Question Paper Solution

Branch : CS & IT | Semester: VI | Subject: TOC | Mid Term: I

Submitted By : Dr. Mukesh Gupta

Q1. If $D=(Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ is the DFA constructed from NFA $N=(Q_N, \Sigma, \delta_N, q_0, F_N)$ by subset construction, then $L(D)=L(N)$

Proof: By induction on $|w|$

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

Each of the $\hat{\delta}$ functions returns a set of states from Q_N, but $\hat{\delta}_D$ interprets this set as one of the states of Q_D (which is power set of Q_N), while $\hat{\delta}_N$ interprets this set as a subset of Q_N

Basis: $|w|=0$; $w=\varepsilon$, by basis definitions of $\hat{\delta}$ for DFA & NFA both $\hat{\delta}_D(\{q_0\}, \varepsilon)$ & $\hat{\delta}_N(q_0, \varepsilon)$ are $\{q_0\}$

Induction: Let $|w|=n+1$ & assume the statement for n. $w=xa$, where a is the final symbol of w. By the inductive hypothesis, $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x)$. Let both these sets of NFA states be $\{p_1, p_2, \dots, p_k\}$

Inductive part of the definition of $\hat{\delta}$ for NFA's tell us

$$\hat{\delta}_N(q_0, w) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \quad -(1)$$

The subset construction on the other hand tells that

$$\delta_D(\{p_1, p_2, \dots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \quad -(2)$$

therefore

$$\hat{\delta}_D(\{q_0\}, w) = \delta_D(\hat{\delta}_D(\{q_0\}, x), a) = \delta_D(\{p_1, \dots, p_k\}, a)$$

$$= \bigcup_{i=1}^{k} \delta_N(p_i, a) \quad -(3)$$

eq. (1) & (3) shows that $\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$.

When we observe that DFA & NFA both accept w iff $\hat{\delta}_D(\{q_0\}, w)$ or $\hat{\delta}_N(q_0, w)$ respectively respectively, contain a state in F_N.

Question Paper Solution

Branch : CS & IT | Semester: VI | Subject: TOC | Mid Term: I

Submitted By : Dr. Mukesh Gupta

NDFA to DFA Conversion

State	I/P 0	1
→A	{A,B}	{A}
B	{C}	{C}
C	{D}	{D}
D	ϕ	ϕ

Respective DFA table By using subset construction.

State	I/P 0	1
→ {A}	{A,B}	{A}
{A,B}	{A,B,C}	{A,C}
{A,B,C}	{A,B,C,D}	{A,C,D}
{A,C}	{A,B,D}	{A,D}
* {A,B,C,D}	{A,B,C,D}	{A,C,D}
* {A,C,D}	{A,B,D}	{A,D}
* {A,D}	{A,B}	{A}

→ Initial State

* Final State

Question Paper Solution

Branch : CS & IT | Semester: VI | Subject: TOC | Mid Term: I

Submitted By : Dr. Mukesh Gupta

Q2. Difference between Mealy & Moore machine.

Mealy machine :- It is an FSM whose o/p depends on the present state as well the present i/p

6 tuples $(Q, \Sigma, S, q_0, \Delta, \lambda, q_0)$

Q is the finite set of states,
Σ: I/P alphabet
Δ: o/p alphabet
S: $\Sigma \times Q \to Q$
λ: o/p function $\Sigma \times Q \to \Delta$
q_0: initial state

Moore machine:
$(Q, \Sigma, \Delta, S, \lambda, q_0)$

- only difference in in o/p function
  λ: $Q \to \Delta$

Mealy Machine	Moore Machine
(a) o/p depends on both the present state & present i/p	o/p depends on the present state only
(b) Generally it has fewer states than moore m/c	More states than Mealy m/c
(c) Mealy m/cs react faster to i/ps. They generally read in the same clock cycle.	In moore m/cs, more logic is required to decode the o/p resulting in more circuit delays. They generally react one clock cycle later
(d) From presentation point of view, output is placed on transition.	output is placed on state

Question Paper Solution

Branch : CS & IT | Semester: VI | Subject: TOC | Mid Term: I

Submitted By: Dr. Mukesh Gupta

Q. Mealy m/c transition table

Present State	Next State I/P a=0 State	O/P	I/P a=1 State	O/P
→ q_0	q_1	N	q_2	N
q_1	q_1	Y	q_2	N
q_2	q_2	Y	q_1	N

The transition table for Moore M/c

present State	Next State a=0	a=1	O/P
→ q_0	q_{1N}	q_{2N}	Y/N
q_{1N}	q_{1Y}	q_{2N}	N
q_{1Y}	q_{1Y}	q_{2N}	Y
q_{2N}	q_{2Y}	q_{1N}	N
q_{2Y}	q_{2Y}	q_{1N}	Y

Q.

Question Paper Solution

Branch : CS & IT | Semester: VI | Subject: TOC | Mid Term: I

Submitted By : Dr. Mukesh Kr. Gupta

Q3 Noam Chomsky defined 4 types of grammar

type 0:- Most general / unrestricted / phrase structure

$U \to V \quad U \in V^* N V^*$

where $V = (N \cup T)$ N is variable, T is terminal

$V \in V^*$

(At least one Non terminal on LHS)

type 1 :- $|U| \leq |V|$ also called length increasing grammar equivalent to context ~~sent~~ sensitive grammar $\alpha A \beta \to \alpha \gamma \beta$

$\alpha, \beta : \in V^*$

A is rewritten as γ in the context of α & β, that's why its context sensitive.

ε cannot occur on the RHS because $|U| \leq |V|$ in type 1.

If we want to allow ε in the grammar then rule will be $S \to \varepsilon$, then S doesn't occur on the RHS of any production rule

Type 2:- $A \to \alpha$

$A \in N, \alpha \in (V)^*$

Such a rule is called ~~a~~ context free.

Most of the arithmetic expressions can be achieved by CFG. For ex:-

$E \to E+E$
$E \to E*E$
$E \to (E)$
$E \to id$

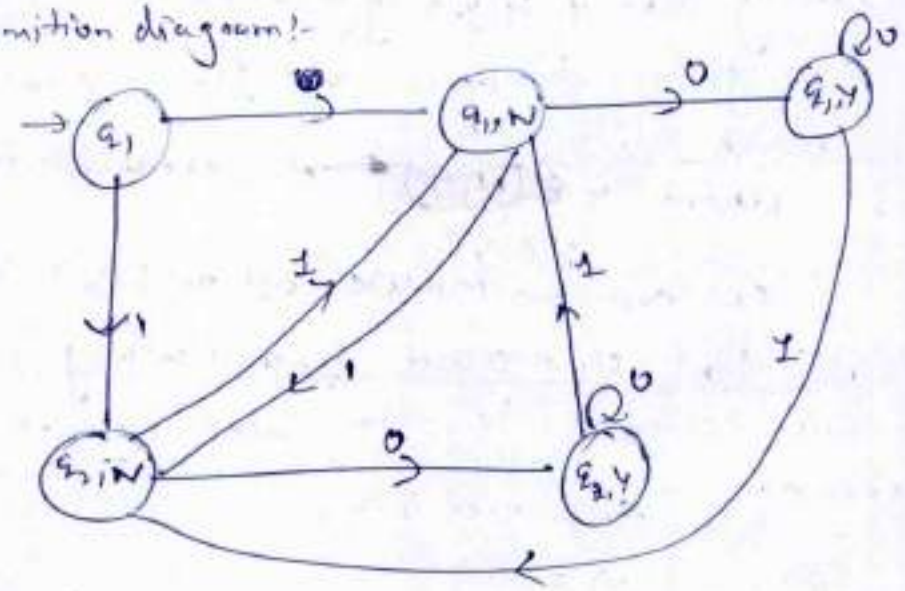
} CFG generating all arithmetic expression using +, *

Question Paper Solution

Branch : CS & IT | Semester: VI | Subject: TOC | Mid Term: I

Submitted By : Dr. Mukesh Gupta

Transition diagram:-



Q3. Equivalence classes will be

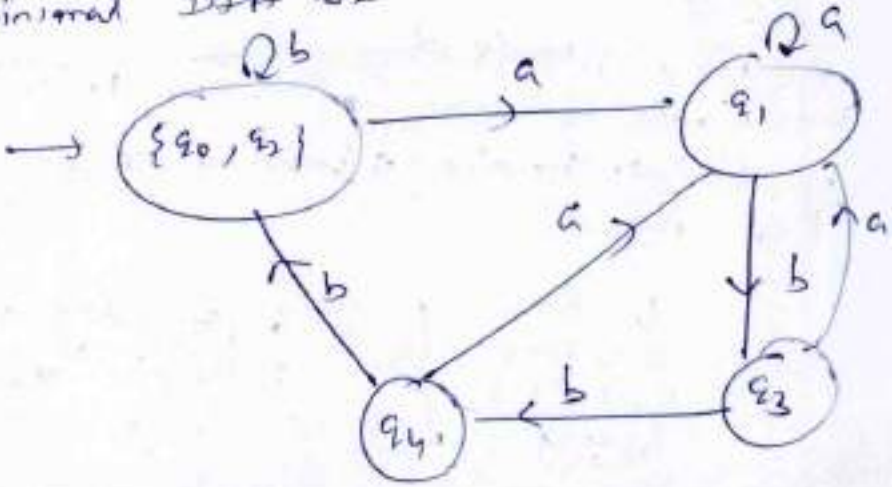
$\pi_0 = \{\{q_0, q_1, q_2, q_3\}, \{q_4\}\}$

$\pi_1 = \{\{q_0, q_1, q_2\}, \{q_3\}, \{q_4\}\}$

$\pi_2 = \{\{q_0, q_2\}, \{q_1\}, \{q_3\}, \{q_4\}\}$

$\pi_3 = \{\{q_0, q_2\}, \{q_1\}, \{q_3\}, \{q_4\}\}$

The minimal DFA is:



Question Paper Solution

Branch : CS & IT | Semester: VI | Subject: TOC | Mid Term: I

Submitted By : Dr. Mukesh Kr. Gupta

Type 3 :- $A \to aB$ $\quad i'A, B \in N$
$A \to b$ $\quad a \in T$
$b \in T \cup \{\varepsilon\}$

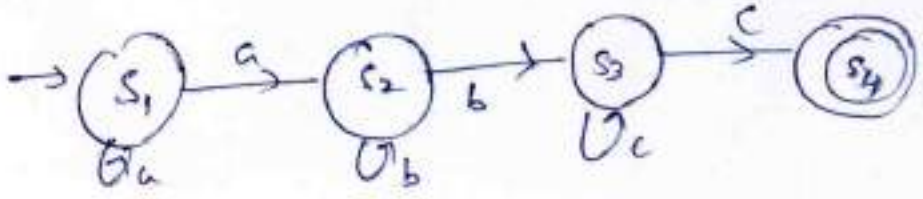
It's Regular Grammar
Right extension is left linear, Right linear but they are equivalent in power

(b) Find the Regular Grammar for the language

$$L = \{a^i b^j c^k \mid i,j,k >= 1\}$$

The Grammar Rule will be

$S_1 \to aS_1 \mid aS_2$
$S_2 \to bS_2 \mid bS_3$
$S_3 \to cS_3 \mid cS_4$
$S_4 \to \varepsilon$



Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017

Question Paper Solution

Branch : Information Technology Semester: VI Subject: Computer Networks Mid Term: I

Submitted By :Kajal Mathur/Basant Agarwal

Q.1 Discuss the reason of congestion in a network. What are the possible techniques to control congestion in network? Also discuss Leaky Buket alogorithm in detail.

