* Comparison betⁿ Dijkstra and Bellman Ford

Bellman Ford	Dijkstra
works when there is (-ve) edge cycle	doesn't work
The result contains info. about other vertices they are connected to.	The result contains whole ~~[illegible]~~ info. about the n/w not only vertices they are connected to.
Can easily be implemented in a distributive way.	cannot be easily be implement - - - -
More time consuming than Dijkstra's	less time consuming
O(VE)	O E(log V)
Dynamic Programming approach is taken to implement the algo.	Greedy approach - - - - -

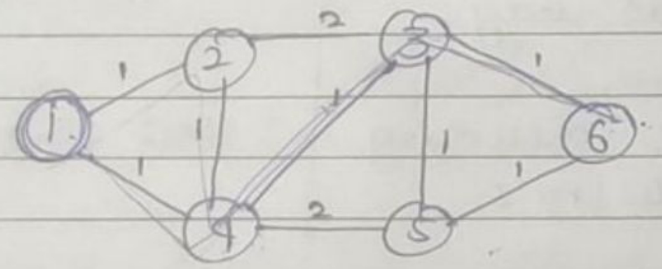
* Shortest Path computation with Candidate Path caching :-

→ there are certain networking environments where a list of possible paths is known or determined ahead of time such path are referred as candidate path list.

→ Path caching refers to storing of a candidate path list at a node ahead of time.

→ the link cost is periodically updated, then the computation of shortest path becomes very easy.

→ e.g,



→ we look for least cost path, we find that 1-4-3-6 is most preferred path due to its lowest end-end cost.

→ Suppose link cost 4-3 changes from 1 to 5 so, if we know the list of candidate path we can recognize the path cost and find out that 1-4-3-6 is no longer the least cost.

→ It is imp. to note that candidate path list is not required to include all possible paths betⁿ nodes i and j, only the preferrable path are considered

* Widest Path computation with Candidate Path Caching :-

→ There are many networking environment in ~~where~~ which the additive cost property is not applicable.
e.g, Dynamic call routing in voice telephone network.
∴ Determining path when the cost in non-additive is also an important problem in network Routing.

→ Suppose than the bandwidth betⁿ node l and m = 0, blm = 0
∴ link is not feasible since there is no bandwidth.

Consider a path between node 1 and node 2 consisting of 3 links, the first link has bandwidth = 10 units, 2nd = 5 units, 3rd = 7 units
Now,
if we say cost of path is additive
i.e, $10 + 5 + 3 = \underbrace{22}$
it is unlikely to make any sense.

Date
Page

Routing Table:

→ Communication n/w connects a set of nodes through link so that the routing traffic can be moved from source to destination.

→ For all the traffic to go to its destination, nodes in the n/w must provide directions.

→ To do that, each node in the n/w maintains a routing table so that the traffic can be forwarded by looking up the routing table.

→ In an IP n/w nodes are routers.

→ It consists of necessary info. to forward packets along the best path towards destination.

→ Entries consist of:
① Network ID
② Subnet Mask.
③ Next Hop
④ Outgoing Interface / Destination N/W
⑤ Metric.

→ Routing tables can be maintained Manually / Dynamically.

→ Dynamically → with the help of Routing protocols.

→ The Routing table must be updated.

→ R.T tells the best path to the Router

→ It also ~~[illegible]~~ helps the Router in managing traffic.

Destination	N/W	Hop	Metric.

ANKIT Date ______ Page ______

(*) Packet Flow:

(a) Ingress Packet Processing:

→ When an IP packet arrives at a network, it contains enough information for route lookups after encapsulation.
→ This packet is sent to forwarding engine in the line card.
→ Forwarding engine searches a table (forwarding table) to determine next hop.
→ The next hop info contains egress line card and the outgoing port.
→ On completion of other functions, this packet is sent to the backplane interface.
→ It contains info. that helps the backplane interface to figure out to which interface card it is to be sent.
→ It then schedules the packet for transmission.

