-bit strings contain exactly r 1s?

A: Choose r places for 1s, the remaining will automatically be 0s. $\therefore {}^nC_r$.

Q: How many ways are there to select a 5-member committee from 25 men and 22 women, containing no less than 2 men and no less than 2 women?

A: Choose 2 men AND choose 2 women AND choose another person.
$\therefore {}^{25}C_2 \cdot {}^{22}C_2 \cdot {}^{43}C_1$.

Q: What is the coefficient of $x^{12}y^{13}$ in the expansion of $(2x-3y)^{25}$?

A: Recall: (for $n>0$) $(X+Y)^n = \sum_{i=0}^{n} {}^nC_i \cdot X^i \cdot Y^{n-i}$. $\therefore$ Reqd value $= -{}^{25}C_{12} \cdot 2^{12} \cdot 3^{13}$.

- $\sum_{k=0}^{n} {}^nC_k = 2^n$ (Put $X=Y=1$)
- $\sum_{k=0}^{n} (-1)^k \cdot {}^nC_k = 0$ (Put $X=-1, Y=1$)
- $\sum_{k=0}^{n} 2^k \cdot {}^nC_k = 3^n$ (Put $X=1, Y=2$)

Q: Prove Pascal's identity: ${}^{n+1}C_k = {}^nC_{k-1} + {}^nC_k$. ($n \ge k > 0$, both $\mathbb{Z}^+$)

A: $\text{RHS} = \frac{n!}{(n-k+1)!\,(k-1)!} + \frac{n!}{(n-k)!\,k!} = \frac{n!}{(n-k)!\,(k-1)!}\left(\frac{1}{n-k+1} + \frac{1}{k}\right)$

$= \frac{n!}{(n-k)!\,(k-1)!}\left(\frac{n+1}{(n-k+1)k}\right) = \frac{(n+1)!}{(n-k+1)!\,k!} = {}^{n+1}C_k = \text{LHS}$. Hence proved.

Remark: Used to generate the Pascal's triangle, where each number is the sum of the two numbers above it. The leftmost and rightmost edges are all 1s.

Extras:

. The Pascal's triangle is

$$\begin{array}{ccccccc} {}^0C_0 & & & & & & \\ {}^1C_0 & {}^1C_1 & & & & & \\ {}^2C_0 & {}^2C_1 & {}^2C_2 & & & & \\ {}^3C_0 & {}^3C_1 & {}^3C_2 & {}^3C_3 & & & \\ \vdots & & & & & & \end{array} = \begin{array}{ccccccc} & & & 1 & & & \\ & & 1 & & 1 & & \\ & 1 & & 2 & & 1 & \\ 1 & & 3 & & 3 & & 1 \\ & 1 & & 4 & & 6 & \\ & & \ddots & & & & \end{array}$$

Note that these are also the binomial coefficients.

Q: Prove Vandermonde's Identity: ${}^{m+n}C_r = \sum_{k=0}^{r} {}^mC_{r-k} \cdot {}^nC_k$.

A: By definition, the LHS is the number of ways to select r objects from (m+n) objects. Now, let's divide the (m+n) objects into 2 groups: one with m objects and the other with n objects.

To select r objects, we can either

choose 0 objects out of m AND r out of n
OR choose 1 object out of m AND (r-1) out of n
OR choose 2 objects out of m AND (r-2) out of n
⋮
OR choose r objects out of m AND 0 out of n.

∴ Total no. of ways = ${}^mC_0 \cdot {}^nC_r + {}^mC_1 \cdot {}^nC_{r-1} + {}^mC_2 \cdot {}^nC_{r-2} + \dots + {}^mC_r \cdot {}^nC_0$ = RHS.
Hence proved.

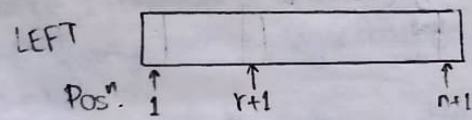
Note: ~~If r>m or r>n, the corresponding terms~~ Only defined if $r \le m$ & $r \le n$.

Q: Prove that $\sum_{k=0}^{n} ({}^nC_k)^2 = {}^{2n}C_n$.

A: Using Vandermonde's identity, ${}^{n+n}C_n = \sum_{k=0}^{n} {}^nC_{n-k} \cdot {}^nC_k \Rightarrow {}^{2n}C_n = \sum_{k=0}^{n} ({}^nC_k)^2$.
($\because {}^nC_{n-k} = {}^nC_k$). Hence proved.

Q: Prove that ${}^{n+1}C_{r+1} = \sum_{j=r}^{n} {}^jC_r$.

A: We can consider the LHS to be the number of (n+1) bit strings containing (r+1) 1s. Let's number the bits 1, 2, 3, ... (n+1) from left to right.



The position of the rightmost 1 can be (r+1), (r+2), ... (n+1). So LHS can also be written as the sum of possibilities of choosing r ~~objs~~ pos^{n}s. from those left of the rightmost bit.

∴ LHS = $\sum_{k=r+1}^{n+1} {}^{k-1}C_r$ (k = pos^n of rightmost 1) = $\sum_{j=r}^{n} {}^jC_r$ (j = k-1). Hence proved.

Extras:

Generalised PnC

- n^r: r-permutation of n objects with repetition
- ${}^{n+r-1}C_r$: r-combination of n objects with repetion
  = No. of ways to distribute r identical objects into n distinguishable objects
- $\frac{n!}{\prod_{i=1}^{k} n_i!}$: No. of permutations of n objects where n_1 are of type 1, n_2 are of type 2, ... n_k are of type k ($\Sigma n_i = n$)
- $P_k(n)$: No. of ways to distribute n identical objects into k identical boxes

Q: There are 4 students A,B,C,D and 3 identical tasks T_1, T_2, T_3. Find the number of ways to assign the tasks to the students. ~~A task may be performed by exactly one student, but one~~ No student must be jobless.

A: $x_1+x_2+x_3=4$, $0 \le x_i \le 4$. $\therefore (0,0,4), (0,1,3), (0,2,2), (1,1,2)$

$(0,0,4) \Rightarrow {}^4C_0 \cdot {}^4C_0 \cdot {}^4C_4 = 1$ 3 ways

$(0,1,3) \Rightarrow {}^4C_0 \cdot {}^4C_1 \cdot {}^3C_3$ 6 ways

$(0,2,2) \Rightarrow$ 3 ways

$(1,1,2) \Rightarrow$ 3 ways

$(T_1, T_2, T_3) = (0,0,4)$ or $(0,4,0)$ or $(4,0,0)$

Same as distributing 4 distinguishable objects into 3 identical boxes (order of arrangement of boxes after distribution is immaterial)

T_1	T_2	T_3
ABCD		
ABC	D	
BCD	A	
CDA	B	
DAB	C	
AB	CD	
BC	DA	
AC	BD	
AB	C	D
BC	D	A
CD	A	B
DA	B	C
AC	B	D
BD	A	C

$\therefore$ There are 14 ways to assign the tasks.

Solving Recurrences

- Recall that a recurrence relation is one in which the $(n+1)^{th}$ term (a_n) is expressed in terms of the previous n terms $(a_0, a_1, \dots a_n)$

Eg: Tower of Hanoi: $H_n = 2H_{n-1}+1$, $H_1 = 3$; the general term is $H_n = 2^n - 1$.

Eg: No. of bit strings not ending with 00: a_2 (No. of strings of length 2) = 3,
$a_n = 2^{n-1} a_{n-1}$.

Extras:

- Eg: Catalan number: $C_n = \sum_{k=0}^{n-1} C_k C_{n-k-1} = \frac{2nC_n}{n+1}$ $(C_0 = 1)$

→ Parentheses Combinations: The number of ways to correctly match n pairs of parentheses. [$C_2 \Rightarrow (())$, $()()$ $C_3 \Rightarrow ((()))$, $(()())$, $(())()$, $()(())$, $()()()$]

→ Triangulation of a Polygon: The number of ways to divide a convex polygon of $(n+2)$ sides into n triangles, using non-intersecting diagonals.

(i) Linear Homogeneous Recurrence Relation of Degree k (with constant coefficients):

- $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k} = \sum_{i=1}^{k} c_i a_{n-i}$, $c_k \neq 0$ is the form of such an eqn.
- Assume $a_n = r^n$ to be a solution for some $r \in \mathbb{R}$. Then, $(r \neq 0)$

$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \dots + c_k r^{n-k} \Rightarrow r^k - c_1 r^{k-1} - c_2 r^{k-2} - \dots - c_k = 0$ (dividing by r^{n-k}), where this equation is called the characteristic equation of the recurrence relation.

- Case 1: $k=2$: The characteristic equation is $r^2 - c_1 r - c_2 = 0$, with roots r_1 and r_2.
  - → Case 1a: $r_1 \neq r_2$, then the solution is of the form $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$.
  - → Case 1b: $r_1 = r_2 = R$, then the solution is of the form $a_n = (\alpha_1 + \alpha_2 n) R^n$.

Q: Solve the recurrence relation $a_n = a_{n-1} + 2a_{n-2}$, given that $a_0 = 2, a_1 = 7$.

A: The characteristic equation is $r^2 - r - 2 = 0$, with roots $r_1 = -1, r_2 = 2$.

$\therefore a_n = \alpha_1(-1)^n + \alpha_2(2)^n$. Given a_0 and a_1, we can write

$\alpha_1 + \alpha_2 = 2$
$-\alpha_1 + 2\alpha_2 = 7$ $\quad \therefore \alpha_1 = -1, \alpha_2 = 3. \quad \therefore a_n = (-1)^{n+1} + 3 \cdot 2^n$.

Q: Solve the recurrence relation $a_n = 6a_{n-1} - 9a_{n-2}$, given that $a_0 = 1, a_1 = 6$.

A: The characteristic equation is $r^2 - 6r + 9 = 0$, with roots $r_1 = r_2 = 3$.

$\therefore a_n = (\alpha_1 + n\alpha_2)3^n$. Given a_0 and a_1,

$a_0 = \alpha_1 = 1$
$a_1 = 3(\alpha_1 + \alpha_2) = 6$ $\quad \therefore \alpha_1 = \alpha_2 = 1 \quad \therefore a_n = 3^n(1+n)$.

- Case 2: $k \geq 3$: The characteristic equation is ~~$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \dots + \alpha_k$~~ a monic degree k poly^{nl}. of
  - → Case 2a: All r_i's are distinct, then $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \dots + \alpha_k r_k^n$.
  - → Case 2b: The roots are r_1 of multiplicity m_1, r_2 of multiplicity m_2, ... r_t of multiplicity m_t. Note that $m_1 + m_2 + \dots + m_t = k$.