Answer

As Internet can be considered as a Queue of packets, where transmitting nodes are constantly adding packets and some of them (receiving nodes) are removing packets from the queue. So, consider a situation where too many packets are present in this queue (or internet or a part of internet), such that constantly transmitting nodes are pouring packets at a higher rate than receiving nodes are removing them. This degrades the performance, and such a situation is termed as Congestion.

causes of congestion

Congestion can occur due to several reasons. For example, if all of a sudden a stream of packets arrive on several input lines and need to be out on the same output line, then a long queue will be build up for that output. If there is insufficient memory to hold these packets, then packets will be lost (dropped).

Slow processors also cause Congestion. If the router CPU is slow at performing the task required for them (Queuing buffers, updating tables, reporting any exceptions etc.), queue can build up even if there is excess of line capacity.

Similarly, Low Bandwidth lines can also cause congestion. Upgrading lines but not changing slow processors, or vice-versa, often helps a little; these can just shift the bottleneck to some other point. The real problem is the mismatch between different parts of the system.

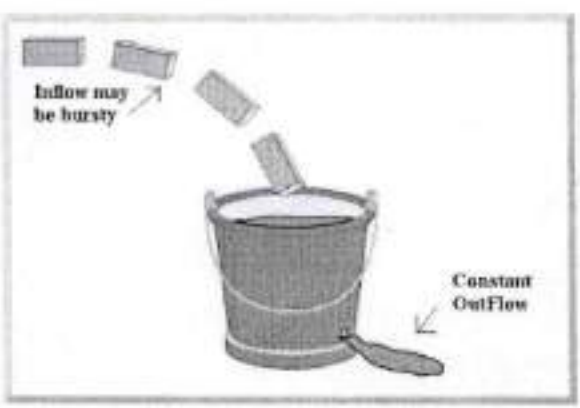
Congestion tends to feed upon itself to get even worse. Routers respond to overloading by dropping packets. When these packets contain TCP segments, the segments don't reach their destination, and they are therefore left unacknowledged, which eventually leads to timeout and retransmission. So, the major cause of congestion is often the bursty nature of traffic. If the hosts could be made to transmit at a uniform rate, then congestion problem will be less common and all other causes will not even led to congestion because other causes just act as an enzyme which boosts up the congestion when the traffic is bursty (i.e., other causes just add on to make the problem more serious, main cause is the bursty traffic).

congestion control techniques

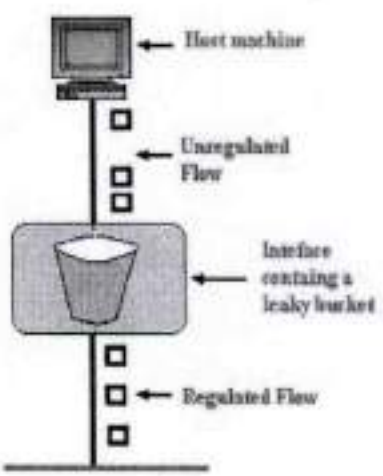
Congestion control refers to the mechanisms and techniques used to control congestion and keep the traffic below the capacity of the network. As shown in Fig. 7.5.2, the congestion control techniques can be broadly classified two broad categories:

- Open loop: Protocols to prevent or avoid congestion, ensuring that the system (or network under consideration) never enters a Congested State.
- Close loop: Protocols that allow system to enter congested state, detect it, and remove it.

Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket, the rate at which water comes out from that small hole is constant. This scenario is depicted in figure . Once the bucket is full, any additional water entering it spills over the sides and is lost (i.e. it doesn't appear in the output stream through the hole underneath). The same idea of leaky bucket can be applied to packets, as shown in Fig.



(a)



(b)

Question Paper Solution

Branch : Information Technology Semester: VI Subject: Computer Networks Mid Term: I

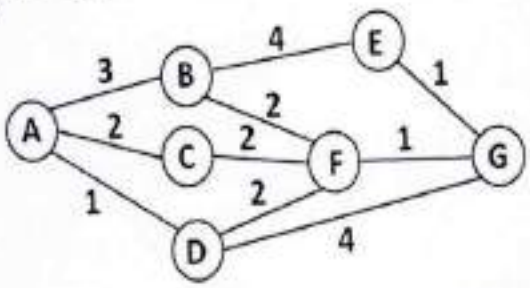
Submitted By :Kajal Mathur/Basant Agarwal

Conceptually each network interface contains a leaky bucket. And the following steps are performed:

- When the host has to send a packet, the packet is thrown into the bucket.
- The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
- Bursty traffic is converted to a uniform traffic by the leaky bucket.
- In practice the bucket is a finite queue that outputs at a finite rate.

This arrangement can be simulated in the operating system or can be built into the hardware. Implementation of this algorithm is easy and consists of a finite queue. Whenever a packet arrives, if there is room in the queue it is queued up and if there is no room then the packet is discarded.

Q.1 What do you understand by routing? What do you understand by routing? The network shown in Figureuses a Link State Routing protocol. Construct a Shortest Path Tree for node A, using Dijkstra's algorithm. (1+4)



Routing is the act of moving information across an inter-network from a source to a destination. Along the way, at least one intermediate node typically is encountered. It's also referred to as the process of choosing a path over which to send the packets. Routing is often contrasted with bridging, which might seem to accomplish precisely the same thing to the casual observer. The primary difference between the two is that bridging occurs at Layer 2 (the data link layer) of the OSI reference model, whereas routing occurs at Layer 3 (the network layer). This distinction provides routing and bridging with different information to use in the process of moving information from source to destination, so the two functions accomplish their tasks in different ways. The routing algorithm is the part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on, i.e. what should be the next intermediate node for the packet. Routing protocols use metrics to evaluate what path will be the best for a packet to travel. A metric is a standard of measurement; such as path bandwidth, reliability, delay, current load on that path etc; that is used by routing algorithms to determine the optimal path to a destination. To aid the process of path determination, routing algorithms initialize and maintain routing tables, which contain route information. Route information varies depending on the routing algorithm used. Routing algorithms fill routing tables with a variety of information. Mainly Destination/Next hop associations tell a router that a particular destination can be reached optimally by sending the packet to a particular node representing the "next hop" on the way to the final destination. When a router receives an incoming packet, it checks the destination address and attempts to associate this address with a next hop. Some of the routing algorithm allows a router to have multiple "next hop" for a single destination depending upon best with regard to different metrics. For example, let's say router R2 is be best next hop for destination "D", if path length is considered as the metric; while Router R3 is the best for the same destination if delay is considered as the metric for making the routing decision. Routing algorithms can be classified based on the following criteria:

- Static versus Adaptive
- Single-path versus multi-path
- Intra-domain versus inter-domain
- Flat versus hierarchical
- Link-state versus distance vector
- Host-intelligent versus router-intelligent

Since we are using a Link State protocol, we need to apply Dijkstra's algorithm to solve the shortest path problem:

Question Paper Solution

Branch : Information Technology Semester: VI Subject: Computer Networks Mid Term: I

Submitted By :Kajal Mathur/Basant Agarwal

Iter	*M*	B	C	D	E	F
1	A	2 AB	1 AC	∞ -	∞ -	∞ -
2	AC	2 AB	1 AC	4 ACD	3 ACE	6 ACF
3	ACB	2 AB	1 AC	4 ACD	3 ACE	6 ACF
4	ACBE	2 AB	1 AC	4 ACD	3 ACE	6 ACF
5	ACBED	2 AB	1 AC	4 ACD	3 ACE	5 ACDF
6	ACBEDF	2 AB	1 AC	4 ACD	3 ACE	5 ACDF

Q.2 Suppose original IP Datagram shown below is moving in network.

Original IP Datagram

Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0	345	5140	0	0	0

When it is pass on the Ethernet network (MTU=1500 bytes). Find out the no of fragments and respective offset values because of Ethernet network.(Fragments information must be in the same format as original datagram)

Answer

IP Fragments (Ethernet)

Sequence	Identifier	Total Length	DF May / Don't	MF Last / More	Fragment Offset
0-0	345	1500	0	1	0
0-1	345	1500	0	1	185
0-2	345	1500	0	1	370
0-3	345	700	0	0	555

OR

Q.2 Describe IPV4 packet header format.

- **IHL:** Internet Header Length; Length of entire IP header.
- **Total Length:** Length of entire IP Packet (including IP header and IP Payload).
- **Fragment Offset:** This offset tells the exact position of the fragment in the original IP Packet.
- **Header Checksum:** This field is used to keep checksum value of entire header which is then used to check if the packet is received error-free.

Question Paper Solution

Branch : Information Technology Semester: VI Subject: Computer Networks Mid Term: I

Submitted By :Kajal Mathur/Basant Agarwal

IPv4	IPv6
The Adress Space is 32 bits	The space is 128 bits.
The length of header is 20 bytes	The length of header is 40
4 bytes for each adress in the header	16 bytes for each adressin the header
The number of Header field 12	The number of header field 8
Checksum field, used to measure error in the headerrequired	Checksum field eliminated from header as error in the IP header are of very crucial
Internet Protocol Security (IPSec) with repect to network security is optional	Internet Protocol Seculrty (IPSec) With respect to net work seculrty is mandatory
No identification to the packet flow (Lack of QoS handing)	The flow level field on the header portion identifies the packet flow and directs to router (Efficient QoS handing)
The Fargmentation is done both by sending host and routers	The framentation is done both by sending hoost; there is no role of the routers.
No identification to the packet flow (Lack of QoS handing)	The flow level field on the header portion identifies the packet flow and directs to router (Efficient QoS handing)
Clients have approach Dynamic Host Configuration server (DHCS) whenever they connect to an network.	Clients do not have to approach any such server as they are given permanent adresses.