(b) Egress Packet Processing:

→ When the packet reaches egress line card, the backplane interface stores it in the memory.
→ Meanwhile the packet is updated with the address of new memory location and sent to the buffer queue.
→ In the buffer queue, the packets are transmitted out of the queue according to their priorities.
→ If the queue is full and there is a problem of congestion the queue manager drops the packet with low priority.
→ Finally the packet arrives at the n/w interface and then the TTL value and checksum is updated.
→ Packet is transferred to appropriate Port.

(x). Packet Processing:

The tasks performed by the router can be divided into:
(a). Time Critical
(b). Non time Critical

Time critical tasks can be broadly grouped into header processing & forwarding.
Header processing includes validation of packets, TTL, checksum calculation, etc.
Non - time critical: Maintanance, Management, Error Handling.

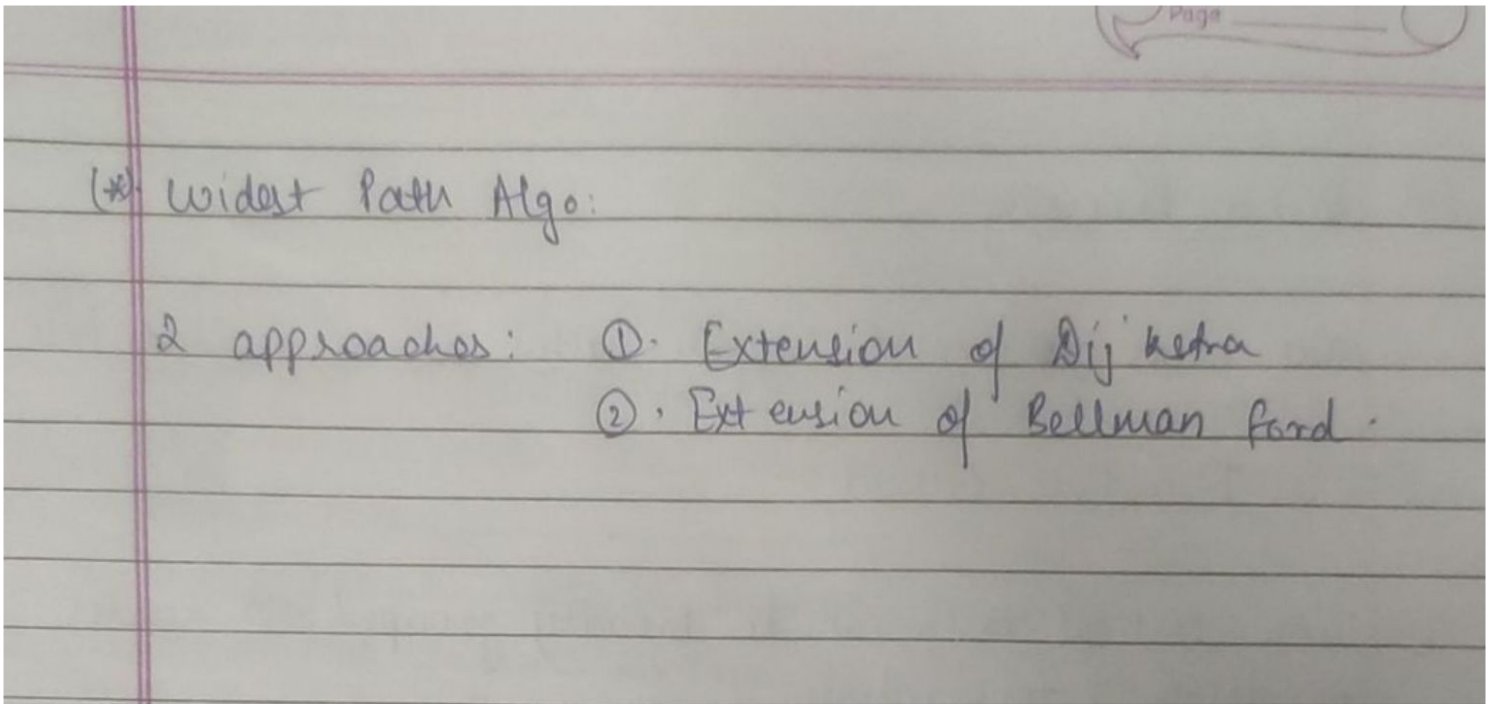
Time critical operations must have High priority under any circumstances.