$\therefore a_n = (\alpha_{10} + \alpha_{11} n + \dots \alpha_{1(m_1-1)} n^{m_1-1}) r_1^n + (\alpha_{20} + \alpha_{21} n + \dots \alpha_{2(m_2-1)} n^{m_2-1}) r_2^n + \dots$
$+ (\alpha_{t0} + \alpha_{t1} + \dots + \alpha_{t(m_t-1)} n^{m_t-1}) r_t^n$.

Q: Solve the recurrence relation $a_n = 6a_{n-1} - 11a_{n-2} + 6a_{n-3}$, $a_0 = 2, a_1 = 5, a_2 = 15$.

A: The characteristic equation $(r^3 - 6r^2 + 11r - 6 = 0)$ has roots $r_1 = 1, r_2 = 2, r_3 = 3$.

Hence $a_n = \alpha_1 + \alpha_2 \cdot 2^n + \alpha_3 \cdot 3^n$. Given

$a_0 = \alpha_1 + \alpha_2 + \alpha_3 = 2$
$a_1 = \alpha_1 + 2\alpha_2 + 3\alpha_3 = 5$
$a_2 = \alpha_1 + 4\alpha_2 + 9\alpha_3 = 15$ $\quad \therefore \alpha_1 = 1, \alpha_2 = -1, \alpha_3 = 2$. Thus $a_n = 2 \cdot 3^n - 2^n + 1$.

Extras:

→ Lattice Paths: C_n is the number of lattice paths along a grid from $(0,0)$ to (n,n), such that each step goes either → or ↑, and the path doesn't cross the diagonal $y = x$.

Q: Solve the recurrence relation $a_n = -3a_{n-1} - 3a_{n-2} - a_{n-3}$, given that $a_0 = 1$, $a_1 = -2$ and $a_2 = -1$.

A: The characteristic equation $((r+1)^3 = 0)$ has roots $r_1 = r_2 = r_3 = -1$.

$\therefore a_n = (\alpha_{10} + \alpha_{11} n + \alpha_{12} n^2)(-1)^n$. Given

$a_0 = \alpha_{10} = 1$

$a_1 = -\alpha_{10} + \alpha_{11} - \alpha_{12} = -2$

$a_2 = \alpha_{10} + 2\alpha_{11} + 4\alpha_{12} = -1$ $\quad \therefore \alpha_{10} = 1, \alpha_{11} = 3, \alpha_{12} = -2.$

Thus $a_n = (1 + 3n - 2n^2)(-1)^n$.

(ii) Linear Non-Homogeneous Recurrence Relation (with constant coefficients):

- The form of such an equation is $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \dots + c_k a_{n-k} + F(n)$, where $F(n)$ is wholly dependent on n and not on any previous terms.
- Its solution is $a_n = a_n^{(h)} + a_n^{(p)}$, where
  - → $a_n^{(h)}$: the solution of the associated homogeneous part
  - → $a_n^{(p)}$: a particular solution to the non-homogeneous recurrence relation
- Let $F(n) = (b_t n^t + b_{t-1} n^{t-1} + \dots + b_1 n + b_0) s^n$, $b_i, S \in \mathbb{R}$.
  - → If S is not a root of the homogeneous part, then $a_n^{(p)} = (p_t n^t + p_{t-1} n^{t-1} + \dots + p_1 n + p_0) s^n$, $p_i \in \mathbb{R}$
  - → If S is a root of the homogeneous part, then $a_n^{(p)} = n^m (p_t n^t + p_{t-1} n^{t-1} + \dots + p_1 n + p_0) s^n$, $p_i, m \in \mathbb{R}$.

Q: Solve the recurrence relation $a_n = 3a_{n-1} + 2n$, given that $a_1 = 3$.

A: The characteristic equation of the homogeneous part is $r - 3 = 0$, thus $a_n^{(h)} = \alpha \cdot 3^n$.

Now since $F(n) = 2n = (2n+0)1^n$, and 1 is not a root of the homogeneous part, thus $a_n^{(p)} = (p_1 n + p_0)1^n = p_1 n + p_0$.

Substituting $a_n^{(p)}$ back into the recurrence relation,

$p_1 n + p_0 = 3(p_1(n-1) + p_0) + 2n = (3p_1 + 2)n + (3p_0 - 3p_1)$. So, $p_1 = 1, p_0 = -\frac{3}{2}$.

Thus $a_n = \alpha \cdot 3^n - n - \frac{3}{2}$. Given $a_1 = 3\alpha - \frac{5}{2} = 3$, thus $\alpha = \frac{11}{6}$. $\therefore a_n = (\frac{11}{6})3^n - n - \frac{3}{2}$.

Q: Solve the recurrence relation $a_n = 5a_{n-1} + 6a_{n-2} + 7^n$, in terms of some parameters.

A: The char. eq. of homog. part is $r^2 - 5r - 6 = 0$. $\therefore a_n^{(h)} = \alpha_1(-1)^n + \alpha_2(6)^n$.

And $a_n^{(p)} = p_0(7)^n$, $\because 7$ is not a root of ↑.

Substituting $a_n^{(p)}$ back, $p_0(7)^n = 5p_0(7)^{n-1} + 6p_0(7)^{n-2} + 7^n$

$\Rightarrow 49p_0 = 35p_0 + 6p_0 + 49 \quad \therefore p_0 = 49/8$.

$\therefore a_n = \alpha_1(-1)^n + \alpha_2(6)^n + \frac{49}{8}(7)^n$, where α_1 and α_2 are arbitrary constants.

Extras:

Master's Theorem

- The Master's Theorem is applicable to recurrence relations of the form

$$T(n) = aT\left(\frac{n}{b}\right) + F(n)$$ ~ polynomial

Where $T(n)$ denotes the n^{th} term, $F(n)$ is an arbitrary function and b is a divisor of n. Also $a \geq 1$, $b > 1$.

Eg: Binary Search - The time complexity $T(n) = T\left(\frac{n}{2}\right) + c$
Merge Sort - The time complexity $T(n) = 2T\left(\frac{n}{2}\right) + cn$

Asymptotic Notation: This notation refers only to approximate values of functions, and not their actual values. Eg: O (Big-O), Ω (Big Omega), Θ (Big Theta), and so on. This notation only focuses on the terms impacting the growth of the f^n.

~~Eg: $a_n = 5n^2 + 10n + 50 = 5n^2\left(1 + \frac{2}{n} + \frac{10}{n^2}\right)$, $n \in \mathbb{N}$.~~
~~As n grows larger and larger, $a_n \approx n^2$ (approx.). Hence, $a_n = O(n^2)$.~~

O - an upper bound: if $f(n) \leq c_1 g_1(n)$ $\forall n > n_1$, $n_1, c_1 \in \mathbb{R}$, then $f(n) = O(g_1(n))$
Ω - a lower bound: if $f(n) \geq c_2 g_2(n)$ $\forall n > n_2$, $n_2, c_2 \in \mathbb{R}$, then $f(n) = \Omega(g_2(n))$
Θ - a tight bound: if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ then $f(n) = \Theta(g(n))$.
This Θ gives the actual growth rate of the function f.

- If $F(n) = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$. - (i)
- If $F(n) = \Theta(n^{\log_b a})$ then $T(n) = \Theta(n^{\log_b a} \log n)$. - (ii)
- If $F(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$, and $aF(n/b) \leq cF(n)$ for some $c > 1$, then $T(n) = \Theta(F(n))$. - (iii)

Q: Write the asymptotic tight bound notation for (i) $T(n) = 9T\left(\frac{n}{3}\right) + n$ (ii) $T(n) = T\left(\frac{2n}{3}\right) + 1$

A: (i) $F(n) = n$, $n^{\log_b a} = n^2$. Since $F(n) = O(n^{2-1})$, thus (from (i))
$T(n) = \Theta(n^2)$.

(ii) $F(n) = 1$, $n^{\log_b a} = n^0 = 1$. Since $F(n) = O(n^0)$, thus (from (ii)) $T(n) = \Theta(\log n)$.

Q: Θ notation for $T(n) = 3T\left(\frac{n}{4}\right) + n \log n$?

A: $F(n) = n \log n$, ~~$n^{\log_b a} = n^{\log_4 3}$~~
~~is not a polynomial~~
~~∴ can't~~

$n^{\log_b a} = n^{\log_4 3}$
Since $F(n) = \Omega(n^{\log_4 3 + \epsilon})$ because $n \log n$ grows faster than $n^{\log_4 3}$ for sufficiently large n, $T(n)$ is $\Theta(n \log n)$, from (iii).

Q: Use Master's Theorem to write the Big θ notation for

(i) $T(n) = \sqrt{2}T(n/2) + \log n$

(ii) $T(n) = \begin{cases} 2T(\lceil \sqrt{n} \rceil) + 1, & n > 2 \\ 2, & n \leq 2 \end{cases}$

(iii) $T(2^k) = 3T(2^{k-1}) + 1$

(iv) $T(n) = 2T\left(\frac{n}{2}\right) + n \log n$

Extras:

A: (i) $F(n)=\log n$, $n^{\log_b a}=n^{1/2}$. ~~As log n grows slower~~ $k=\sqrt{n}$.
Since $\log n$ grows slower than $\sqrt{n}$, by (i), $F(n)=\Theta(\sqrt{n})$.

(ii) Let $n=2^k$ for some k. Define $S(k)=T(2^k)$, where S is another sequence.

$\therefore$ We have, $S(k)=\begin{cases} 2S(k/2)+1, & k\geq 1 \\ 2, & k<1 \end{cases}$

$\therefore F(k)=1$, $k^{\log_b a}=k$. Since k grows faster than 1, by (i), $S(k)=\Theta(k)$.

$\therefore T(2^k)=\Theta(k) \Rightarrow T(n)=\Theta(\log n)$. (No need to specify base)

(iii) Let $2^k=n$ for some n. ~~Define S(k)=~~ Thus $T(n)=3T(\frac{n}{2})+1$.

$\therefore F(k)=1$, $n^{\log_b a}=n^{\log_2 3}$. As $n^{\log_2 3}$ grows faster than 1, $T(n)=\Theta(n^{\log_2 3})$,
i.e. $T(2^k)=\Theta(3^k)$.