Q.3 Explain sub netting in network. 3 Explain sub netting in network. A company is granted a site address 201.70.64.0. The company needs Six subnets. Design the subnets.

Answer

A **subnet** allows the flow of network traffic between hosts to be segregated based on a network configuration. By organizing hosts into logical groups, subnetting can improve network security and performance.

Subnetting in Practice

Subnetting works by applying the concept of extended network addresses to individual computer (and other network device) addresses. An extended network address includes both a network address and additional bits that represent the subnet number. Together, these two data elements support a two-level addressing scheme recognized by standard implementations of IP.

The network address and subnet number, when combined with the host address, therefore support a three-level scheme.

Consider the following real-world example. A small business plans to use the 192.168.1.0 network for its internal (intranet) hosts. The human resources department wants their computers to be on a restricted part of this network because they store payroll information and other sensitive employee data. But because this is a Class C network, the

default subnet mask of 255.255.255.0 allows all computers on the network to be peers (to send messages directly to each other) by default.

The first four bits of 192.168.1.0 -

1100

place this network in the Class C range and also fix the length of the network address at 24 bits. To subnet this network, more than 24 bits must be set to '1' on the left side of the subnet mask. For instance, the 25-bit mask 255.255.255.128 creates a two-subnet network as shown in Table 1.

For every additional bit set to '1' in the mask, another bit becomes available in the subnet number to index additional subnets. A two-bit subnet number can support up to four subnets, a three-bit number supports up to eight subnets, and so on.

Question Paper Solution

Branch : Information Technology Semester: VI Subject: Computer Networks Mid Term: I/~~II/Extra/Imp~~
Submitted By : Kajal Mathur / Barani Agarwal

Ans: Since the given address is a class C network, so, the default mask is 255.255.255.0

- As we need 6 subnets, we need three entries ($2^n = 6$ for $n=3$ we have 8 subnets from which we can choose any (6)
- So subnet mask is 255.255.255.~~224~~ as.

11111111.11111111.11111111.11100000
↳ 3 entries for subnets

- We required only 6 subnets among 8 (possible). The six subnets are as -

Combination	Subnet number	Address
000	Subnet 1	201.70.64.0 to 201.70.64.31
001	Subnet 2	201.70.64.32 to 201.70.64.63
010	Subnet 3	201.70.64.64 to 201.70.64.95
011	Subnet 4	201.70.64.96 to 201.70.64.127
100	Subnet 5	201.70.64.128 to 201.70.64.159
101	Subnet 6	201.70.64.160 to 201.70.64.191

- Remaining unused subnets are created by using 110 & 111

Question Paper Solution

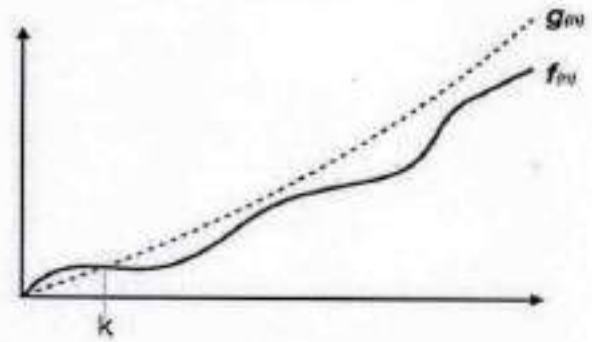
Branch : I I Semester: 6 Sem Subject: D A A Mid Term: I/II/Extra/Imp.

Submitted By : NEHA MATHUR

Asymptotic Notations are languages that allow us to analyze an algorithm's running time by identifying its behavior as the input size for the algorithm increases. This is also known as an algorithm's growth rate.

Big-O notation:

Big-O, commonly written as **O**, is an Asymptotic Notation for the worst case, or ceiling of growth for a given function. It provides us with an **asymptotic upper bound** for the growth rate of runtime of an algorithm. Say f(n) is your algorithm runtime, and g(n) is an arbitrary time complexity you are trying to relate to your algorithm. f(n) is O(g(n)), if for some real constants c ($c > 0$) and n_0, $f(n) <= c\ g(n)$ for every input size n ($n > n_0$).



For example, for a function ***f*(n)**

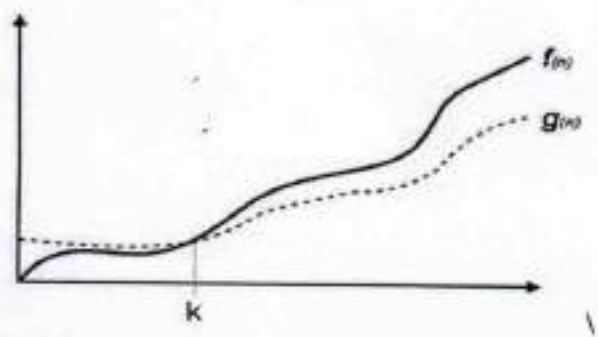
$O(f(n)) = \{ g(n) : \text{there exists } c > 0 \text{ and } n_0 \text{ such that } f(n) \leq c.g(n) \text{ for all } n > n_0. \}$

Big-Omega (Ω) notation:

Big-Omega, commonly written as **Ω**, is an Asymptotic Notation for the best case, or a floor growth rate for a given function. It provides us with an **asymptotic lower bound** for the growth rate of runtime of an algorithm.

f(n) is Ω(g(n)), if for some real constants c ($c > 0$) and n_0 ($n_0 > 0$), f(n) is $>= c\ g(n)$ for every input size n ($n > n_0$).

The asymptotic growth rates provided by big-O and big-omega notation may or may not be asymptotically tight. Thus we use small-o and small-omega notation to denote bounds that are not asymptotically tight.



For example, for a function ***f*(n)**

Question Paper Solution

Branch : IT Semester: 6 Subject: DAA Mid Term: I/II/Extra/Imp.

Submitted By : NEHA MATHUR

$\Omega(f(n)) \geq \{ g(n) : \text{there exists } c > 0 \text{ and } n_0 \text{ such that } g(n) \leq c.f(n) \text{ for all } n > n_0. \}$

Small-o notation:

Small-o, commonly written as **o**, is an Asymptotic Notation to denote the upper bound (that is not asymptotically tight) on the growth rate of runtime of an algorithm.

f(n) is o(g(n)), if for some real constants c (c > 0) and n_0 (n_0 > 0), f(n) is < c g(n) for every input size n (n > n_0).

The definitions of O-notation and o-notation are similar. The main difference is that in f(n) = O(g(n)), the bound f(n) <= g(n) holds for **some** constant c > 0, but in f(n) = o(g(n)), the bound f(n) < c g(n) holds for **all** constants c > 0.

Small-omega notation:

Small-omega, commonly written as **ω**, is an Asymptotic Notation to denote the lower bound (that is not asymptotically tight) on the growth rate of runtime of an algorithm.

f(n) is ω(g(n)), if for some real constants c (c > 0) and n_0 (n_0 > 0), f(n) is > c g(n) for every input size n (n > n_0).

The definitions of Ω-notation and ω-notation are similar. The main difference is that in f(n) = Ω(g(n)), the bound f(n) >= g(n) holds for **some** constant c > 0, but in f(n) = ω(g(n)), the bound f(n) > c g(n) holds for **all** constants c > 0.

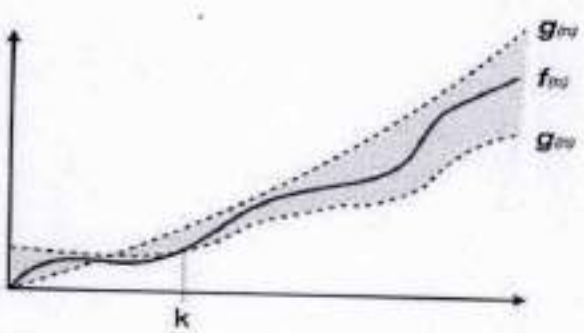
Theta (Θ) notation:

Theta, commonly written as **Θ**, is an Asymptotic Notation to denote the **asymptotically tight bound** on the growth rate of runtime of an algorithm.

f(n) is Θ(g(n)), if for some real constants c1, c2 and n_0 (c1 > 0, c2 > 0, n_0 > 0), c1 g(n) is < f(n) is < c2 g(n) for every input size n (n > n_0).

∴ f(n) is Θ(g(n)) implies f(n) is O(g(n)) as well as f(n) is Ω(g(n)).

Feel free to head over to additional resources for examples on this. Big-O is the primary notation use for general algorithm time complexity.



$\Theta(f(n)) = \{ g(n) \text{ if and only if } g(n) = O(f(n)) \text{ and } g(n) = \Omega(f(n)) \text{ for all } n > n_0. \}$

Question Paper Solution

Branch : I.T. Semester: 6 Subject: DAA Mid Term: I/II/Extra/Imp.

Submitted By : NEHA MATHUR

OR

Q1: Solve the recurrence relation

(i) $T(n) = 2T(n/2) + 3n$

Solving by using Master Method, as recurrence relation is of form $T(n) = aT\left(\frac{n}{b}\right) + f(n)$,

where $a = 2$, $b = 2$ and $f(n) = 3n$

$\Rightarrow \log_b a = \log_2 2 \Rightarrow 1$

$\therefore n^{\log_b a} = n^1 \Rightarrow n$

$f(n) = n$

Since $n^{\log_b a} = kf(n)$,

This supports Case 2, when $n^{\log_b a}$ and $f(n)$ have same order, then we view the algorithm as $f(n)$ number of steps followed by some c number of recursive calls

$\therefore T(n) = \Theta(n^{\log_b a} \log n)$

$\boxed{\therefore T(n) = \Theta(n \log n)}$

(ii) $T(n) = 4T(n/2) + n^2$

Solving by using Master Method, as recurrence relation is of form $T(n) = aT(n/b) + f(n)$

where $a = 4$, $b = 2$ and $f(n) = n^2$

$\Rightarrow \log_b a = \log_2 4 \Rightarrow \log_2 (2)^2 \Rightarrow 2$

Question Paper Solution

Branch : I.T. Semester: 6 Subject: DAA Mid Term: I/II/Extra/Imp.

Submitted By : NEHA MATHUR

$\therefore n^{\log_b a} = n^2$

and $f(n) = n^2$

Since $n^{\log_b a} = f(n)$

This supports Case 2 : when $n^{\log_b a}$ and $f(n)$ have same order, then we view the algorithm as $f(n)$ number of steps followed by same number of recursive calls.