Packets → CPU (route lookup) → destination

Fast Path functions Include: Header processing
Packet forwarding
Packet Classification
Packet Queuing & Scheduling

Slow Path func: ARP processing
Fragmentation
Reassembly
Advanced IP processing.

given more priority.



(*) Widest Path Algo:

2 approaches: ①. Extension of Dijkstra
②. Extension of Bellman ford.

(*). Basic forwarding functions

(a). IP Header Validation: Every IP packet needs to be validated. It ensures that only well-formed packets are processed further and rest are discarded. Also ensures that version is correct, Header length is valid & ~~also~~ matches checksum.

(b) Packet Lifetime Control: Routers must decrement TTL field in the IP packet header to prevent looping. If the TTL value is zero or negative, the packet is discarded.

(c) Checksum Recalculation: Since the value of TTL is changed, the value of checksum ~~so~~ must also be updated.

(d). Route Lookup: The destination address is used to search forwarding table for output. The result of this search will indicate whether the packet is destined to single port (unicast) or multiple ports (multicast).

(e). Fragmentation: If the Maximum transmission Unit value of output port is less than the size of packet, then the packet needs to be fragmented.

(f). Handling IP options: The presence of IP option field indicate that there are special processing needs for the packet. The router needs to support these needs.

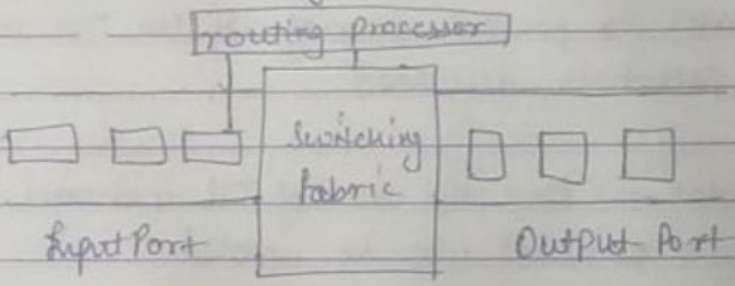
(*) Complex Forwarding Functions:

(a) Packet classification: The process of differentiating packets and taking necessary actions according to certain rules.

(b). Packet Translation: Router acts as a gateway for networks ~~[illegible]~~ to support Network Address Translation (NAT). This is done because IPV4 addresses are being exhausted.

(c). Traffic Prioritization: Applies priority to different packets for easy transmission of data packets.

(*). <u>Basic Architecture of a Router:</u>



→ Input Port: Interface by which packets are admitted into the router. It terminates the physical link at router.

$$\begin{tikzcd}\boxed{\text{Link termination}} \arrow[r] & \boxed{\text{Link layer functions}} \arrow[r] & \boxed{\text{Forwarding table lookup}}\end{tikzcd}$$

→ Switching fabric: Heart of the router. Connects I/P ports with O/P ports. It is a kind of network inside a networking device.
Implementation:
(a). Switching via memory: Processor copies the packet from I/P ports & sends to appropriate O/P port.
(b). Switching via bus: We have a bus that connects all I/P ports to all O/P ports
(c) Switching via Interconnection N/W: Instead of a single bus, we use ~~[illegible]~~ 2N bus to connect n ~~[illegible]~~ Input ports to n Output ports.

→ Output ports: Interface by which packets are transmitted out of router. Transmits the packet to outgoing link.

→ Routing Processor: Executes Routing Protocols. Employs various Routing Algorithms to prepare forwarding table.

(*). Routing table versus forwarding table.

Routing table	forwarding Table.
①. Process of finding path b/w two n/w based on their address.	①. Process of sending network data to its destination port.
② Used by routers to forward traffic from one n/w to another.	②. Used by devices such as switches/ bridges that process packet faster
③. Stores destination address for networks.	③ Responsible for storing next hop for each network
④. Contains path routing info.	④. Contains port info.
⑤. All routing tables are a form of forwarding tables.	⑤. forwarding tables are not a form of routing tables.
⑥. Contains all the paths to different destinations-	⑥. Contains only best path to every destination.