(iv) ~~Cannot be solved, because F(n)=n log n is not a polynomial.~~
$F(n)=n\log n$, $n^{\log_b a}=n^{\log_2 2}=n$. Since $n\log n=\Omega(n)$, $T(n)=\Theta(n\log n)$.

Q: Θ notation for (i) $f(n)=5f(\frac{n}{2})+3$ (ii) $f(n)=2f(\frac{n}{2})+2$ (iii) $f(n)=2f(\frac{n}{2})+n$
(iv) $f(n)=7f(\frac{n}{2})+\frac{15}{4}n^2$.

A: (i) $F(n)=3$, $n^{\log_b a}=n^{\log_2 5}$ (grows faster than 3). $\therefore f(n)=\Theta(n^{\log_2 5})=\Theta(5^{\log_2 n})$.

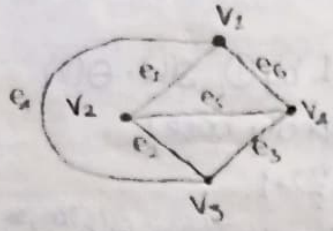
(ii) $F(n)=2$, $n^{\log_b a}=n$ (grows faster than 2). $\therefore f(n)=\Theta(n)$.

(iii) Merge Sort: $F(n)=n$, $n^{\log_b a}=n$. $\therefore f(n)=\Theta(n\log n)$.

(iv) Strassen's Formula: $F(n)=\frac{15}{4}n^2$, $n^{\log_b a}=n^{\log_2 7}$ (grows faster than $\frac{15}{4}n^2$).
$\therefore f(n)=\Theta(n^{\log_2 7})$.

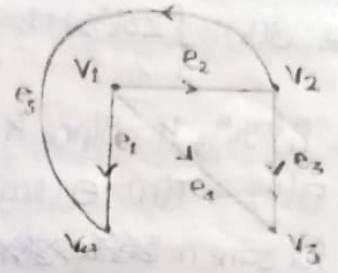
Graph Theory

- A graph can be thought of as a pictorial representation of a relation between points (vertices). Two points are related in some way if they are connected by an edge.



← here $v_1 R v_2$ AND $v_2 R v_1$ (two-way)

an undirected graph (for symmetric rel'ns)



← here $v_1 R v_2$ but NOT $v_2 R v_1$ (only one-way)

a directed graph (default assumption)

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

the incidence matrix of the undirected graph

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 & 1 \\ 0 & 0 & -1 & -1 & 0 \\ -1 & 0 & 0 & 0 & -1 \end{bmatrix} \end{array}$$

indegree: (of a vertex) no. of -1s in its row

outdegree: (of a vertex) no. of 1s in its row

the incidence matrix of the directed graph [assume no self-loops]

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

the adjacency matrix of the undirected graph

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

the adjacency matrix of the directed graph

- **Incidence Matrix:** A matrix where a_{ij} corresponds to vertex i & edge j, and a_{ij}

$$= \begin{cases} 1, & \text{if edge } e_j \text{ connects } v_i \text{ to another vertex,} \\ 0, & \text{otherwise.} \end{cases} \quad \text{(Undirected)}$$

$$= \begin{cases} 1, & \text{if edge } e_j \text{ is incident FROM vertex } v_i, \\ -1, & \text{if edge } e_j \text{ is incident ON vertex } v_i, \\ 0, & \text{otherwise.} \end{cases} \quad \text{(Directed)}$$

- **Adjacency Matrix:** A matrix where a_{ij} corresponds to vertices v_i & v_j, and a_{ij}

$$= \begin{cases} 1, & \text{if } v_i \text{ and } v_j \text{ share an edge,} \\ 0, & \text{otherwise.} \end{cases} \quad \text{(Undirected)}$$

$$= \begin{cases} 1, & \text{if there exists an edge FROM } v_i \text{ TO } v_j, \\ 0, & \text{otherwise.} \end{cases} \quad \text{(Directed)}$$

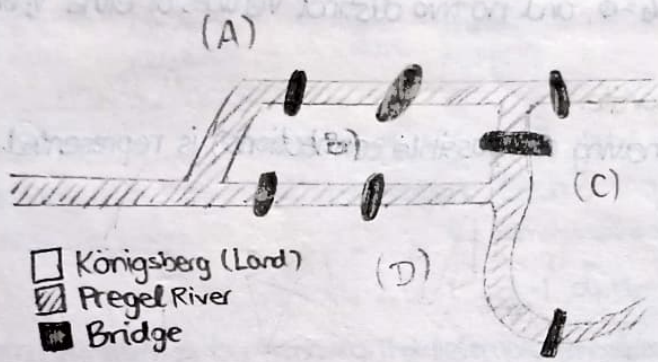
→ The adjacency matrix is a square matrix. Additionally, for an undirected graph, this matrix is symmetric about the main diagonal.

→ A 1 in the principal diagonal of the adjacency matrix represents a self-loop at the corresponding vertex.

- **Parallel Edges:** Two edges are parallel (connected between the same vertices and in the same direction) if their corresponding columns in the incidence matrix are identical.
- **Isomorphic Graphs:** Two graphs G_1 and G_2 are isomorphic if their incidence matrices M_1 and M_2 can be obtained from each other through some permutation.

Paths and Circuits - Eulerian & Hamiltonian

- **Seven Bridges of Königsberg:** A historically notable problem in mathematics, where the task was to devise a walk through the city of Königsberg, Prussia that would cross each of the seven bridges across the Pregel river once and only once.



- Leonhard Euler proved in 1736 that the problem has no soln. This is because an Eulerian path does not exist.

Eulerian Path: A path in a graph G(V,E), which traces every edge exactly once.

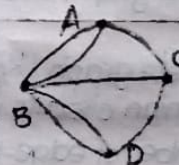
Hamiltonian Path: A path in a graph G(V,E) which visits every vertex/node exactly once.

Eulerian Circuit: A circuit in a graph G(V,E) which traces every edge exactly once, and begins and ends on the same node.

Hamiltonian Circuit: A circuit in a graph G(V,E) which visits every non-terminal vertex exactly once. It begins and ends at the same vertex.

Q: Prove that a graph can have an Eulerian path only if it has 0 or 2 odd vertices.

A: During traversal of an Eulerian path, every node along it should have edges that come in pairs. One in each pair functions as an entrance to the node, and the other, the exit. There are 2 exceptions: when we leave without entering (start) or enter without leaving (end). ∴ ∃ a path if # O.V. = 0/2. ∴ Proved.



4 odd vertices = No Eulerian path

Graph - 7 Bridges of Königsberg

- **Theorem 1:** The sum of degrees of all vertices of an undirected graph is twice the number of edges.
  - → **Proof:** The theorem can be proved by induction.
    - **Base case:** Null graph → $0 = 2 \times 0$, which is true. $(P(0))$
    - **Inductive step:** Assume that this holds true for some G(V,E). Now if a new edge is added between two vertices of V, the sum of degrees will increase by 2. $(P(k) \rightarrow P(k+1))$. Hence proved.
- **Theorem 2:** The number of odd vertices in a graph is even.
  - → **Proof:** From theorem 1, the sum of degrees of all odd vertices and degrees of all even vertices is even. But the sum of degrees of all even vertices is clearly even. Hence the sum of all degrees of odd vertices must be even, thus the theorem is implied from here.

Note: "odd vertices" = vertices of odd degree, "even vertices" = vertices of even degree

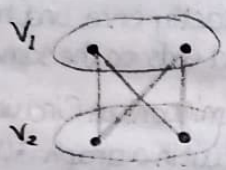
- **Simple Graph:** A graph without self-loops or parallel edges.
- **Regular Graph:** A graph where all vertices have the same degree.
- **Complete Graph:** A graph $G(V,E)$ where $\forall v_i, v_j \in V,\ i \neq j$, there exists an edge connecting v_i and v_j. It usually does not have self-loops.

K_3 - complete graph with 3 nodes

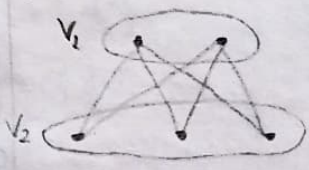
K_4

K_5

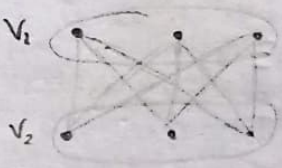
- **Null Graph:** A graph $G(V,E)$ where $E = \phi$. (No edges)
- **Bipartite Graph:** A graph $G(V,E)$ is said to be bipartite if V can be partitioned into V_1 & V_2 such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \phi$, and no two distinct vertices of either V_1 or V_2 have an edge connecting them.
  - → A null graph is trivially bipartite.
  - → A complete bipartite graph (having all possible connections) is represented as $K_{|V_1|,|V_2|}$. For example,



$K_{2,2}$



$K_{2,3}$



$K_{3,3}$

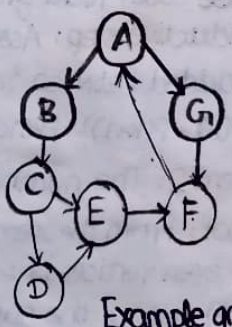
> The travelling salesperson problem involves finding a Hamiltonian cycle over a weighted graph, such that the cost is minimised.

- A complete graph (K_n) always has a Hamiltonian circuit. The number of possible Hamiltonian circuits in such a graph is $\frac{1}{2}(n-1)!$.
- The number of edge-disjoint circuits is $\frac{1}{2}(n-1)$.

<u>Graph Traversal</u> (assume graph is directed)

- There are two ways to traverse a graph:
  - → breadth-first search (BFS): Layer-by-layer traversal. It's done using a queue.
  - → depth-first search (DFS): Exploration of 1 path fully before moving to the next. It's done using a stack.
- As an example, consider the graph on the right.

Extras:



Example graph

→ **BFS:** Initialise the queue to be empty, and begin with any vertex which has neighbours (here, neighbour of a vertex V means a node connected to V by an edge that points from V to it). Enqueue this vertex. While all vertices have not been accounted for, dequeue the head and enqueue its new neighbours. Then dequeue the rest of the elements (if any) one by one.

→ **DFS:** Initialise the stack to be empty, and push any vertex which has neighbours. While all vertices have not been accounted for, pop the top off and push its new neighbours. Then pop the rest of the elements one by one.

Queue	BFS
NULL	NULL
A	NULL
BG	A
GC	AB
CF	ABG
FDE	ABGC
DE	ABGCF
E	ABGCFD
NULL	ABGCFDE

Stack	DFS
NULL	NULL
A	NULL
BG	A
BF	AG
B	AGF
C	AGFB
DE	AGFBC
D	AGFBCE
NULL	AGFBCED

∴ One possible BFS traversal is
A→B→G→C→F→D→E.