$\therefore T(n) = \Theta(n^{\log_b a} \log n)$

$\Rightarrow T(n) = \Theta(n^2 \log n)$

Q2 Schedule the following jobs so as to have maximum profit

	Job1	Job2	Job3	Job4	Job5	Job6	Job7	Job8
Deadline d	2	1	3	2	4	1	3	3
Profit p	10	15	8	20	9	12	16	11

Soln Using the algorithm Job-Greedy-1:

(1) for t = 1 to max-deadline repeat Step 2 to Step 3

(2) Prepare a set of jobs with d[i] = t

(3) select the job with maximum profit

(4) Exit

Question Paper Solution

Branch : I.T. Semester: 6 Subject: DAA Mid Term: I/II/Extra/Imp.
Submitted By : NEHA MATHUR

At $t=1$, $d=t=1$, the jobs with this deadline are J_2 and J_6
→ Compare their profits, $P_2 > P_6$, select J_2.
⇒ $J = \{J_2\}$ [Here J is a set of jobs selected]
$t=2$, for $d_i = t = 2$, J_1 and J_4 have this deadline
Since $P_1 < P_4$, select J_4
$J = \{J_2\} \cup \{J_4\} = \{J_2, J_4\}$ ∴
$t=3$, for $d=t=3$, J_3 J_7 & J_8 have this deadline
Maximum profit P_7, hence select J_7.
∴ $J = \{J_2, J_4, J_7\}$
$t=4$, for $d=t=4$ only J_5 has this deadline, so directly select
∴ $J = \{J_2, J_4, J_7, J_5\}$
The job sequence is $J_2 \rightarrow J_4 \rightarrow J_7 \rightarrow J_5$
Total profit $= P_2 + P_4 + P_7 + P_5$
$= 15 + 20 + 16 + 9 = 60$

Using the Job - Greedy - 2 algorithm

1. Arrange the job in descending order of profits.
2. Put first job in set of jobs J, $J := \{J_1\}$
3. $t := 2$;
4. for $i := 2$ to n Repeat step 5 for all jobs J_i.
5. if $[d_i] >= t$
6. $J := J \cup \{J_i\}$;

Question Paper Solution

Branch : I.T. Semester: 6 Subject: DAA Mid Term: I/II/Extra/Imp.

Submitted By : NEHA MATHUR

(7) $t := t+1$;

(8) Exit

First we arrange the jobs according to their profit values:

$P = \{20, 16, 15, 12, 11, 10, 9, 8\}$

$d = \{2, 3, 1, 1, 3, 2, 4, 3\}$

Now, Starting with $t=2$ and $i=2$, check $d[2] > t$, yes

$\therefore J = \{J_1, J_2\}$ and $t = t+1 = 3$

Now, $t=3$ and $i=3$, check $d[3] >= t$, no

$t=3$ and $i=4$, check $d[4] >= t$, no

$t=3$ and $i=5$, check $d[5] >= t$, yes

$\therefore J = \{J_1, J_2, J_5\}$ and $t = t+1 = 4$

Now, $t=4$ and $i=6$, check $d[6] >= t$, no

$t=4$ and $i=7$, check $d[7] >= t$, yes

$\therefore J = \{J_1, J_2, J_5, J_7\}$, and $t = t+1 = 5$

$t=5$ and $i=8$, check $d[8] >= t$, no

The algorithm stops here and the schedule obtained is $J_1 \rightarrow J_2 \rightarrow J_5 \rightarrow J_7$, for which the total profit $= p_1 + p_2 + p_5 + p_7$

$= 20 + 16 + 11 + 9 = 56$

Question Paper Solution

Branch : I.T Semester: 6 Subject: DAA Mid Term: I/II/Extra/Imp.

Submitted By : NEHA MATHUR

OR

Q2 Optimal Merge Pattern

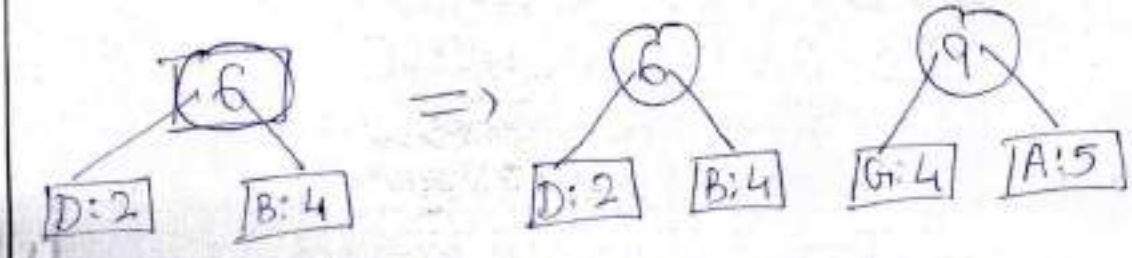
A	B	C	D	E	F	G	H
5	4	7	2	9	11	4	8

arrange them in ascending order

D	B	G	A	C	H	E	F
2	4	4	5	7	8	9	11

=>

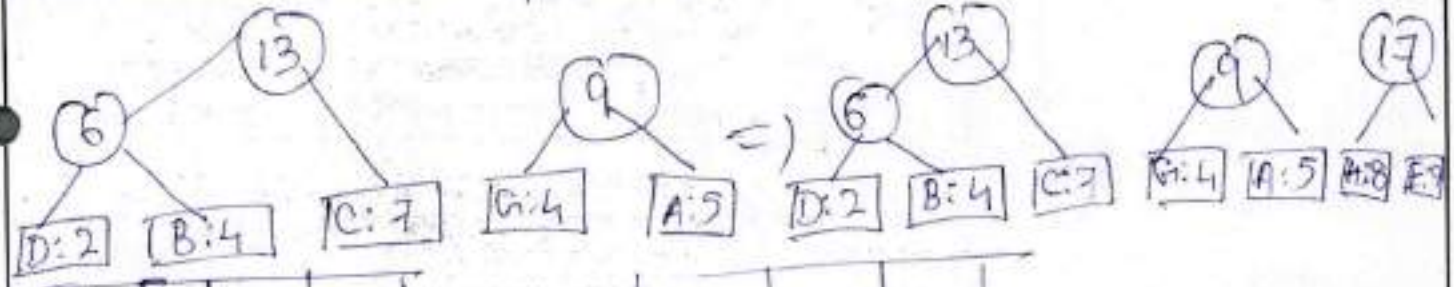
G	A		C	H	E	F
4	5	6	7	8	9	11



	C	H	E		F
6	7	8	9	9	11

=>

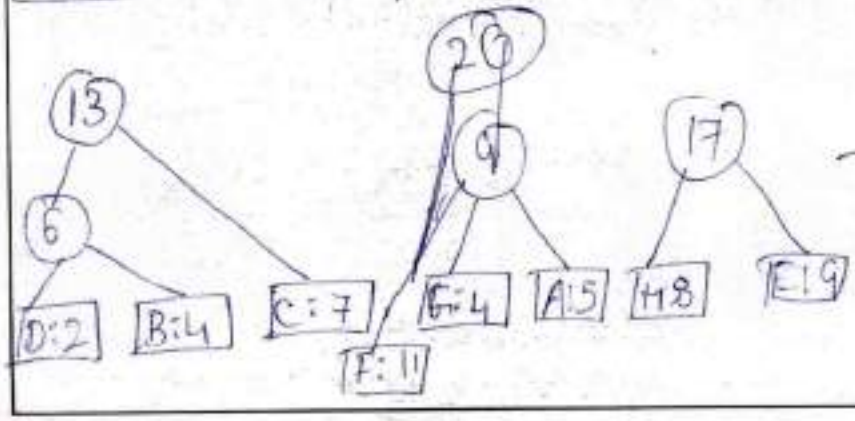
H	E		F	
8	9	9	11	13



	F		
9	11	13	17

=>

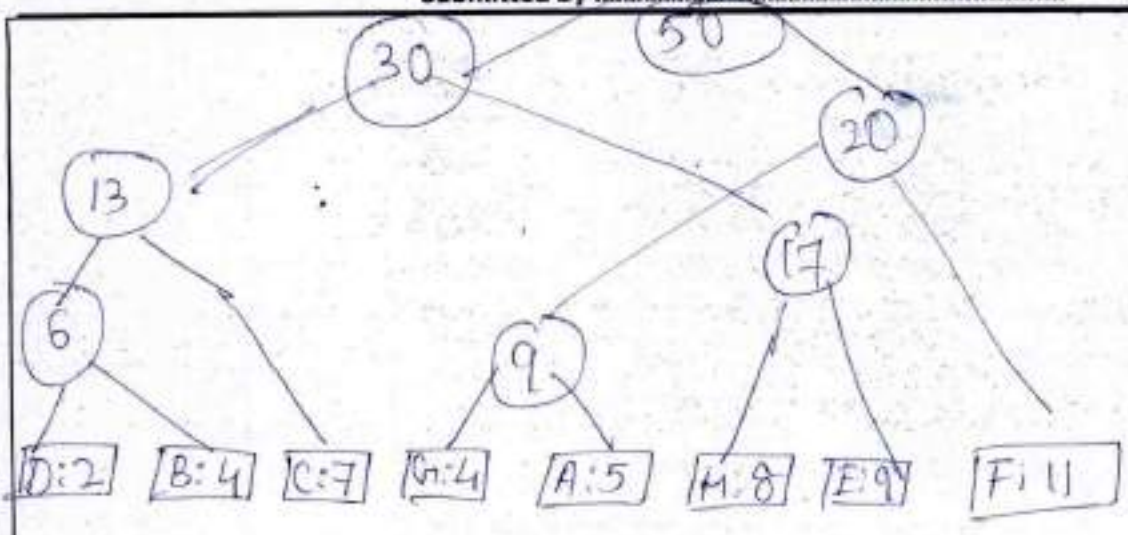
13	17	20



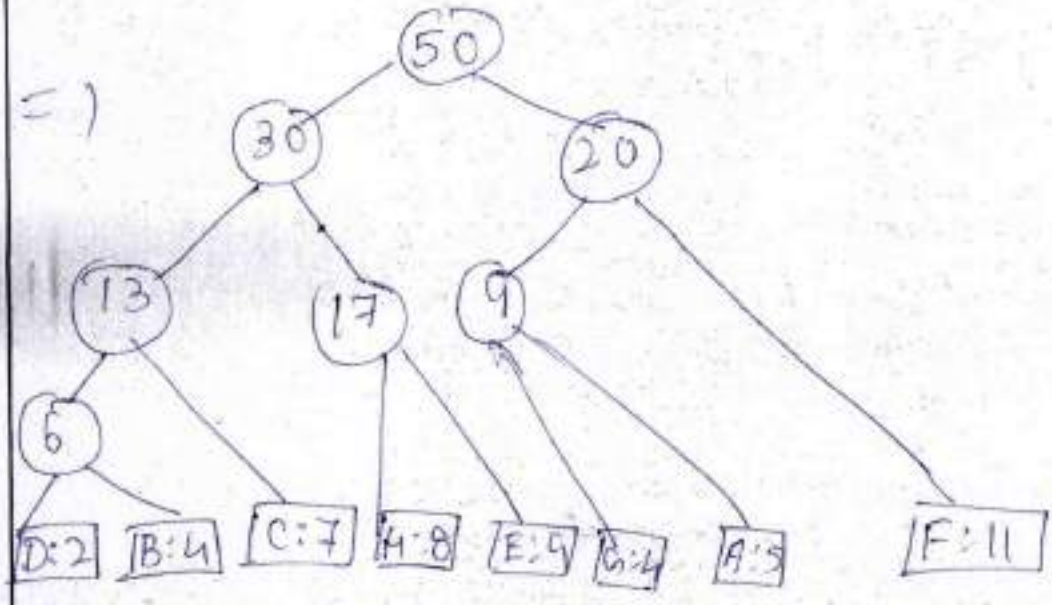
Question Paper Solution

Branch : IT Semester: 6th Subject: DAA Mid Term: I/II/Extra/Imp.