(*). Types of Routers: (core, edge, Enterprise)

(a). Core Routers: → Used for inter connecting a few thousand small networks.
→ Cost of moving traffic is shared among a large customer base
→ Capable of Handling large amount of traffic.
→ High speed & reliability are primary requirement.
→ With an increase in no. of systems connected, demand is placed on core routers to forward more packets per second.
→ Special algorithms are used for efficient and fast lookups.

→ form critical nodes in a N/w should not fail under any condition.

→ Software is enhanced so that when one of element fails, packet forwarding and Routing Protocols continue to function.

(b). Edge Routers: → Also known as access Routers.

→ deployed at the edge of the N/w for providing connectivity to customers.

→ Should be capable of Handling large amount of traffic.

→ Support a large number of ports.

(c). Enterprise Routers: → interconnect end systems located in companies, universities, etc.

→ Provide connectivity at low cost to a large no. of systems.

→ Many ethernet segments that are connected by Hubs, bridges & switches

→ Inexpensive devices. Can be easily installed.

→ Tends to degrade in performance as size of N/w increases.

→ They should support large no. of ports.

(v). Elements of Routers:

A generic Router consists of 8 major functional modules:

(a) Network Interfaces: → Contains many ports that provide connectivity to physical N/w links. Port serves as the entry & exit point for incoming & outgoing packets.

→ N/w Interfaces understands various data link Protocols so that when the packet arrives, it can decapsulate the packet.

→ It extracts the IP headers.

→ Encapsulates the packet b/w sending out ~~to~~ on the link

(b). forwarding Engines: → Responsible for deciding which n/w the incoming packet should be forwarded to.
→ On receiving a packet, it decapsulates ~~and~~ headers and sends the entire packet or just the packet header to forwarding engine.
→ forwarding engine consults a table and determines the n/w to which the packet should be sent. This table is forwarding table.

(c). Queue Manager: Provides buffer for temporary storage of packets when outgoing link is overbooked.
→ When these buffer queues overflow due to congestion, Queue Manager selectively drops packets.

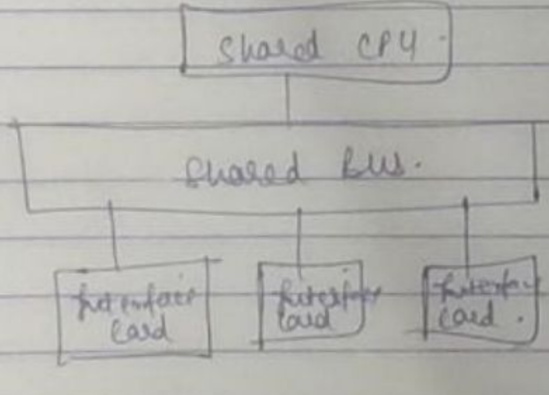
(d). Traffic Manager: Responsible for prioritizing and regulating outgoing traffic.
Sometimes the functionality of Queue Manager & Traffic Manager are merged into a single component.

(e). Back Plane: → Provides connectivity for N/W interfaces.
It can either be shared, where only 2 interfaces can communicate at any instance. or be switched, where multiple interfaces can communicate simultaneously.

(f). Route Control Processor: Responsible for implementing and executing routing protocols.
Maintains a routing table. From routing table, forwarding table is computed & updated.
It also handles errors.