One possible DFS traversal is
A→G→F→B→C→E→D.

Havel-Hakimi Algorithm

- Graph Realisation Problem: Let $A=(s, t_1, t_2, \ldots t_s, d_1, d_2, \ldots d_n)$ be a finite sequence of non-negative, non-increasing integers. (Such a sequence is called graphic.) Define a new sequence

(the elements are a sequence of degrees of vertices in a graph)

$$A' = (t_1-1, t_2-1, \ldots t_s-1, d_1, d_2, \ldots d_n) \quad \text{(say } s\text{)}$$

obtained from A by removing the leftmost (maximum) element, and then subtracting 1 from the first s elements (after removing s). If A' does not follow a non-decreasing order, arrange it in such an order.

Can any sequence be checked for whether it is graphic or not?

- The Havel-Hakimi algorithm is a recursive algorithm that checks for "graphicity".
  - → Base Case: A sequence of 0s is graphic. (=Null graph)
  - → Recursive Step: Generate A' from A as defined above. If the base case is reached, the sequence is graphic. If at any point, the terms become -ve, the sequence is not graphic.
- The recursive step is equivalent to removing the highest degree vertex (and all edges associated with it) of a graph.

Q: Use the Havel-Hakimi algorithm to check whether $A=\{6,5,5,4,3,2,1\}$ is graphic or not.

A: $A=\{6,5,5,4,3,2,1\}$
$A_1=\{4,4,3,2,1,0\}$
$A_2=\{3,2,1,0,0\}$
$A_3=\{1,0,-1,0\}$ ☹ = The sequence isn't graphic.

Fleury's Algorithm

- This is an algorithm that not only checks whether a graph is Eulerian or not (i.e. if ∃ an Eulerian path/circuit or not), but also generates one in an Eulerian graph.

Extras:

- Step 1: If the graph is disconnected, exit with the result "Not an Eulerian graph", else go to step 2.
- Step 2: If the number of vertices of odd degree is neither 0 nor 2, exit with the result "Not an Eulerian graph", else go to step 3.
- Step 3: If the number of odd vertices is 0, display the result "Eulerian cycle exists", begin at any vertex, and display it. If the number of odd vertices is 2, display the result "Eulerian path exists", begin at any odd vertex, and display it like so:
- Step 4: Choose an edge from the current vertex such that its removal does not disconnect the graph.
- Step 5: Move to the other vertex along this edge, remove this edge, and print the current vertex.
- Step 6: If the graph still has edges remaining, go to step 4, else exit.

Shortest Path Problem

It can be of three types:

- Single source, single destination
- Single source, all destination
- All source, single destination
  - → This case can be solved from "single source, all destination" by reversing directions of edges (in a directed graph), and using "single source, all dest" taking the given destination as a source.

- The shortest path problem deals with **weighted graphs**, which are graphs $G(V,E)$ where each edge is associated with a real number according to a weight function $W: E \to \mathbb{R}$.
- We are required to find a path in the graph from a source vertex (s) to a destination vertex (d) such that the sum of weights of the traversed edges is minimised.
- There are two algorithms - (i) Dijkstra's Algorithm (ii) Bellman-Ford Algorithm

Dijkstra's Algorithm

- To apply this algorithm, all edge weights must be **non-negative**.
- Imagine the vertices to be oil depots, and the edges to be pipelines. We set fire to the source vertex and assume the fire to travel at equal speeds along all pipelines.
- Then, the shortest path from the source to a particular vertex will be the time at which that vertex burns. (Shortest path to the source is 0.)
- **Algorithm:**
  - → **Step 1:** Start.
  - → **Step 2:** Initialise the source as 'visited' and the remaining vertices as 'not visited'. Assign the number 0 to the source, and ∞ to the rest of the vertices.
  - → **Step 3:** If all vertices are 'visited', go to Step 7.
  - → **Step 4:** Select a 'visited' vertex having a neighbour (vertex sharing an edge) which is 'not visited'. (Suppose A is the selected 'visited' vertex and B is its 'non-visited' neighbour.)
  - → **Step 5:** If $l(A) + \min(W(A,B)) \leq l(B)$, update $l(B) = l(A) + \min(W(A,B))$, where
    - l: the function used to assign numbers to vertices (eg: initialisation → $l(s) = 0$, and for all other vertices, $l(v) \to \infty$.).
    - $\min(W(A,B))$: the minimum of the weights of all edges connecting A and B
  - → **Step 6:** Update B as 'visited', then go to step 3.
  - → **Step 7:** The number associated with each vertex is the minimum weight of the path from the source to that vertex, i.e. the weight of the shortest path.
  - → **Step 8:** Stop.

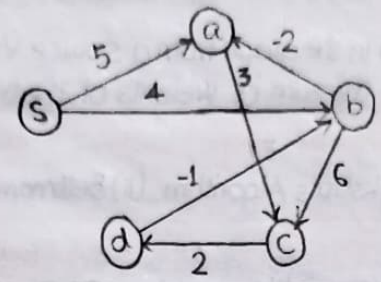
Greedy Approach: It is a problem-solving technique involving
- → **Local Optimisation:** ~~The best possible choice is made at each step~~ The option that seems the most beneficial at the moment is picked, hoping that it will lead to an optimal solution for the entire problem.
- → **Irrevocable Decisions:** No backtracking - a choice once made is never reconsidered.
- → The greedy approach is simple and finds a solution quickly, but the solution it finds isn't guaranteed to be optimal. Dijkstra's algo. is a greedy approach.

Bellman-Ford Algorithm

- This algorithm can be applied even if some edges have negative weight. However, it has a higher time complexity.
- The presence of negative cycles (cycles of 2 or more edges, where the sum of weights is negative) is also detected.
- The algorithm will be demonstrated in this question:

Q: Report the shortest path weights from the source, s, to every other vertex using both Dijkstra's & Bellman-Ford algorithms. Hence, show why Dijkstra's algorithm fails here.

this is a directed graph with -ve weighted edges!



A: Here Bellman-Ford algorithm: We initialise the distance of the source vertex (s) to be 0, and for the rest, the distance is initialised to ∞.

Distance = [0, ∞, ∞, ∞, ∞] (in the order [s, a, b, c, d])

Since |V| = 5, we perform |V| = 5 iterations: the first |V|-1 = 4 iterations are used for edge relaxations (i.e. adjusting distances if possible), and the last one is used to check if further updations are possible. (Do this blindly.)

Iteration 1: (edge relax'n:)

Edge s→b: $dist(b) = \min(dist(b), dist(s) + \min(w(s,b))) = \min(\infty, 0+4) = 4$

Edge s→a: $dist(a) = \min(dist(a), dist(s) + \min(w(s,a))) = \min(\infty, 0+5) = 5$

Edge d→b: $dist(b) = \min(dist(b), dist(d) + \min(w(d,b))) = \min(4, \infty-1) = 4$ (no updation)

Edge a→c: $dist(c) = \min(dist(c), dist(a) + \min(w(a,c))) = \min(\infty, 5+3) = 8$

Edge b→c: $dist(c) = \min(dist(c), dist(b) + \min(w(b,c))) = \min(8, 4+6) = 8$ (no updation)

Edge c→d: $dist(d) = \min(dist(d), dist(c) + \min(w(c,d))) = \min(\infty, 8+2) = 10$

Edge b→a: $dist(a) = \min(dist(a), dist(b) + \min(w(b,a))) = \min(5, 4-2) = 2$

∴ Distance = [0, 2, 4, 8, 10]

Remarks: (i) The edge updation of distances is done the same way as in Dijkstra's.
(ii) There is no specific "correct" order to process the edges. They can be processed in ANY order.

Iteration 2 (edge relaxation):

Edge s→b: $dist(b) = \min(dist(b), dist(s) + \min(w(s,b))) = \min(4, 0+4) = 4$ (no updation)

Edge s→a: $dist(a) = \min(dist(a), dist(s) + \min(w(s,a))) = \min(2, 0+2) = 2$ (no upd.)

Edge d→b: $dist(b) = \min(dist(b), dist(d) + \min(w(d,b))) = \min(4, 10-1) = 4$ (no upd.)

Edge a→c: $dist(c) = \min(dist(c), dist(a) + \min(w(a,c))) = \min(8, 2+3) = 5$

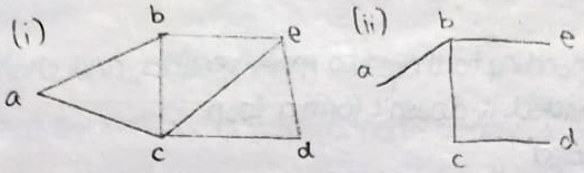
(contd. on pg 30)

(Contd. from pg. 27)

Tree

- A tree is an acyclic (circuitless) graph with no parallel edges or self-loops.
- A tree with n vertices has n-1 edges. As this is the minimum no. of edges required to have no isolated vertex, a tree is also a minimally connected graph.
- In a tree, there exists one and only one path between any two vertices.
- Spanning Tree: (of a graph G) a tree connecting all vertices of G using some or all the edges of G. It is also known as the skeleton of G, or the maximal tree subgraph of G.

A tree with ≥2 vertices has a minimum of two pendant vertices.



(i) a graph $G_1=(V,E)$ (ii) a spanning tree of G_1

- In a spanning tree of a graph G,
  - → branches are the edges of G included in the spanning tree.
  - → chords are the edges of G excluded from the spanning tree.
- If $|V|=n$ and $|E|=e$ for $G_1=(V,E)$, then, wrt a spanning tree,
  - → the rank (r) is the no. of branches of G_1, and $r=n-1$.
  - → the nullity (μ) is the no. of chords of G_1, and $\mu=e-n+1$.

(Remark: If there are 2 or more components, a spanning tree will exist for each component, and the set of these trees forms a spanning forest. For 'k' components, $r=n-k$, $\mu=e-n+k$. Note that all components are disconnected.)

Minimum Spanning Tree

↷ used in 'travelling salesman' prob^m.

- For a weighted graph $G_1=(V,E)$, the minimum spanning tree of G is the spanning tree with minimum total weight (of edges).
- There are two algorithms to find the minimum spanning tree, and both of them are greedy algorithms: (Note: Graphs are assumed to be undirected)

(i) Kruskal's Algorithm

- Initialise $G'(V',E')$ to be an empty graph, i.e. $V'=\phi$, $E'=\phi$.
- Sort the edges according to their weights.
- Perform the following steps for $|V|-1$ iterations:
  - → Choose the smallest weight edge.
  - → If it forms a cycle with any edges in E', discard it and move on to the next smallest edge. If not, add this edge to E' and its end vertices to V'.
- Now $G'(V',E')$ is the required minimum spanning tree.