Submitted By : NEHA MATHR



=)



Question Paper Solution

Branch : I.T. Semester: 6 Subject: DAA Mid Term: I/II/Extra/Imp.

Submitted By : NEHA MATHUR

Q3 =

Items	I_1	I_2	I_3	I_4
Weights	5	4	6	3
Price	40	30	20	50
P/W	8	7.5	3.33	16.66

Step1: Calculate Price per weight (P/w) ratio

Step2 Arrange items with decreasing order of Ratio

Items	Weight	Price	P/w
I_4	3	50	16.66
I_1	5	40	8
I_2	4	30	7.5
I_3	6	20	3.33

Greedy-fractional-krapsack (w, v, W)

```
for i := 1 to n
    do x[i] = 0, here x[i] gives fraction of
    weight = 0                          weight
while weight < W
do i = best remaining item
    if weight + w[i] = W
        then x[i] = 1
            weight = weight + w[i]
    else
        x[i] = (w - weight) / w[i]
        weight = W
return x
```

Following above algorithm

Question Paper Solution

Branch : I.T Semester: 6 Subject: DAA Mid Term: I/II/Extra/Imp.
Submitted By : NEHA MATHUR

Items	Pi/wi	wi	Pi	x[i]	W (Capacity of knapsack) 10
I_4	16.66	3	50	1	10-3 = 7
I_1	8	5	40	1	7-5 = 2
I_2	7.5	4	30	2/4	0
I_3	3.33	6	20	0	0

Profit = $(1 \times 50) + (1 \times 40) + (\frac{2}{4} \times 30) + (0 \times 20)$
= 50 + 40 + 15 + 0
= 105

OR

Q3.

Items	I_1	I_2	I_3	I_4	I_5
weight	5	10	20	30	40
values	30	20	100	90	160
V/W	6	2	5	3	4

Step1: Calculate values/weight ratio

Step2: Arrange items with decreasing order of their ratio.

Items	Vi/wi	wi	Pi	x[i]	W (Capacity of knapsack) 60
I_1	6	5	30	1	60-5 = 55
I_3	5	20	100	1	55-20 = 35
I_5	4	40	160	35/40	35-35 = 0
I_4	3	30	90	0	0
I_2	2	10	20	0	0

Profit = $(1 \times 30) + (1 \times 100) + (\frac{35}{40} \times 100) + (0 \times 90) + (0 \times 20)$
= 30 + 100 + 87.5 + 0 + 0 = 217.5

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017

Question Paper Solution

Branch : IT Semester :VI Subject: JAVA Mid Term: I.

Submitted By :Shubhra Saxena

Q1: Explain different feature of Java that are supported by OOP's principle with examples. (4)

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Dynamic
9. Interpreted
10. High Performance
11. Multithreaded
12. Distributed

Q.1. Explain the working of Java virtual machine. What is the significance of Java byte code in the Java programming language. (4)

Working of JVM

JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed.

1) Classloader

Classloader is a subsystem of JVM that is used to load class files.

2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

It is the runtime data area in which objects are allocated.

4) Stack

Java Stack stores frames.It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register. It contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

7) Execution Engine

It contains:

1) A virtual processor

2) Interpreter: Read bytecode stream then execute the instructions.

3) Just-In-Time(JIT) compiler: It is used to improve the performance.JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time needed for compilation.Here the term ?compiler? refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

Bytecodes are the machine language of the Java virtual machine. When a JVM loads a class file, it gets one stream of bytecodes for each method in the class. The bytecodes streams are stored in the method area of the JVM. The bytecodes for a method are executed when that method is invoked during the course of running the program. They can be executed by intepretation, just-in-time compiling, or any other technique that was chosen by the designer of a particular JVM.

In Java the compiler Javac converts your .java file to bytecode under .class file and then the Interpreter built under JVM converts this byteCode to machine code required on that Machine. Bytecode is a portable form of executable code. Bytecode is platform independent .

Constructors are used to initialize the object's state. Like methods, a constructor also contains **collection of statements(i.e. instructions)** that are executed at time of Object creation.

Q.2. What do you mean by the constructor in JAVA. Also explain different manner of constructor chaining in Java.(4)

Constructors are used to initialize the object's state. Like methods, a constructor also contains **collection of statements(i.e. instructions)** that are executed at time of Object creation.

There are two type of constructor in Java:

1. **No-argument constructor:** A constructor that has no parameter is known as default constructor. If we don't define a constructor in a class, then compiler creates **default constructor(with no arguments)** for the class. And if we write a constructor with arguments or no-argument then compiler does not create default constructor. Default constructor provides the default values to the object like 0, null etc. depending on the type.
2.

```
// Java Program to illustrate calling a
// no-argument constructor
import java.io.*;

class Geek
{
    int num;
    String name;

    // this would be invoked while object
    // of that class created.
    Geek()
    {
        System.out.println("Constructor called");
    }
}
```

```
class GFG
{
    public static void main (String[] args)
    {
        // this would invoke default constructor.
        Geek geek1 = new Geek();

        // Default constructor provides the default
        // values to the object like 0, null
        System.out.println(geek1.name);
        System.out.println(geek1.num);
    }
}
```

3. Run on IDE
4. Output :

```
5. Constructor called
6. null
7. 0
```

8. **Parameterized Constructor:** A constructor that has parameters is known as parameterized constructor. If we want to initialize fields of the class with your own values, then use parameterized constructor.

```
// Java Program to illustrate calling of
// parameterized constructor.
import java.io.*;

class Geek
{
    // data members of the class.
    String name;
    int id;

    // contructor would initialized data members
    // with the values of passed arguments while
    // object of that class created.
    Geek(String name, int id)
    {
        this.name = name;
        this.id = id;
    }
}

class GFG
{
    public static void main (String[] args)
    {
        // this would invoke parameterized constructor.
        Geek geek1 = new Geek("adam", 1);
        System.out.println("GeekName :" + geek1.name +
                           " and GeekId :" + geek1.id);
    }
}
```

Constructor Chaining :

- **Within same class**: It can be done using **this()** keyword for constructors in same class
- **From base class:** by using **super()** keyword to call constructor from the base class.

Constructor chaining occurs through **inheritance**. A sub class constructor's task is to call super class's constructor first. This ensures that creation of sub class's object starts with the initialization of the data members of the super class. There could be any numbers of classes in inheritance chain.

```
// Java program to illustrate Constructor Chaining
// within same class Using this() keyword
class Temp
{
   // default constructor 1
   // default constructor will call another constructor
   // using this keyword from same class
   Temp()
   {
      // calls constructor 2
      this(5);
      System.out.println("The Default constructor");
   }

   // parameterized constructor 2
   Temp(int x)
   {
      // calls constructor 3
      this(5, 15);
      System.out.println(x);
   }

   // parameterized constructor 3
   Temp(int x, int y)
   {
      System.out.println(x * y);
   }

   public static void main(String args[])
   {
      // invokes default constructor first
      new Temp();
   }
}
```

Run on IDE

Output:

```
The Default constructor
5
75
```

Q.2. How objects are used as parameter in Java. Explain using an appropriate example.(4)

```
class Rectangle {
```

```
    int length;
    int width;

    Rectangle(int l, int b) {
        length = l;
        width = b;
    }

    void area(Rectangle r1) {
        int areaOfRectangle = r1.length * r1.width;
        System.out.println("Area of Rectangle : "
                + areaOfRectangle);
    }
}
class RectangleDemo {
    public static void main(String args[]) {
        Rectangle r1 = new Rectangle(10, 20);
        r1.area(r1);
    }
}
```

Output of the program :

```
Area of Rectangle : 200
```

1. We can pass Object of any class as parameter to a method in java.
2. We can access the instance variables of the object passed inside the called method

Q.3. Describe the overloading and overriding method in JAVA.(4)

Overloading allows different methods to have same name, but different signatures where signature can differ by number of input parameters or type of input parameters or both. Overloading is related to compile time (or static) polymorphism.

```
// Java program to demonstrate working of method
// overloading in Java.

public class Sum {

    // Overloaded sum(). This sum takes two int parameters
    public int sum(int x, int y) {
        return (x + y);
    }

    // Overloaded sum(). This sum takes three int parameters
```

```
    public int sum(int x, int y, int z) {
        return (x + y + z);
    }

    // Overloaded sum(). This sum takes two double parameters
    public double sum(double x, double y) {
        return (x + y);
    }

    // Driver code
    public static void main(String args[]) {
        Sum s = new Sum();
        System.out.println(s.sum(10, 20));
        System.out.println(s.sum(10, 20, 30));
        System.out.println(s.sum(10.5, 20.5));
    }
}
```

Method Overriding:

If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in java**.