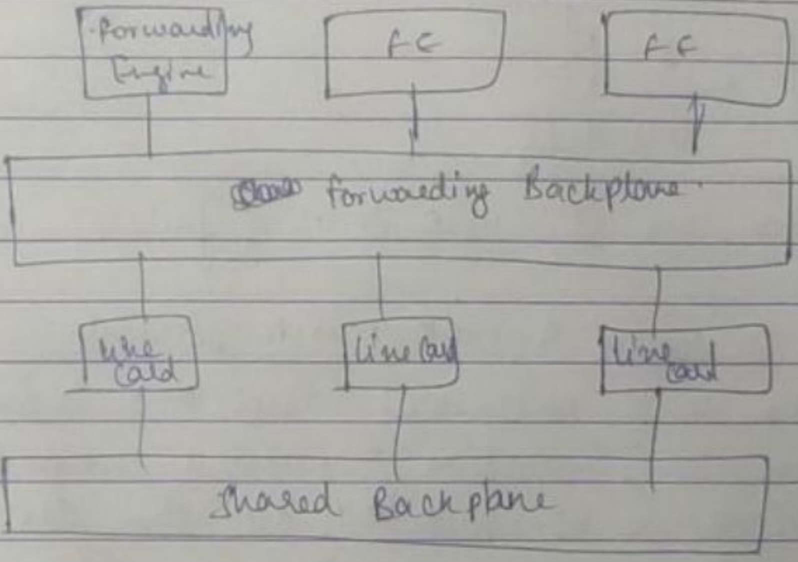
(v). New Router Architectures:

(a). Shared CPU Architecture → Similar to the Architecture of Conventional computers.
→ Shared backplane connects multiple line cards
→ functional Modules such as traffic manager, forwarding engine are implemented in software.
→ All the interface cards share CPU for the functions.
→ When packet is received at ingress interface, an interrupt is raised at the CPU.
→ Interrupt handler copies the packet to main memory of CPU where CPU performs route lookup for eagress port
→ finally it is written into buffer of relevant eagress interface.
→ Advantages: Simplicity & Implementation.
→ Disadvantage: lack of scalability.

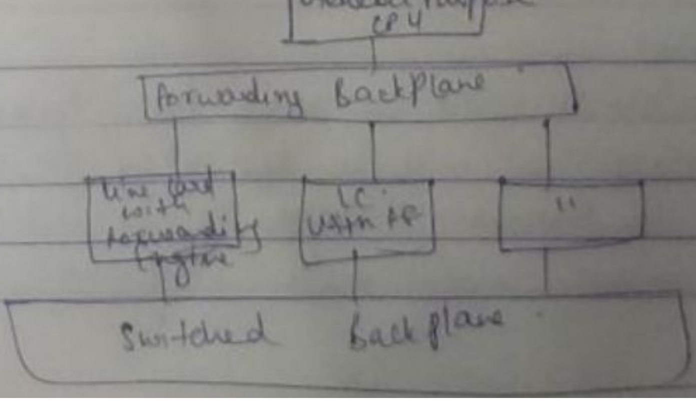


(b). Shared forwarding Engine architecture:
→ forwarding is faster than CPU architecture.
→ Each forwarding engine would have a different ~~process~~ dedicated processor that will perform route lookup.
→ forwarding engine has memory that can be used to store forwarding table.
→ Line cards are connected to each other via a ~~f~~ back plane so that

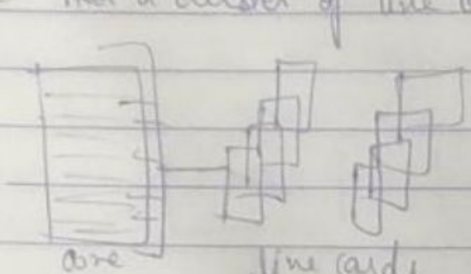
the packets can be transferred from one line card to another
→ The shared forwarding engines are connected to seperated backplane
→ Keeps the forwarding traffic seperate from regular traffic
This helps in achieving high throughput.
→ When a packet arrives, it is sent to forward engine for lookup
→ Advantage → Higher throuput rate.
→ Disadvantage → low bandwidth.



(*) Shared Nothing Architecture:
→ Does not have anything in common.
→ All the line cards have their own processing Hardware
→ When a packet arrives, it is sent to forwarding Engine.
→ FE determines appropriate egress interface and then the packet is sent to the buffer of egress via switched fabric



(*) Clustered Architecture : Used for increasing the no. of line cards.
→ A packet entering N/w interface in a line card depending on the result of route lookup may be destined to line card in same cluster ~~or~~ or line card in a different cluster.
→ The packet must be forwarded to the appropriate cluster.
Advantage → Add a cluster of line card as per need.