Extras:

Prim's Algorithm

- We create a weight matrix which takes a similar form to the adjacency matrix:

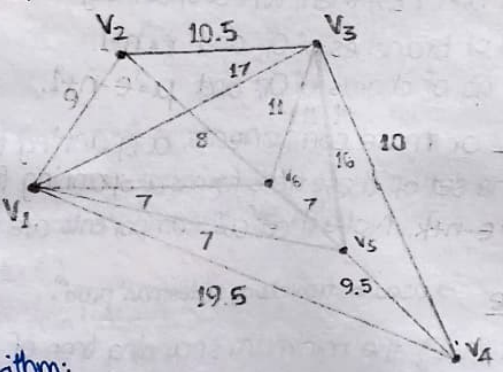
$$a_{v_i v_j} = \begin{cases} -, & \text{if } i=j \quad \text{(Because no self-loops)} \\ W(v_i, v_j), & \text{if } i \text{ \& } j \text{ are connected by an edge } (i \neq j) \\ \infty, & \text{otherwise} \end{cases}$$

	v_1	v_2	...	v_n
v_1	-			
v_2		-		
⋮				
v_n				-

- Consider the first row and choose the minimum element in it. Connect the corresponding nodes.
- Now consider the rows corresponding to these two vertices, and choose the minimum of their entries, provided it doesn't form a loop.
- Repeat until all vertices are included.

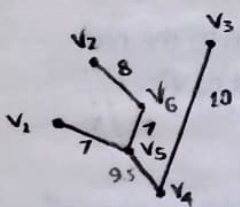
> **Remark:** The subgraph constructed in Prim's algorithm will be connected at all stages, but it may be disconnected in Kruskal's, for some stages.

Q: Find a minimal spanning tree for the following graph using both Prim's and Kruskal's algorithms.



A: Using Prim's algorithm:

	v_1	v_2	v_3	v_4	v_5	v_6
→ v_1	-	~~9~~	17	19.5	(7)	~~7~~
→ v_2	~~9~~	-	10.5	∞	∞	~~8~~
→ v_3	17	10.5	-	~~10~~	16	11
→ v_4	19.5	∞	(10)	-	~~9.5~~	∞
→ v_5	~~7~~	∞	16	(9.5)	-	(7)
→ v_6	~~7~~	(8)	11	∞	~~7~~	-

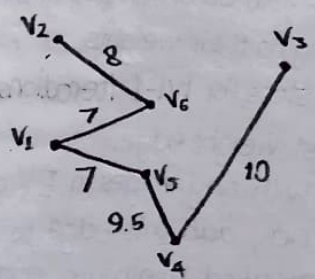


The obtained minimum weight is 41.5.

Using Kruskal's algorithm:

v_1v_5	v_1v_6	v_6v_5	v_2v_6	v_1v_2	v_4v_5	v_3v_4	v_2v_3	v_3v_6	v_3v_5
(7)	(7)	~~7~~	(8)	~~9~~	(9.5)	(10)	10.5	11	16

v_1v_3	v_1v_4
17	19.5



The obtained minimum weight is 41.5.

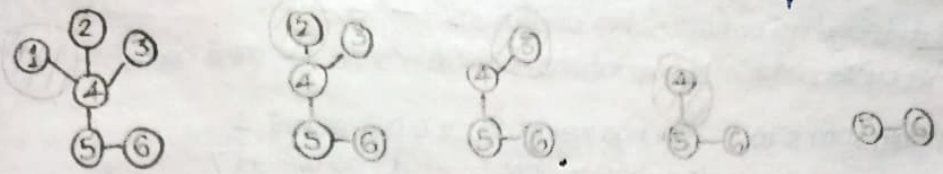
- **Cayley's Formula:** The number of ~~spanning~~ labelled forests (spanning forests, in particular) in a graph with n nodes and k components, where each component is completely connected, is $T_{n,k} = kn^{n-k-1}$ (in particular, $T_{n,1} = n^{n-2}$) → proof on next pg

Proof: We first introduce Prüfer's encoding, a way to represent a tree as a sequence of numbers. The encoding is done as follows:

While the tree has more than 2 vertices, (Assume that the vertices are NUMBERED)

→ Choose the pendant vertex with the lowest number.

→ Remove this vertex and add its root's number to the sequence.



Seq: ϕ Seq: {4} Seq: {4,4} Seq: {4,4,4} Seq: {4,4,4,5}

Note that the number of times a node appears in the sequence is **one less than its degree**. Hence pendant vertices do not appear at all.

The Prüfer's encoding of any labelled tree is unique.

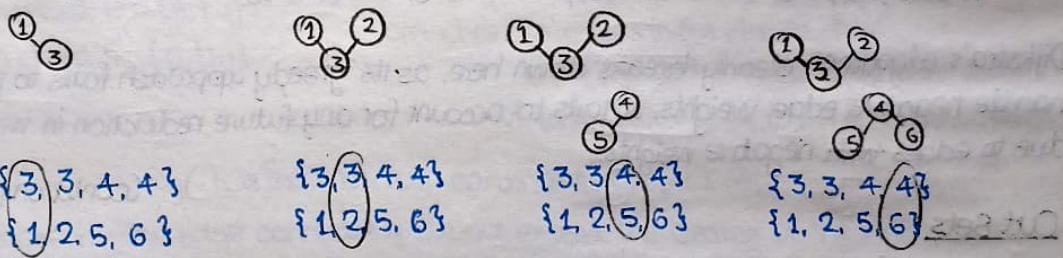
To decode a Prüfer's encoding - say {3,3,4,4}:

→ The number of vertices is 2 more than the number of elements in the sequence, (here, 4+2=6)

∴ Vertices are {1,2,3,4,5,6}. We can also infer that the degrees of vertices 1,2,3,4,5 and 6 are respectively 1,1,3,3,1 and 1.

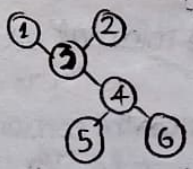
→ Note the vertices absent from the sequence - here 1,2,5 & 6. - Pendant Vertices

→ For $i \in [1, n-2]$ (n = No. of vertices), i initialised to 1 and incremented by 1, take the i^{th} pendant vertex and attach it to the i^{th} number in the Prüfer's encoding:



{3, 3, 4, 4} {3, 3, 4, 4} {3, 3, 4, 4} {3, 3, 4, 4}
{1, 2, 5, 6} {1, 2, 5, 6} {1, 2, 5, 6} {1, 2, 5, 6}

→ Now add any missing edges, in order to make the vertices consistent with their degrees.



Final tree

 (contd. from pg. 26)

Edge b→c : dist(c) = min(dist(c), dist(b) + W(b,c))) = min (5, 4+6) = 5 (no upd.)
Edge c→d : dist(d) = min(dist(d), dist(c) + W(c,d))) = min (10, 5+2) = 7
Edge b→a : dist(a) = min(dist(a), dist(b) + W(b,a))) = min (2, 4-2) = 2 (no upd.)

∴ Distance = [0,2,4,5,7]

Iterations 3&4 (edge relaxations): No updations.
Iteration 5 (-ve cycle check): No updations ⇒ There are no -ve edge cycles.

∴ The shortest path from s to → a has weight 2, → b has weight 4,
→ c has weight 5, → d has weight 7.

Dijkstra's Algorithm: The initialisation step:

Distance ([s,a,b,c,d] in that order) = [0, ∞, ∞, ∞, ∞]
Visited = {s}, Not visited = {a,b,c,d}

Vertex 'a': dist(a) = min(dist(a), dist(s) + min(W(s,a))) = min(∞, 0+5) = 5 (s→a)
Distance = [0, 5, ∞, ∞, ∞], Visited = {s, a}, Not visited = {b,c,d}
Vertex 'b': dist(b) = min(dist(b), dist(s) + min(W(s,b))) = min(∞, 0+4) = 4 (s→b)
Distance = [0, 5, 4, ∞, ∞], Visited = {s,a,b}, Not visited = {c,d}
Vertex 'c': dist(c) = min(dist(c), dist(a) + min(W(a,c))) = min(∞, 5+3) = 8 (a→c)
Distance = [0,5,4,8,∞], Visited = {s,a,b,c}, Not visited = {d}
Vertex 'd': dist(d) = min(dist(d), dist(c) + min(W(c,d))) = min(∞, 8+2) = 10 (c→d)
Distance = [0,5,4,8,10], Visited = {s,a,b,c,d}, Not visited = ϕ

∴ The weight of the shortest path from s to
→ a is 5, → b is 4, → c is 8, → d is 10.

Dijkstra's algorithm clearly breaks down here, as its greedy approach fails to properly handle negative edge weights. It fails to account for any future reduction in weight due to edges with negative weights.

(contd. on pg. 27)

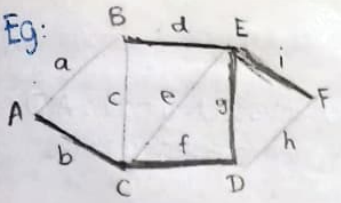
Cut Sets

- It is a set of edges of a connected graph, such that removing these edges leaves the graph disconnected.
- **Theorem:** In a connected graph, the minimal set of edges containing one branch from every spanning tree is a cut set.
  - → **Proof:** Suppose this is not a cut set (on the contrary).
    - ∴ If we remove this minimal set of edges, then according to our assumption, the remaining graph is still connected.
    - ∴ We can obtain a spanning tree of this graph. Clearly this is also a spanning tree of the original connected graph.

    But note that no edge (branch) of this spanning tree belongs to the minimal set of edges containing one branch from every spanning tree, a contradiction.

    Therefore our assumption is wrong. Thus the minimal set of edges containing one branch from every spanning tree is a cut set.
- **Fundamental Circuit:** A fundamental circuit wrt a spanning tree T of a graph G is a cycle in G consisting exactly one chord (non-branch) of T, the rest being branches of T.