In other words, If subclass provides the specific implementation of the method that has been provided by one of its parent class, it is known as method overriding.

```
class Bank{
int getRateOfInterest(){return 0;}
}

class SBI extends Bank{
int getRateOfInterest(){return 8;}
}

class ICICI extends Bank{
int getRateOfInterest(){return 7;}
}
class AXIS extends Bank{
int getRateOfInterest(){return 9;}
}

class Test2{
public static void main(String args[]){
SBI s=new SBI();
ICICI i=new ICICI();
AXIS a=new AXIS();
System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());
System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());
System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());
}
}
```

```
class Bank{
int getRateOfInterest(){return 0;}
}
```

```
class SBI extends Bank{
int getRateOfInterest(){return 8;}
}

class ICICI extends Bank{
int getRateOfInterest(){return 7;}
}
class AXIS extends Bank{
int getRateOfInterest(){return 9;}
}

class Test2{
public static void main(String args[]){
SBI s=new SBI();
ICICI i=new ICICI();
AXIS a=new AXIS();
System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());
System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());
System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());
}
}
```

Q.3. How can you implement one dimensional and multidimensional array in JAVA . Also give an example.(4)

There are two types of array.

- Single Dimensional Array
- Multidimensional Array

Example of single dimensional java array

Let's see the simple example of java array, where we are going to declare, instantiate, initialize and traverse an array.

```
class Testarray{
public static void main(String args[]){

int a[]=new int[5];//declaration and instantiation
a[0]=10;//initialization
a[1]=20;
a[2]=70;
a[3]=40;
a[4]=50;

//printing array
for(int i=0;i<a.length;i++)//length is the property of array
System.out.println(a[i]);

}}
```

Multidimensional Array

```
class Testarray5{
```

Question Paper Solution

Branch : IT Semester :VI Subject: JAVA Mid Term: I.

Submitted By :Shubhra Saxena

```
public static void main(String args[]){
//creating two matrices
int a[][]={{1,3,4},{3,4,5}};
int b[][]={{1,3,4},{3,4,5}};

//creating another matrix to store the sum of two matrices
int c[][]=new int[2][3];

//adding and printing addition of 2 matrices
for(int i=0;i<2;i++){
for(int j=0;j<3;j++){
c[i][j]=a[i][j]+b[i][j];
System.out.print(c[i][j]+" ");
}
System.out.println();//new line
}

}}
```

Q.4. Write short note of any two (3)

(i) Bitwise and Shift Operator

Bitwise Operators : These operators are used to perform manipulation of individual bits of a number. They can be used with any of the integer types. They are used when performing update and query operations of Binary indexed tree.

- **& , Bitwise AND operator:** returns bit by bit AND of input values.
- **| , Bitwise OR operator:** returns bit by bit OR of input values.
- **^ , Bitwise XOR operator:** returns bit by bit XOR of input values.
- **~ , Bitwise Complement Operator:** This is a unary operator which returns the one's compliment representation of the input

Java program to illustrate

```
// bitwise operators
public class operators
{
  public static void main(String[] args)
  {

    int a = 0x0005;
    int b = 0x0007;

    // bitwise and
    // 0101 & 0111=0101
    System.out.println("a&b = " + (a & b));

    // bitwise and
    // 0101 | 0111=0111
    System.out.println("a|b = " + (a | b));

    // bitwise xor
    // 0101 ^ 0111=0010
    System.out.println("a^b = " + (a ^ b));

    // bitwise and
    // ~0101=1010
    System.out.println("~a = " + ~a);
```

```
    // can also be combined with assignment erator to provide
shorthand
    // assignment
    // a=a&b
    a &= b;
    System.out.println("a= " + a);
  }
}
```

- Run on IDE
- **Output :**

```
a&b = 5
a|b = 7
a^b = 2
~a = -6
a= 5
```

Shift Operators : These operators are used to shift the bits of a number left or right thereby multiplying or dividing the number by two respectively. They can be used when we have to multiply or divide a number by two. General format-

```
number shift_op number_of_places_to_shift;
```

- **<< , Left shift operator:** shifts the bits of the number to the left and fills 0 on voids left as a result. Similar effect as of multiplying the number with some power of two.
- **>> , Signed Right shift operator:** shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit depends on the sign of initial number. Similar effect as of dividing the number with some power of two.
- **>>> , Unsigned Right shift operator:** shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit is set to 0.

```
// Java program to illustrate
// shift operators
public class operators
{
  public static void main(String[] args)
  {

    int a = 0x0005;
    int b = -10;

    // left shift operator
    // 0000 0101<<2 =0001 0100(20)
    // similar to 5*(2^2)
    System.out.println("a<<2 = " + (a << 2));

    // right shift operator
    // 0000 0101 >> 2 =0000 0001(1)
    // similar to 5/(2^2)
    System.out.println("a>>2 = " + (a >> 2));
```

```
    // unsigned right shift operator
    System.out.println("b>>>2 = "+ (b >>> 2));

  }
}
```

Run on IDE

Output :

```
a<>2 = 1
b>>>2 = 1073741821
```

(II) Super

The **super** keyword in java is a reference variable which is used to refer immediate parent class object.

Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

Usage of java super Keyword

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

1) super is used to refer immediate parent class instance variable.

We can use super keyword to access the data member or field of parent class. It is used if parent class and child class have same fields.

```
class Animal{
String color="white";
}
class Dog extends Animal{
String color="black";
void printColor(){
System.out.println(color);//prints color of Dog class
System.out.println(super.color);//prints color of Animal class
}
}
class TestSuper1{
public static void main(String args[]){
Dog d=new Dog();
d.printColor();
}}
```

(iii) Static vs Instance variable

Instance variable Vs Static variable

- Each object will have its **own copy** of instance variable whereas We can only have **one copy** of a static variable per class irrespective of how many objects we create.
- Changes made in an instance variable using one object will **not be reflected** in other objects as each object has its own copy of instance variable. In case of static, changes **will be reflected** in other objects as static variables are common to all object of a class.
- We can access instance variables **through object references** and Static Variables can be accessed **directly using class name.**

Question Paper Solution

Branch : IT Semester :VI Subject: JAVA Mid Term: I.

Submitted By :Shubhra Saxena

(iv) Using final with Inheritance

---

final is a keyword in java used for restricting some functionalities. We can declare variables, methods and classes with final keyword.

We can declare final method in any subclass for which we want that if any other class extends this subclass, then it must follow same implementation of the method as in the that subclass.

```
Java program to illustrate
// use of final with inheritance

// base class
abstract class Shape
{
    private double width;

    private double height;

    // Shape class parameterized constructor
    public Shape(double width, double height)
    {
        this.width = width;
        this.height = height;
    }

    // getWidth method is declared as final
    // so any class extending
    // Shape cann't override it
    public final double getWidth()
    {
        return width;
    }

    // getHeight method is declared as final
    // so any class extending Shape
    // can not override it
    public final double getHeight()
    {
        return height;
    }


    // method getArea() declared abstract because
    // it upon its subclasses to provide
    // complete implementation
    abstract double getArea();
}

// derived class one
class Rectangle extends Shape
{
    // Rectangle class parameterized constructor
    public Rectangle(double width, double height)
    {
        // calling Shape class constructor
        super(width, height);
```

```
    }

    // getArea method is overridden and declared
    // as final so any class extending
    // Rectangle cann't override it
    @Override
    final double getArea()
    {
        return this.getHeight() * this.getWidth();
    }

}

//derived class two
class Square extends Shape
{
    // Rectangle class parameterized constructor
    public Square(double side)
    {
        // calling Shape class constructor
        super(side, side);
    }

    // getArea method is overridden and declared as
    // final so any class extending
    // Square cann't override it
    @Override
    final double getArea()
    {
        return this.getHeight() * this.getWidth();
    }

}

// Driver class
public class Test
{
    public static void main(String[] args)
    {
        // creating Rectangle object
        Shape s1 = new Rectangle(10, 20);

        // creating Square object
        Shape s2 = new Square(10);

        // getting width and height of s1
        System.out.println("width of s1 : "+ s1.getWidth());
        System.out.println("height of s1 : "+ s1.getHeight());

        // getting width and height of s2
        System.out.println("width of s2 : "+ s2.getWidth());
        System.out.println("height of s2 : "+ s2.getHeight());

        //getting area of s1
        System.out.println("area of s1 : "+ s1.getArea());

        //getting area of s2
        System.out.println("area of s2 : "+ s2.getArea());

    }
}
```

Question Paper Solution

Branch : IT Semester :VI Subject: JAVA Mid Term: I.

Submitted By :Shubhra Saxena

```
width of s1 : 10.0
height of s1 : 20.0
width of s2 : 10.0
height of s2 : 10.0
area of s1 : 200.0
area of s2 : 100.0
```

Example: Using Final to prevent Inheritance:

```
class DataV1 {

   final void dataValues()
   {
      System.out.println("DataV1 Values");
   }
}

class DataV2 extends DataV1{

   void dataValues()
   {
      System.out.println("DataV2 Values");
   }
}

public class Javaapp {

   public static void main(String[] args){

      DataV2 obj = new DataV2();
      obj.dataValues();
   }
}
```

Question Paper Solution

Branch : IT Semester: VI Subject: ITC Mid Term: I/II/Extra/Imp.

Submitted By : SUMAN SHARMA

Soln (i)

$f_s = 2k \times 10 = 20\ kHz$

$$H(X) = \sum_{i=1}^{m} P(x_i)\, I(x_i)$$

$$= \frac{1}{4}\log_2 4 + \frac{1}{5}\log_2 5 + \frac{1}{5}\log_2 5 + \frac{1}{10}\log_2 10 + \frac{1}{10}\log_2 10 + \frac{1}{20}\log_2 20 + \frac{1}{20}\log_2 20 + \frac{1}{20}\log_2 20$$

or

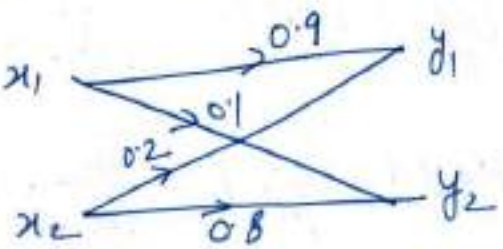
$$H(X) = \frac{1}{4}\log_2 4 + \frac{2}{5}\log_2 5 + \frac{1}{20}\log_2 20$$

$$= 2.84 \text{ bits/message}$$

$r = 20000$ message/sec.

$R = r\,H(X) = 20{,}000 \times 2.84 = 56800$ bit/sec.

Soln (ii) or part



$$P(Y) = P[X] \cdot P\left[\frac{Y}{X}\right]$$

Question Paper Solution

Branch : IT Semester: II Subject: ITC Mid Term: I/II/Extra/Imp.