Disadvantage: Switch core is single point of failure.

(*) Impact of Addressing on lookup : 8- Not much efficient
→ Addressing Architecture is of fundamental importance to Routing Architecture
→ With Classful Addressing scheme, forwarding of packets is straightforward.
→ Routers only need to examine N/w part ~~to~~ of destination address to forward it to destination. Thus, forwarding table needs to store single entry. Such technique is called address aggregation & using prefixes to represent a group of addresses.
→ For classful addressing, the destination can be found using first few bits only. To make a correct match, routers must do more than just prefix matching because prefixes can be same for different addresses. It needs to find most specific match that is longest matching prefix.

(*) Longest Prefix Matching:
→ Algo used by routers to select an entry from forwarding table
→ lookups the IP prefixes that will be destination for next
→ Routers look at the destination address's IP prefix
→ The router implements longest match as follows:

① It receives a packet
② While processing header, it compares destination IP address bit by bit with the entries in routing table.
③ The entry that has longest no. of ~~[illegible]~~ N/w bits that match the IP destination address is the best match.

Example.

→ Router receives a packet with destination IP → 192.168.1.33
→ Routing Table contains.

192.168.1.32 /28
192.168.1.0 /24
192.168.0.0 /16.

To determine longest match convert IP address to binary & compare.

192.168.1.33 → 11000000. 10101000. 00000001. 00100001

Now match all the Routing Table Addresses with this.

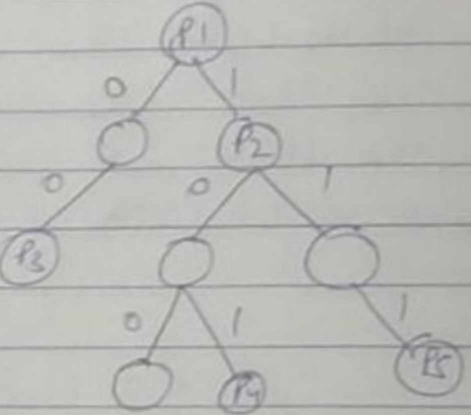
In this case 192.168.1.32/28 is the best match.

(*) Binary Tries

A trie,
→ Also called digital tree / prefix tree is a type of search tree.
• for locating specific keys within a set.
→ In order to access key, the trie is traversed depth-first.
→ A node's position defines the key with which it is associated.
→ Allow finding longest prefix that matches dest IP.
→ While traversing a node which is recently traversed is marked as prefix.

Prefix database

P1	*
P2	1*
P3	00*
P4	101*
P5	111*



complexity O(n)
n is the no. of prefixes.

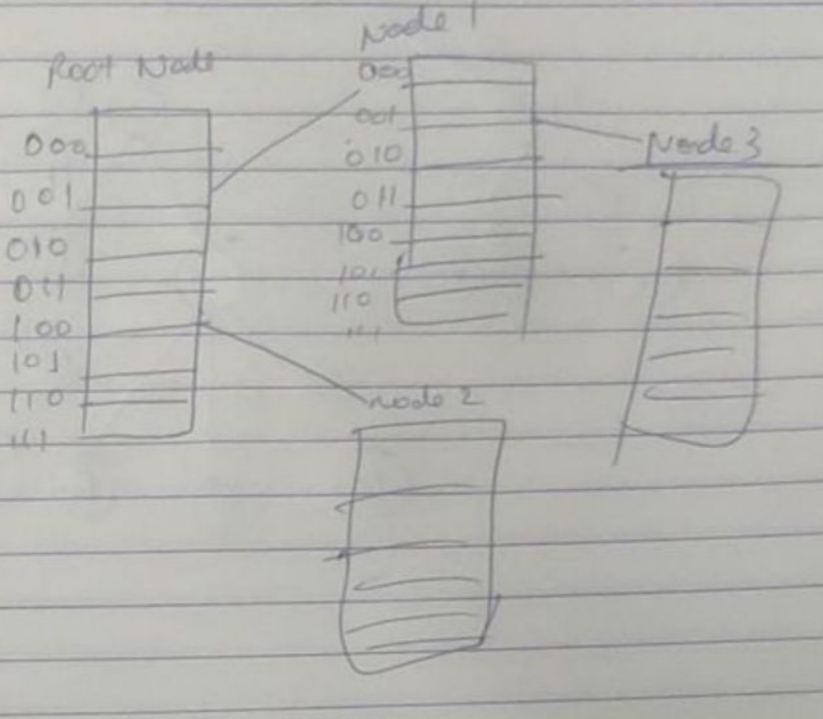
(*) ~~Multibit~~ tries:

→ Main principle is to examine several bits at a time (called a stride) in order to improve performance.
→ No. of bits to be inspected is called a stride.
→ strides can be either fixed or variable size.

Eg: Stride length = 3

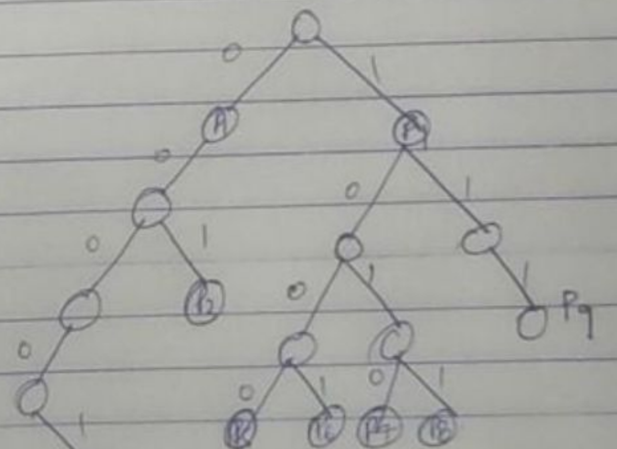
Prefix Database

P1	*
P2	1*
P3	00*
P4	101*
P5	111*
P6	1000*

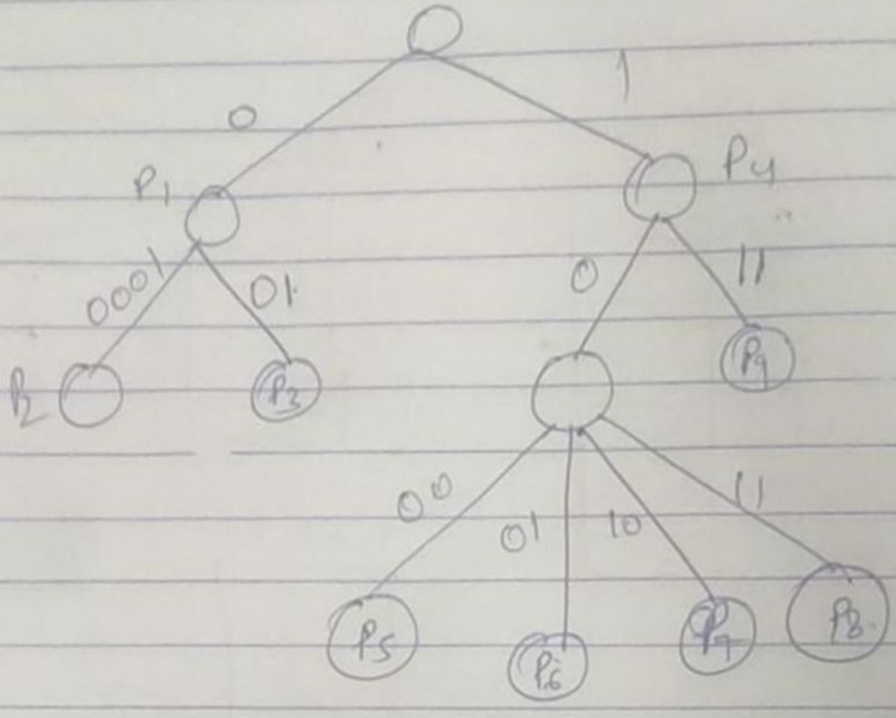


Compression multibit ~~[illegible]~~ Strides
→ ~~[illegible]~~ exists for multibit tries (compression algo)
→ multibit tries uses prefix expansion to reduce the no. of levels.
→ ↑ storage space