Eg:



$G_1 = (\{A,B,C,D,E,F\}, \{AB, AC, BC, BE, CD, CE, DE, DF, EF\})$

$T = (\{A,B,C,D,E,F\}, \{AC, BE, CD, DE, EF\})$

Some fundamental circuits are: (branches, chords)

$\{DE, EF, DF\}$, $\{BE, ED, CD, BC\}$, $\{AB, AC, CD, DE, BE\}$

- **Fundamental Cut Set:** A fundamental cut set wrt a spanning tree T of a graph G is a ~~set of edges of~~ cut set of G containing **exactly one branch** of T.

~~Eg: A fundamental cut~~

→ A fundamental cut set must have as low of a cardinality as possible.

→ Eg: A fundamental cut set

- with branch 'b': {a,b}
- with branch 'f': {a,c,e,f}
- with branch 'i': {h,i}
- with branch 'd': {a,c,d}
- with branch 'g': {a,c,e,g,h}

- **Theorem:** With respect to a spanning tree ~~of~~ T of a graph G,
  - → a chord that determines a fundamental circuit T appears in every fundamental cutset associated with the branches of T, and nowhere else.
  - → a branch that determines a fundamental cut set S appears in every fundamental circuit associated with the chords of S, and nowhere else.

Eg: In the above graph, a fundamental circuit

- → with chord 'a': {a,b,d,f,g}
- → with chord 'c': {c,d,f,g}
- → with chord 'e': {e,f,g}
- → with chord 'h': {g,h,i}

** fundamental cut sets & circuits

Hence, in accordance with the theorem,

chords: 'a' occurs in the cut sets of 'b', 'd', 'f' & 'g'
'c' occurs in the cut sets of 'd', 'f' & 'g'
'e' occurs in the cut sets of 'f' & 'g'
'h' occurs in the cut sets of 'g' & 'i'

branches: 'b' occurs in the circuits of 'a'
'd', 'f', 'g' and 'i' occur only in the circuits of {a,c}, {a,c,e}, {a,c,e,h} & h respectively.

- **Edge Connectivity:** It is the minimum cardinality of a cut set.
  - → **Theorem:** The edge connectivity cannot exceed the degree of the vertex of the graph with least degree. (Upper bound on edge connectivity)
- **Vertex Connectivity:** It is the minimum number of vertices to be removed from a graph such that the remainder is disconnected.
  - → **Theorem:** The vertex connectivity cannot exceed the edge connectivity of a graph. (Lower bound on edge connectivity)
  - → A graph whose vertex connectivity is 1 is said to be a **separable** graph. For a separable graph, this special vertex is called the **cut vertex.**
  - → **Cut Vertex:** A vertex z is a cut vertex iff ∃ two mutually exclusive sets of vertices V_1 and V_2 in G (such that $V_1 \cup V_2 = V - \{z\}$) such that any path from some $x \in V_1$ to some $y \in V_2$ contains the vertex z.
  - → **Theorem:** The maximum vertex connectivity of a graph G(V,E), where $|V| = v$ and $|E| = e$ is floor($2e/v$).

$\therefore$ vertex connectivity $\leq$ edge connectivity $\leq \frac{2e}{v}$.

(Since $2e/v$ is the mean degree of a vertex, ∃ a vertex whose degree is $\leq 2e/v$. The vertex with minimum degree has to be one of these kinds of vertices.)

- A connected graph is said to be **k-connected** iff
  - → every pair of vertices is connected with a minimum of k non-intersecting paths, AND
  - → there exists a pair of vertices connected with exactly k non-intersecting paths.

  Such a graph has a **vertex connectivity of k.**

- A connected graph has an **edge connectivity of k** iff every pair of vertices is connected by k or more edge-disjoint paths, and there exists a pair of vertices connected with exactly k edge-disjoint paths.

Note the difference:

- → Edge-disjoint: No edges should be in common, but vertices may overlap.
- → Non-intersecting: No edges or vertices should be common (except start & end).

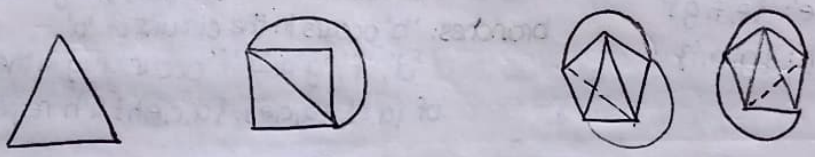
A graph is non-separable (vertex connectivity > 1) if every pair of vertices in the graph can be placed in some circuit of the graph.

Planar Graphs

- **Embedding:** An embedding of a graph is its geometric representation on a surface, where vertices are represented by points and edges by arcs between these vertices.
  - → No two edges intersect in an embedding.
- **Planar Graph:** A graph is said to be planar if its embedding on the XY plane exists.

Q: ~~Kuratowski graphs are completely connected graphs~~ K_i denotes a fully connected graph with i vertices. Show that K_3 and K_4 are planar, but K_5 is not.

A:



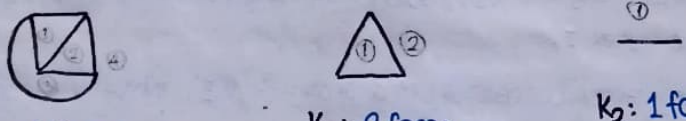
K_3 ☺ K_4 ☺ K_5 ☹

There is no way to accommodate all 10 edges of K_5 on the XY plane's embedding. Hence K_5 is non-planar. However, if one edge was removed, then the rest would be planar.

Kuratowski Graphs: They are K_5 and $K_{3,3}$. (See page 22)

- K_5 : The non-planar graph with the smallest number of vertices (5 vertices, 10 edges)
- $K_{3,3}$: The non-planar graph with the smallest number of edges (6 vertices, 9 edges)

- **Face:** A face is a part of the surface demarcated by the edges of the embedding. An infinite part is also considered to be a face.



K_4: 4 faces K_3: 2 faces K_2: 1 face

- **Euler's Theorem:** Let a planar graph have v vertices and e edges, and divide the surface/plane into f faces. Then

$$f + v = e + 2$$ (2 is Euler's characteristic)

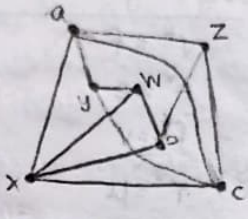
- Necessary (may not be sufficient) conditions for a graph to be planar are:
  - → for a simply connected graph: $e \geq 3f/2$, $e \leq 3v-6$
  - → for a bipartite graph: $e \geq 3f/2$, $e \leq 2v-4$

Note: The notion of a face only arises in case the graph is planar.

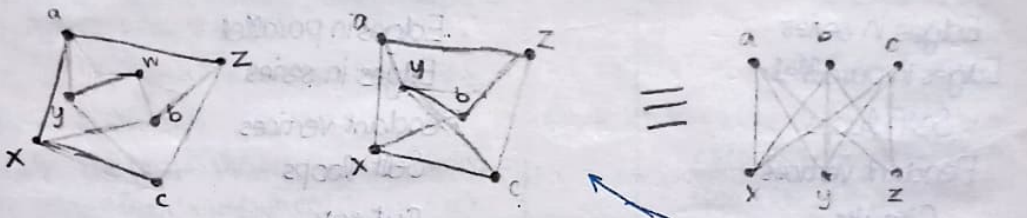
A map is an example of a planar graph; countries/states are nodes and borders are edges.

- **Theorem:** For a graph G to be planar, a necessary and sufficient condition is that neither of the Kuratowski graphs, nor any graph which is homeomorphic to them, is a subgraph of G.

Q: Is this graph planar?



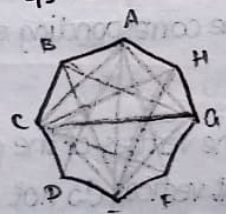
A: Consider this subgraph:



(Remove edges wx & ac) ("smoothen out" y-w-b) $K_{3,3}$ homeomorphism

Hence the graph is not planar.

Q: Show that K_5 and $K_{3,3}$ are both subgraphs of this graph.

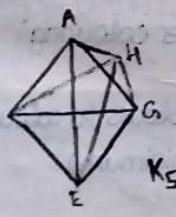


$V=8$, $e=21$

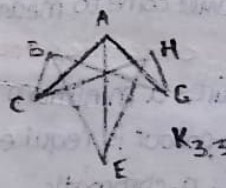
$21 \leq 3 \cdot 8 - 6$ is FALSE

$\therefore$ non-planar

A:



K_5



$K_{3,3}$

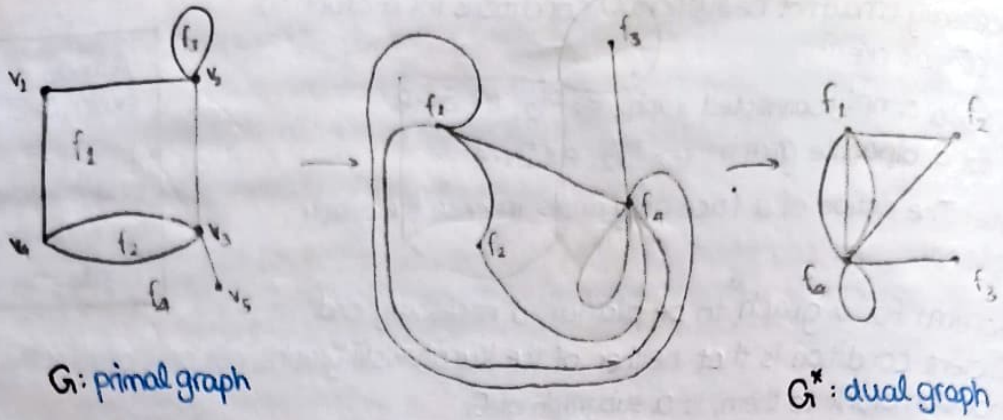
Note: You do not need to select every edge of a vertex if you choose to include it in the subgraph. Just check if the required ~~vertices~~ edges are there.

- **Dual of a Planar Graph:** Consider a planar graph G. The dual of G is a graph G^* such that
  - → the faces into which G divides the plane each correspond to one vertex of G^*
  - → for every edge $(u,v) \in G$, demarcating the plane, an edge of G^* is constructed between the vertices of the faces on either side of (u,v)

- Eg: (PTO)

Extras:

The dual of a planar graph's dual is itself.



G: primal graph

G^*: dual graph

Things in the primal graph	Corresponding things in the dual graph
Vertex count: v	$v^* = f$
Edge count: e	$e^* = e$
Face count: f	$f^* = v$
Rank: r	$r^* = \mu$
Nullity: μ	$\mu^* = r$
Edges in series	Edges in parallel
Edges in parallel	Edges in series
Self-loops	Pendant vertices
Pendant vertices	Self-loops
Circuits	Cut sets
Cut sets	Circuits

- **Theorem:** There exists a one-one correspondence between the edges of G and G^*; a set of edges in G forms a circuit iff the corresponding edges in G^* form a cut set.