Submitted By : Suman Sharma

$$P[Y] = [0.5 \quad 0.5] \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$$

$$P[Y] = [0.55 \quad 0.45] = [P[y_1] \quad P[y_2]]$$

$$\Rightarrow P[y_1] = 0.55, \quad P[y_2] = 0.45$$

(iii) $P[X,Y] = [P[X]][P[Y/X]]$

$$= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix} = \begin{bmatrix} 0.45 & 0.05 \\ 0.1 & 0.4 \end{bmatrix}$$

$\Rightarrow P(x_1, y_2) = \underline{0.05}$ & $P(x_2, y_1) = \underline{0.1}$

Soln (2) $C = B \log_2 (1 + \frac{S}{N})$ b/s

$$C = B \log_2 (1 + \frac{S}{\eta B}) \quad \because N = \eta B$$

Let $S/\eta B = \lambda$ then,

$$C = \frac{S}{\eta \lambda} \log_2 (1+\lambda) = \frac{1}{\ln 2} \frac{S}{\eta} \frac{\ln(1+\lambda)}{\lambda}$$

$$C_\infty = \lim_{B \to \infty} B \log_2 (1 + \frac{S}{\eta B}) = \frac{1}{\ln 2} \cdot \frac{S}{\eta}$$

$$= \lim_{\lambda \to 0} \frac{\ln(1+\lambda)}{\lambda}$$

Sin $\lim_{\lambda \to 0} [\ln(1+\lambda)/\lambda] = 1$

$$C_\infty = \frac{1}{\ln 2} \cdot \frac{S}{\eta} = 1.44 \frac{S}{\eta} \text{ b/s}$$

Question Paper Solution

Branch : IT Semester: VI Subject: ITC Mid Term: I/II/Extra/Imp.

Submitted By : Suman Sharma

Soln (2) (a) part)

Source coding theorem:

$$L \geq H(X)$$

&

$$\eta = \frac{H(X)}{L} \quad ; \text{ if } L_{min} = L$$

Kraft Inequality

$$K = \sum_{i=1}^{m} 2^{-n_i} \leq 1.$$

where n_i = codeword length

1) Codeword length: No. of bits in the codeword.

2) Average codeword length:

$$L = \sum_{i=1}^{m} P(x_i)\, n_i$$

3) Code Efficiency: $\eta = \frac{L_{min}}{L}$

4) Code Redundancy $\Rightarrow \gamma = 1 - \eta$

Soln (2)

x_i	$P(x_i)$	step 1	step 2	step 3	Code
x_1	0.2	0	0		00
x_2	0.2	0	1		01
x_3	0.2	1	0		10
x_4	0.2	1	1	0	110
x_5	0.2	1	1	1	111

Question Paper Solution

Branch : IT Semester: 7th Subject: ITC Mid Term: I/II/Extra/Imp.

Submitted By : Suman Sharma

$$H(X) = -\sum_{i=1}^{5} P(x_i) \log_2 P(x_i)$$

$$= 5(-0.2 \log_2 0.2) = 2.32$$

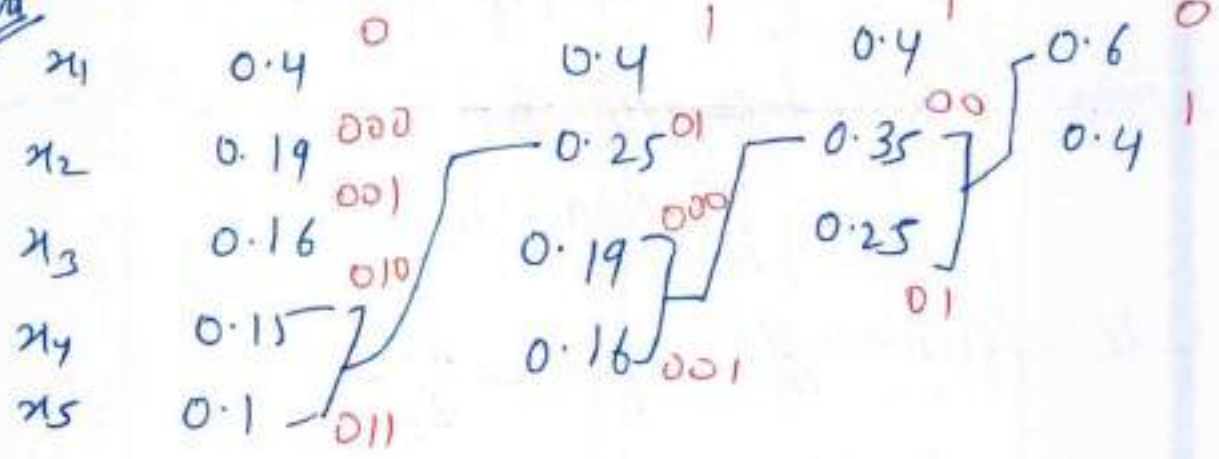
$$L = \sum_{i=1}^{5} P(x_i) n_i = 0.2(2+2+2+3+3)$$

$$= 2.4$$

$$\eta = \frac{H(X)}{L} = \frac{2.32}{2.4} = 0.967 \Rightarrow 96.7\%$$

Sol 7(a) or Part

x_i	$P(x_i)$	code
x_1	0.4	0
x_2	0.19	000
x_3	0.16	001
x_4	0.15	010
x_5	0.1	011



$$H(X) = -\sum_{i=1}^{M} P(x_i) \log_2 P(x_i) = 2.15$$

$$\eta = \sum_{i=1}^{M} P(x_i) n_i =$$

$$= 0.4(1) + 3[0.19 + 0.16 + 0.15 + 0.1]$$

$$= 2.2$$

Question Paper Solution

Branch : IT Semester: 4 Subject: ITC Mid Term: I/II/Extra/Imp.

Submitted By : Suman Sharma

$$\eta = \frac{H(X)}{L} = \frac{2.15}{2.2} = .977 = 97.7\%$$

Soln

0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1
1 2 3 4 5 6 7 8 9 10 11 12 13

0 → A

1 → B

A B 2B 1B 3A 1A 4A 2A 8A 4B 7B 7A 2

Code 0 | 00101 | 00011 | 00110 | 00010 | 10000 | 00100 | 10000 | 01001 | 01111 | 01110, 00010

Soln (or part)

Error detection methods:

1) parity check
2) checksum detection
3) cyclic redundancy check (CRC)

1) Parity check:-

(a) even parity

P	dataword
0	0110

P	dataword
1	0111

(b) odd parity

P	dataword
0	0010

P	Data word
1	0101

Question Paper Solution

Branch : IT Semester: 8 Subject: ITC Mid Term: I/II/Extra/Imp.

Submitted By : Suman Sharma

check sum :→

0100 1111 | 0101 1011 | 11001 | 00

K=1 K=2

Tx Side

```
  0100 1111
+ 0101 1011
-----------
  1010 0101 0
+ 1100 1100
-----------
  0111 0110
        1
-----------
  0111 0111  ← Sum
  1000 1000  ← checksum
```

Rx side

```
  0111 0111
+ 1000 1000
-----------
  1111 1111
```

invert

0000 0000 → correct code

3) CRC :→

Tx side

```
      1
1000)10110000  → K
     1000
     ----
      0011 0
      1000
      ----
       1011 0
       1000
       ----
        0011 000
           1000
           ----
            0000
```

Rx side

→ add Remainder to K

→ divide by same polynomial

if r = 000 ⇒ correct

Swami Keshvanand Institute of Technology, Management &Gramothan,
Ramnagaria, Jagatpura, Jaipur-302017

Question Paper Solution

Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....
Mid Term: I
Submitted By :.....Sanju Choudhary.....

Q1. Explain Layered architecture, Policies and Mechanism of operating system. Differentiate between Monolithic and Microkernel Operating-System structure.

Ans. The policies what is to be done while the mechanism specifies how it is to be done? For instance, the timer construct for ensuring CPU protection is mechanism. On the other hand, the decision of how long the timer is set for a particular user is a policy decision. The separation of mechanism and policy is important to provide flexibility to a system. If the interface between mechanism and policy is well defined, the change of policy may affect only a few parameters.

Operating-System Structure

This structures the operating system by removing all nonessential portions of the kernel and implementing them as system and user level programs. Generally they provide minimal process and memory management, and a communications facility. Communication between components of the OS is provided by message passing.

1. **Monolithic**

- Functionality of the OS is invoked with simple function calls within the kernel, which is one large program.
- Device drivers are loaded into the running kernel and become part of the kernel.

2. **Microkernel**

This structures the operating system by removing all nonessential portions of the kernel and implementing them as system and user level programs.

- Generally they provide minimal process and memory management, and a communications facility.
- Communication between components of the OS is provided by message passing.

The *benefits* of the microkernel are as follows:

- Extending the operating system becomes much easier.

Question Paper Solution

Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....

Mid Term: I

Submitted By :.....Sanju Choudhary.....

- Any changes to the kernel tend to be fewer, since the kernel is smaller.
- The microkernel also provides more security and reliability.

Main *disadvantage* is poor performance due to increased system overhead from message passing.

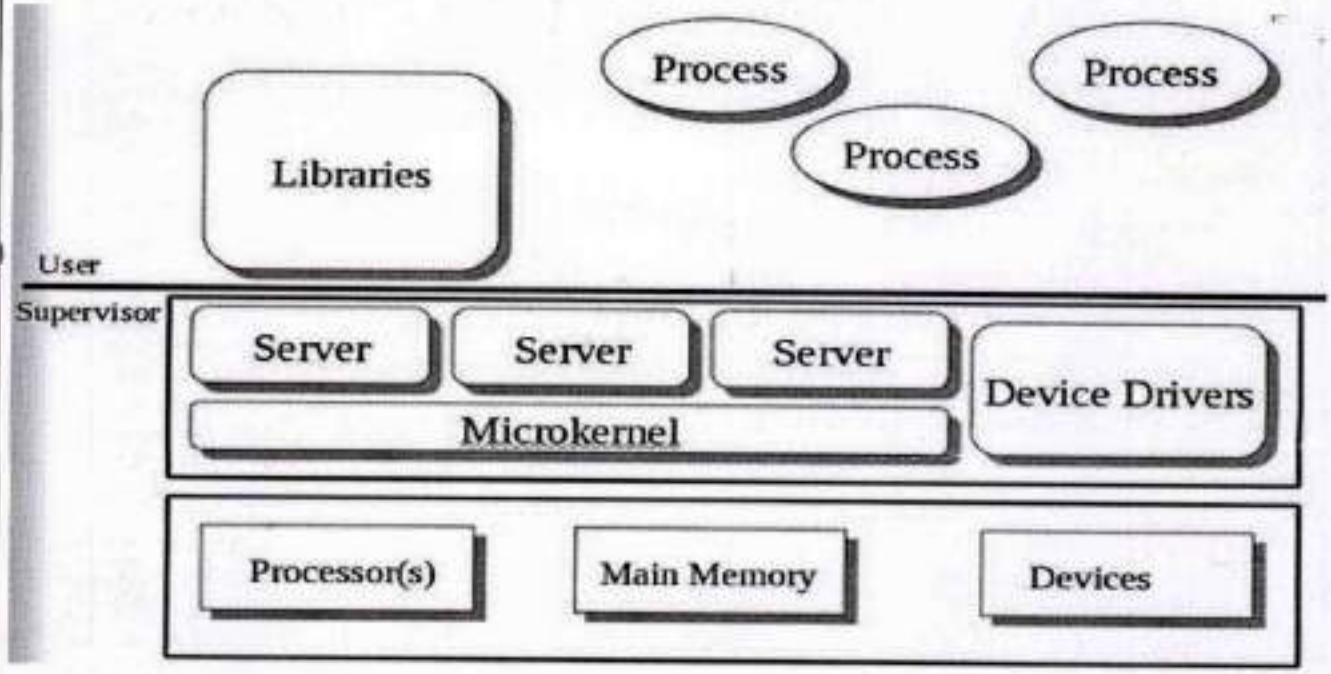


Fig. Microkernel architecture.