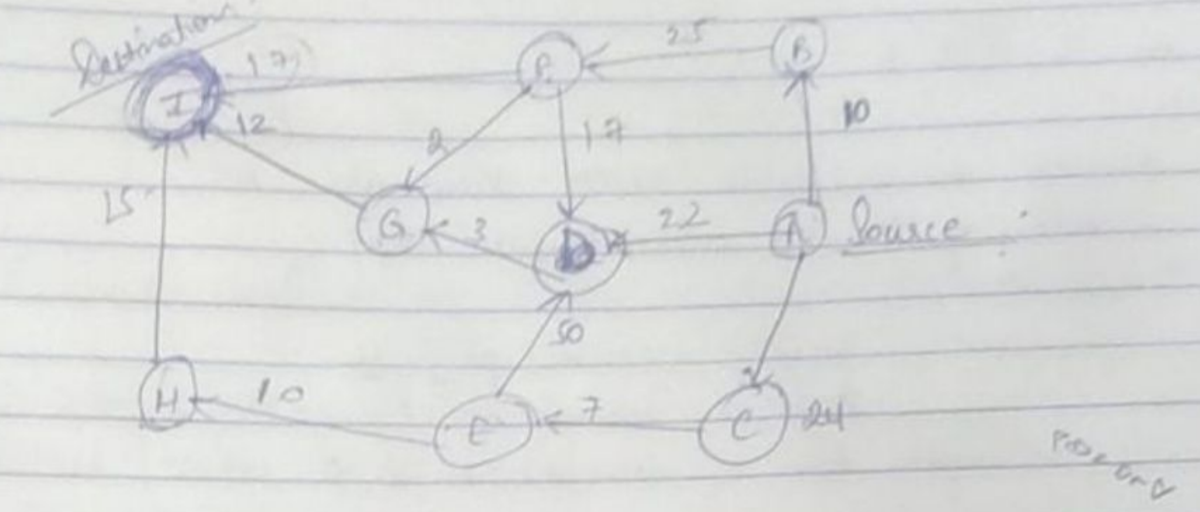
eg:



compressed



K - Shortest Path Algorithm: 1st Shortest Path + 2nd Shortest Path



A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	∞	∞	∞	∞
	10	24	22	∞	∞	∞	∞	∞
		24	22	∞	25	∞	∞	∞
		24		∞	25	3	∞	∞
		24		∞	25		∞	12
		24		∞	25		∞	

A	B	C	D	E	F	G	H	I
0	∞	∞	∞	∞	∞	∞	∞	∞
	10	24	22	∞	∞	∞	∞	∞
		24	22	∞	35	∞	∞	∞
		24		∞	35	25	∞	∞
				31	35	25	∞	∞
				31	35		∞	37
					35		41	37

$35 + 2 + \underline{12} = 49$

Nested Loops:

Iterate on I, G, D, A.
{
Iterate on reachable nodes those are not belonging to the shortest path itself.
}

from I, iterate on F, H
from G, iterate on F
from D, iterate on F, E.

In each iteration, we should compute the difference b/w the shortest distance from A to both vertex plus the length of the edge connecting them.

AF − AI + FI
35 − 37 + 15
− 2 + 17 = $\boxed{15}$

AH − AI + HI
~~[illegible]~~ 41 − 37 + 15
$\boxed{= 19}$

AF − AG + $\boxed{FG}$
35 − 25 + 2 = $\boxed{12}$

AF − AD + FD
35 − 22 + 17 = $\boxed{40}$

AE − AD + ED
31 − 22 + 50
= $\boxed{59}$

$$\begin{tikzcd} A \arrow[r] & \boxed{F \rightarrow G} \arrow[r] & I \end{tikzcd}$$

35 + $\boxed{12}$ + 12
49