Graph Colouring

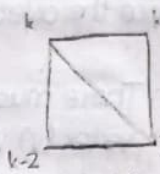
- As the name suggests, we apply colours to the vertices of the graph.
- A colouring is said to be proper if adjacent vertices do not share the same colour. (Unless stated otherwise, the term "colouring" will come to mean "proper colouring" from now)
- A graph is said to be k-chromatic if it requires a minimum of k colours to colour it. (chromatic number)
  - → A null graph $G(V, \phi)$ is monochromatic. (1-colour is required at minimum)
  - → A fully connected graph having n vertices is n-chromatic.
  - → All trees are bichromatic (2-chromatic), and so too are all bipartite graphs.
  - → Circuits having an even number of vertices are bichromatic and those with an odd number of vertices are trichromatic. (3-chromatic)
- **Four Colour Theorem:** A planar graph requires no more than 4 colours to make a proper colouring.

Extras:

Kempe (1879) - hypothesised "5 colour th^m"
Heawood (1890) - Proved

- If d_{max} is the highest degree of any vertex of a graph, then its chromatic number cannot exceed $d_{max}+1$. Furthermore, if there is no complete subgraph having $d_{max}+1$ vertices, the chromatic number cannot exceed d_{max}.

- **Chromatic Polynomial**: It is a polynomial $P_G(k)$, where $k \geq$ chromatic no, which gives the number of possible colourings with k colours.
  - → For a complete graph K_n: $P_G(k) = k(k-1)\dots(k-n+1)$
  - → For a tree with n vertices: $P_G(k) = k(k-1)^{n-1}$



$P_G(k) = k(k-1)(k-2)^2$

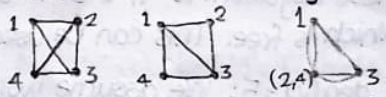
Fundamental Reduction Theorem

- This theorem is applicable only to connected graphs. ↝ if disconnected, $P_G(k) = P_{G_1}(k) \times P_{G_2}(k) \times \dots \times P_{G_m}(k)$ (product of chrom. poly. of all connected components)
- Any pair of vertices in a graph
  - → if adjacent: must have different colours
  - → if not adjacent: may have either the same colour or different colours

We constrain the pair of vertices to have the same colour by contracting an edge.

- **Theorem**: $P(G,k) = P(G-e,k) - P(G/e,k)$
  - → $P(G,k)$: The chromatic polynomial of a graph G, given k colours.
  - → $G-e$: The graph obtained by removing the edge e from G.
  - → G/e: The graph obtained by contracting the edge e from G (squish endpts. 2gether). This forces endpts. to have the same colour.

Q: Use the Fundamental Reduction Theorem to compute the chromatic polynomial of K_4.



A: $P(K_4,k) = P(K_4-\{(2,4)\},k) - P(K_3)$
$= k(k-1)(k-2)^2 - k(k-1)(k-2)$
$= k(k-1)(k-2)(k-3)$.

- Some features of the chromatic polynomial $P_G(k) = a_p x^p + a_{p-1}x^{p-1} + \dots + a_1 x + a_0$:
  - (A) → Its degree (p) is the number of vertices.
  - (B) → it is a monic polynomial → $a_p = 1$
  - (C) → its constant term is zero - $a_0 = 0$
  - (D) → $P_G(x) = x^n$ or the sum of coefficients of all terms of $P_G(x)$ is 0.

- **Proofs**:
  - → Put $x=0$ (No colours), thus $P_G(0) = a_0$. Since there are no ways to colour the graph with 0 colours, $a_0 = 0$. - (C) proved.
  - → The sum of coefficients of $P_G(x)$'s terms is $P_G(1)$, the number of ways to colour the graph with 1 colour. If the graph is not null (set of edges is not ϕ), then $P_G(1) = 0$ (no ways to colour). If the graph is null, $P_G(x) = x^n$ (n vertices, x ways to colour each). - (D) proved.
  - → **Base Case**: For a null graph $G(V,\phi)$ where $|V| = n$, $P_G(k) = k(k-1)\dots(k-n+1)$, a monic polynomial of degree n.
  - **Inductive Step**: (Using strong induction) Assume that all graphs $G_{sub}(V,E)$, where $|E| < e$, the facts (A) & (B) hold true for $P_{G_{sub}}(k)$. Now add a new edge to make e edges in total. Using the Fundamental Reduction theorem, $P(G,k) = P(G-newEdge,k) - P(G/newEdge,k)$ which is also of deg n & monic. - (A) & (B) proved. (deg n, monic; deg n-1, monic)

If d_{max} is the maximum degree of a vertex, the graph's chromatic number is $\leq d_{max}+1$.

Q: Prove the "Five Colour Theorem", which states that a planar graph can be properly coloured with at most 5 colours.

A: A necessary condition for a planar graph with e edges and n nodes is $e \leq 3n-6$.
Hence the average degree, $\frac{2e}{n} \leq \frac{6n-12}{n} = 6 - \frac{12}{n} < 6$.

∴ There must exist some vertex v with degree ≤ avg., i.e. at least 1 vertex (v) with degree ≤ 6 (≤ 5).

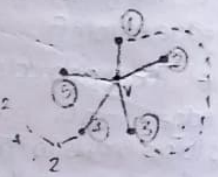
Base case: Any planar graph with n = 1, 2, 3, 4 or 5 vertices can be properly coloured with a maximum of 5 colours.

Inductive step: Assume that for any planar graph with k-1 colours, 5 colours are sufficient. Now we take a planar graph with k vertices. Let v be a vertex of this graph with degree ≤ 5. On removing v, the remaining graph is colourable with 5 colours. Now we add v back.

→ If deg(v) < 5: Besides the colours used to colour those adjacent to v, there is definitely one colour which is free. This can be used for v.

→ If deg(v) = 5: We assume worst case scenarios.

- Let 1 and 3 be connected by a path that CANNOT allow them to have any other colour.
- Since the graph is planar, 2 cannot be connected to 4 or 5 by any path/edge.



- Applications of Colouring: Exam Scheduling, Register Scheduling, Sudoku

Chromatic Partitioning

- A chromatic partitioning of a graph is a distribution of the set of its vertices into the minimum no. of mutually exclusive & exhaustive subsets, such that each of the vertices in a subset can be given the same colour in a proper colouring.

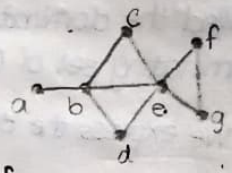
- Independent Set: A set $S \subseteq V$, which is non-empty, is said to be an independent set of a graph G if no two vertices in S are adjacent.
  - → The trivial case of an independent set is a set with a single vertex.
  - → An independent set S is said to be maximal if adding any vertex to S violates the property of independence.

- The coefficient of internal stability of a graph G, denoted by β(G), is the maximum cardinality of a maximal independent set of G. For a k-chromatic graph with n vertices, the lower limit of β(G) is n/k, i.e. $\beta(G) \geq n/k$.

Extras:

In the Fundamental Reduction Theorem:
- P(G, k): Endpoints of e MUST have diff. colours.
- P(G-e, k): Endpts. of e CAN have same/diff colours.
- P(G/e, k): Endpts. of e MUST have same colour.

- To find maximal independent sets of a graph:
  - → Express each edge as a product of its endpoints, eg: Edge (b,e) as be.
  - → Obtain the Boolean expression for the sum of edges:

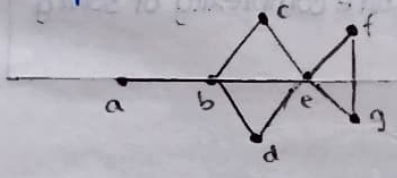


$$\Phi = \sum_{e \in E} e := ab + bc + bd + be + ce + de + ef + eg + fg$$

  - → ~~Expressing the above as a~~ The above should not be valid (i.e. equal 0) for any independent set of vertices. (Each vertex is treated as a Boolean literal, and 1 ~~indicates~~ means that this vertex is included in a set of vertices, 0 means it's excluded)
  - → Thus, if we express Φ' as a ~~Product of sums (PoS)~~ sum of products (SoP), each product represents a maximal independent set.
  - → $\Phi' = (a'+b')(b'+c')(b'+d')(b'+e')(c'+e')(d'+e')(e'+f')(e'+g')(f'+g')$
    - = ~~$(b'+ac)(d'+e'+bd)$~~ $(b'+c'd')(e'+c'd')(b'+a'e')(e'+f')(e'+g')(f'+g')$
    - = $(b'+a'c'd'e')(e'+c'd'f')(g'+e'f')$
    - = $b'e'g' + b'e'f' + b'c'd'f'g' + b'c'd'e'f' + a'c'd'e'g' + a'c'd'e'f' + a'c'd'f'g'$
  - → Φ' must be equated to 1 for an independent set.
    - $b'e'g'$ ⇒ Exclude b,e,g to get a maximal independent set {a,c,d,f}
    - $b'e'f'$ ⇒ exclude them ⇒ {a,c,d,g}
  - ∴ ~~All maximal~~ independent sets are:
    {a,c,d,f}, {a,c,d,g}, {a,e}, ~~{a,}~~ {b,f}, {b,g}

Maximal

- A graph is said to be uniquely ~~colour~~ colourable if its chromatic partitioning is unique.



- This graph is not uniquely colourable because >1 partitioning exist:
  {a,c,d,g}, {b,f}, {e}
  {a,c,d,f}, {b,g}, {e}
  {a,e}, {b,f}, {c,d,g}, ...

- **Dominating Set:** A set $S \subseteq V$ is said to be a dominating set of a graph G if (v,s) are adjacent ∀ $v \in V-S$ and ∀ $s \in S$.
  - → The trivial case of a dominating set is S=V.
  - → A dominating set S is said to be minimal if removing any vertex from S violates the property of dominance.

> Every maximal independent set is a minimal dominating set, but the converse is false. Eg: In the above graph, {b,e} is min'. dominating set but not a max'. ~~ind~~ independent set.

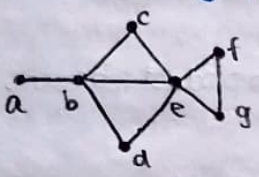
- The coefficient of external stability of a graph G, denoted by $\alpha(G)$ (also called the domination number), is the minimum cardinality of a minimal dominating set of G. We have $\alpha(G) \leq \beta(G)$.

- If we express the Boolean expression

this is a PoS → $\ominus = \prod_{v_i \in G} \frac{(v_i + v_{i_1} + v_{i_2} + \dots + v_{i_{d_i}})}{\text{vertex + sum (all its neighbours)}}$ (where d_i = degree of v_i)

as a sum of products (SoP), each of the products is a minimal dominating set.

- Thus for this graph:



$\ominus = (a+b)(b+a+c+d+e)(c+b+e)(d+b+e)$
$(e+c+d+f+g)(f+e+g)(g+e+f)$
$= (a+ab+ac+ad+ae+b+bc+bd+be)$
$(b+cd+e)(e+ef+eg+ce+cf+cg+be+bf+bg$
$+de+df+dg+fg+f+g)$

$= (a+ab+ac+ad+ae+b+bc+bd+be)(b+cd+e)(e+ef+eg+ec+eb+bf+cf+df+gf+f+g+cg+bg+dg)$

$= (a+b)(b+cd+e)(e+f+g) = (b+ea+e)(ae+af+ag+be+bf+bg)$
$= (ab+acd+ae+b+bcd+be)(e+f+g) = (b+acd+ae)(e+f+g)$
$= abe+abf+abg+acde+acdf+acdg+ae+aef+aeg+be+bf+bg+$
↳ $be+bf+bg+acde+acdf+acdg+ae+aef+aeg$
$= be+bf+bg+ae+acdf+acdg.$

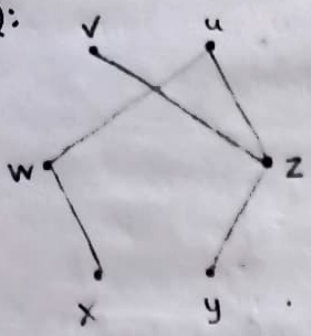
∴ The dominating sets (minimal) are:
{b,e}, {b,f}, {b,g}, {a,e}, {a,c,d,f}, {a,c,d,g}

Note that {b,e} is a minimal dominating set but not a maximal independent set.

Boolean Satisfiability Problem: It is a problem of assigning values to variables in a Boolean expression (either True or False), such that the exp^n evaluates to True. (We shd find all solns)

If the problem is a Boolean exp in n variables, the time complexity of solving the problem is exponential - $O(2^n)$.

Q:



(i) Find a maximal independent set of this graph.

(ii) Find a minimal vertex cover of this graph.

A: (i) $\{u, v, x, y\}$

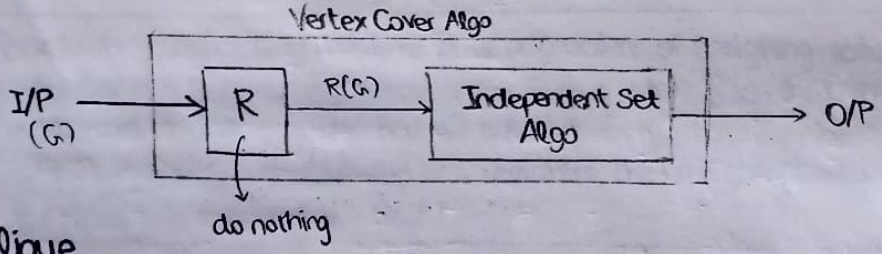
(ii) $\{w, z\}$

- Note that we know the algorithm of finding all maximal independent sets.
- The complement of every maximal independent set is a minimal vertex cover.

∴ We have solved the vertex cover problem by solving the independent set ~~vertex~~ problem. This is called **reduction**. The general process of reduction is:

- **Prerequisites:** We know to apply Algo1 on an input to get an output.
- **Goal:** To find (or devise) an algorithm Algo2.
- **Process:** (Reduction)
  - → Convert the input of Algo2 into a form which can be accepted by Algo1.
  - → Use Algo1 on the converted input to get output.
  - → This output is what is required. ∴ Algo2 = (Reduction + Algo1)

- Eg: In the question we just solved, we knew Indep. set but not Vertex Cover.
  - → **Reduction:** Conversion of input graph to a graph that can be accepted by the indep. set. algo. (Here, the this step is to do nothing)
  - → Apply the indep. set. algo, then take complement, to obtain a minimal vertex cover.
  - → Thus we have solved the vertex cover problem.



Clique

- A clique of a graph G_1 is a fully connected subgraph of G_1.
- The vertex cover of a graph G_1 (minimal) can be reduced to a clique problem (find a clique in a graph ~~if it exists~~) as follows:
  - → **Reduction:** Construct a graph G_1' from G_1, where the vertices are the same, but $E' = \{\text{edges of } K_n\} - E$.

    ∴ if $G_1 = (V, E)$ and $|V| = n$, then $G_1' = (V, E')$ where $E' = \{\text{edges of } K_n\} - E$.
  - → **Clique Algorithm:** Apply the clique algorithm on G_1' to find a clique. Let S' be the set of vertices in the clique. (Note that this clique shd. be of max. size)
  - → **Output of Vertex Cover:** $S = V - S'$ is a minimal vertex cover.

Matching

- A matching in a graph G is a set of edges of G (i.e. a subset of E), no two of which share a common vertex.

Q: Find a matching set in the following graph.

A: Initialise $E' = E$, $M = \phi$, $C = \phi$.
While $E' \neq \phi$,
→ Choose a random edge (u,v) from E'.
→ $M = M \cup (u,v)$, $C = C \cup \{u,v\}$.
→ $E' = E' - \{e : e \text{ is incident from any vertex in } M\}$

At the end, M will be a matching. C will be a vertex cover (not necessarily minimal). Eg: $M = \{(a,b),(d,f),(g,h)\}$, $C = \{a,b,d,f,g,h\}$.
However, M is a maximal matching set - what this means is that if you add an edge to M, the resultant set won't be a matching set.

By definition of this algorithm, we have $n(C) = 2 \cdot n(M)$.

time complexity $O(|E|)$

The size of a maximal matching set cannot exceed that of the minimum vertex cover. (it can only be equal at best)

Approximation Ratio

- The approximation ratio of an algorithm is defined as

$$\text{Approx. Ratio} = \frac{\text{Size of output of algo}}{\text{Optimal solution}}$$

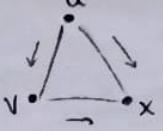
- It is always greater than or equal to 1. The closer it is to 1, the better.
- For the algorithm defined above,

$$\text{Approx Ratio} = \frac{n(C)}{n(O)} = \frac{n(M) \cdot 2}{n(O)} \leq \frac{2 \cdot n(M)}{n(M)} = 2 \quad (\text{Since } n(M) \leq n(O))$$

↳ The least

Travelling Salesperson Problem

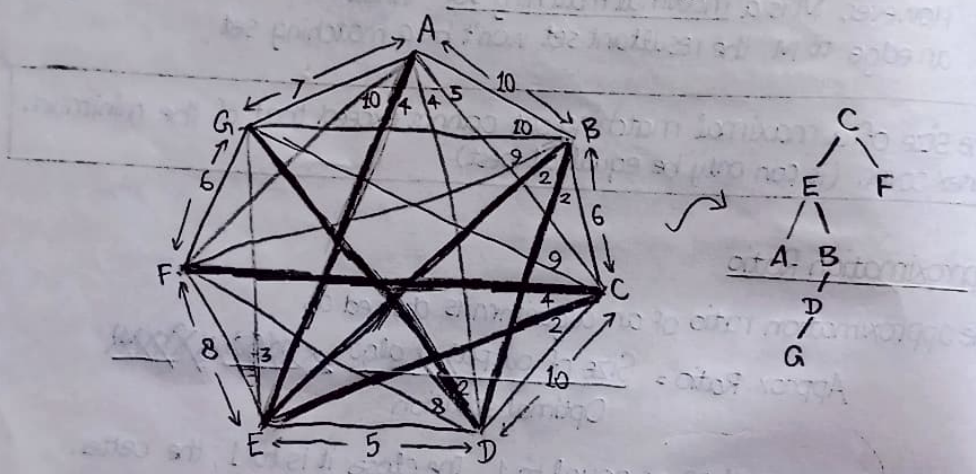
- **Setup:** Cities are represented by nodes, and paths between cities as edges. Each edge is associated with a cost. The graph as a whole is a complete graph.
- **Problem:** A salesman is to visit all cities exactly once and leave the group of cities the way they entered. This equates to finding a Hamiltonian ~~path~~ circuit of ↓ cost.
- **Assumption (Triangular Inequality):** The cost of travelling from a node N_1 to another node N_2 cannot exceed that of the cost of the direct edge b/w N_1 and N_2.



$\text{cost}(u \xrightarrow{v} x) \geq \text{cost}(u \xrightarrow{\phi} x)$.

- Procedure:
  - → Use Prim's or Kruskal's algorithm to find the minimal spanning tree (MST) for the complete graph (of cities).
  - → Start from any city. Taking this city as the root of the MST, do a preorder traversal of the MST (visit the root before traversing each subtree, sequentially) from left to right.
  - → Follow the edges of the tree along the preorder traversal. In case we encounter an edge which isn't in the MST, this implies backtracking in the tree (avoid this for Hamiltonian circuits!).
  - → Avoid backtracking by directly taking the cross-edge to the next vertex in the preorder traversal. By the triangular inequality assumption, the cost maintained is still optimal.
  - → Once the preorder traversal is done, take the edge back to the starting vertex (city). This completes the Hamiltonian circuit.

Q: Find an optimal route for a salesperson to travel through the following city network (costs of edges are represented next to them).



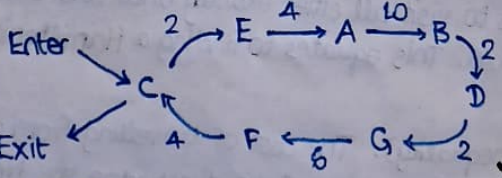
A: A minimum spanning tree can be generated using Kruskal's algorithm:

(B,D)	(B,E)	(C,E)	(D,G)	(E,G)	(A,E)	(A,D)	(C,F)
2	2	2	2	3	4	4	4
✓	✓	✓	✓	×	✓	×	✓

(6 edges selected)

Let's arbitrarily start from city C as shown: preorder traversal gives C E A B D G F.
Non-tree edges are (A,B) and (F,G). So the Hamiltonian circuit involves these edges as well.

∴ An optimal Hamiltonian circuit of cities is:



The minimum cost is 30.