Note

- Andrew Tanenbaum's Minix is an example of a microkernel system. Minix was developed primarily to facilitate teaching graduate level operating system classes. Tanenbaum has authored several text books and is with VA University in Amsterdam.
- Another well known microkernel system is **Mach**, which was developed at Carnegie Mellon University in the mid-1980's. Mach was used as the low-level part of Apple OS X.

OR

Q.1 Explain thread libraries and describe different threading issues related to threading applications in operating system to support multithreading programming? What is Remote Method Invocation? How does it work and also write the advantages of RMI?

Ans. ***Thread Libraries***

Thread libraries provides programmers with API for creating and managing of threads.

Thread libraries may be implemented either in user space or in kernel space. The user space involves API functions implemented solely within user space, with no kernel support. The kernel space involves system calls,

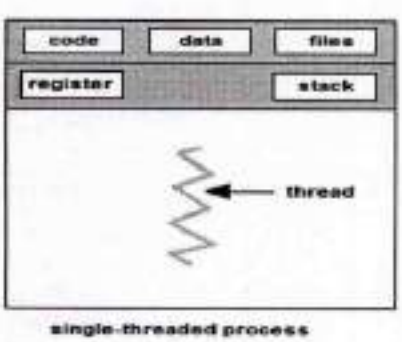
and requires a kernel with thread library support.

There are three types of thread:

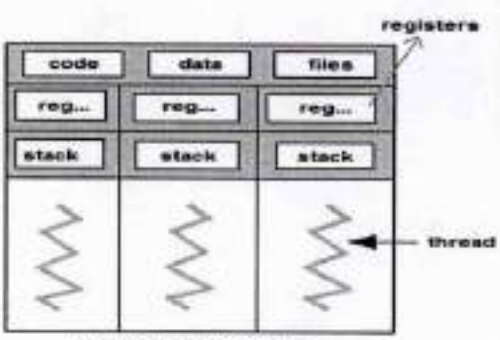
- POSIX Pitheads may be provided as either a user or kernel library, as an extension to the POSIX standard.
- Win32 threads are provided as a kernel-level library on Windows systems.
- Java threads - Since Java generally runs on a Java Virtual Machine, the implementation of threads is based upon whatever OS and hardware the JVM is running on, i.e. either Pitheads or Win32 threads depending on the system

Thread is an execution unit which consists of its own program counter, a stack, and a set of registers. Threads are also known as Lightweight processes. Threads are popular way to improve application through parallelism. The CPU switches rapidly back and forth among the threads giving illusion that the threads are running in parallel.

As each thread has its own independent resource for process execution, multpile processes can be executed parallely by increasing number of threads.



single-threaded process



multithreaded process

Types of Thread

There are two types of threads:

- User Threads
- Kernel Threads

User threads, are above the kernel and without kernel support. These are the threads that application programmers use in their programs.

Kernel threads are supported within the kernel of the OS itself. All modern OSs support kernel level threads, allowing the kernel to perform multiple simultaneous tasks and/or to service multiple kernel system calls simultaneously

Benefits of Multithreading

Question Paper Solution

Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....

Mid Term: I

Submitted By :.....Sanju Choudhary.....

1. Responsiveness
2. Resource sharing, hence allowing better utilization of resources.
3. Economy. Creating and managing threads becomes easier.
4. Scalability. One thread runs on one CPU. In Multithreaded processes, threads can be distributed over a series of processors to scale.
5. Context Switching is smooth. Context switching refers to the procedure followed by CPU to change from one task to another.

Multithreading Issues

1. **Thread Cancellation**.

   Thread cancellation means terminating a thread before it has finished working. There can be two approaches for this, one is **Asynchronous cancellation**, which terminates the target thread immediately. The other is **Deferred cancellation** allows the target thread to periodically check if it should be cancelled.

2. **Signal Handling**.

   Signals are used in UNIX systems to notify a process that a particular event has occurred. Now in when a Multithreaded process receives a signal, to which thread it must be delivered? It can be delivered to all, or a single thread.

3. **fork() System Call**.

   fork() is a system call executed in the kernel through which a process creates a copy of itself. Now the problem in Multithreaded process is, if one thread forks, will the entire process be copied or not?

4. **Security Issues** because of extensive sharing of resources between multiple threads.

There are many other issues that you might face in a multithreaded process, but there are appropriate solutions available for them. Pointing out some issues here was just to study both sides of the coin.

Q2. Explain different types of file allocation and file access methods in os with an example.

The main idea behind allocation is effective utilization of file space and fast access of the files. There are three types of allocation:

1. contiguous allocation
2. linked allocation
3. indexed allocation

In addition to storing the actual file data on the disk drive, the file system also stores metadata about the files: the name of each file, when it was last edited, exactly where it is on the disk, and what parts of the disk are "free". Free areas are not currently in use by the file data or the metadata, and so available for storing new files. (The places where this metadata is stored are often called "inodes", "chunks", "file allocation tables", etc.)

Question Paper Solution

Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....

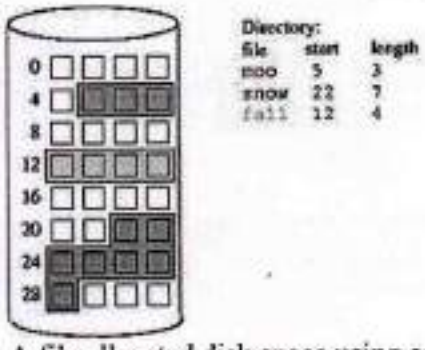
Mid Term: I

Submitted By :.....Sanju Choudhary.....

To keep track of the free space, the file system maintains a free-space list which tracks all the disk blocks which are free. To create a file, the required space is reserved for the file and the corresponding space is removed from the free list linked to each other.

Contiguous allocation

With contiguous allocation, each file occupies contiguous blocks on the disk. The location of a file is defined by the disk address of the first block and its length.



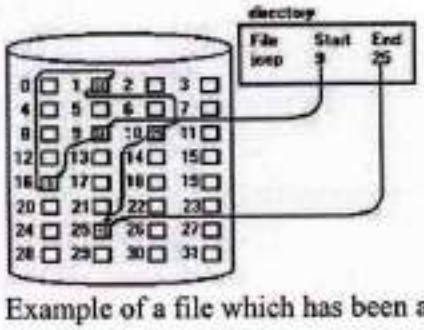
A file allocated disk space using contiguous allocation method

Both sequential access and direct/Random access are supported by the contiguous allocation. We can say that it supports random access as using Disk Block Address we can jump directly on the required location.

The disadvantage of contiguous allocation is that it is often difficult to increase the size of a file as the next contiguous block may not be free. Moreover, one is often not sure of the space required while creating a new file. The various methods adopted to find space for a new file suffer from external fragmentation. Internal fragmentation may exist in the last disk block of a file.

Linked allocation

In linked allocation, each file is a linked list of disk blocks. The directory contains a pointer to the first and (optionally the last) block of the file. For example, a file of 5 blocks which starts at block 4, might continue at block 7, then block 16, block 10, and finally block 27. Each block contains a pointer to the next block and the last block contains a NIL pointer. The value -1 may be used for NIL to differentiate it from block 0.



Example of a file which has been allocated disk space using linked allocation method.

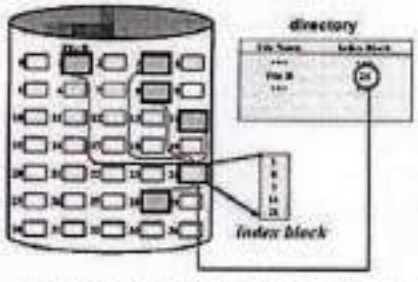
With linked allocation, each directory entry has a pointer to the first disk block of the file. This pointer is initialized to nil (the end-of-list pointer value) to signify an empty file. A write to a file removes the first free

block and writes to that block. This new block is then linked to the end of the file. To read a file, the pointers are just followed from block to block.

There is no external fragmentation with linked allocation. Any free block can be used to satisfy a request. Notice also that there is no need to declare the size of a file when that file is created. A file can continue to grow as long as there are free blocks. Linked allocation, does have disadvantages, however. The major problem is that it is inefficient to support direct-access; it is effective only for sequential-access files. To find the block of a file, it must start at the beginning of that file and follow the pointers until the block is reached. Note that each access to a pointer requires a disk read.

Indexed allocation

Linked allocation does not support random access of files, since each block can only be found from the previous. Indexed allocation solves this problem by bringing all the pointers together into an index block. One disk block is just used to store DBAs (disk block addresses) of a file.



a file allocated disk space using Indexed allocation Method

Every file is associated with its own index node. If a file is very large then one disk block may not be sufficient to hold all associated DBAs of that file. If a file is very small then some disk block space is wasted as DBAs are less and a single disk block could still hold more DBAs.

This method solves the problem of fragmentation as the blocks can be stored in any location.

OR

Q2. What is device driver & device controller? Explain the different types of buses and interfaces in detail.

A **device driver** is a program that controls a particular type of **device** that is attached to your **computer**. There are **device drivers** for printers, displays, CD-ROM readers, diskette drives, and so on. When you buy an operating system, many **device drivers** are built into the product. A device driver is a small piece of software that tells the operating system and other software how to communicate with a piece of hardware.

For example, printer drivers tell the operating system, and by extension whatever program you have the thing you want to print open in, *exactly* how to print information on the page

Sound card drivers are necessary so your operating system knows exactly how to translate the 1's and 0's that

comprise that MP3 file into audio signals that the sound card can output to your headphones or speakers.

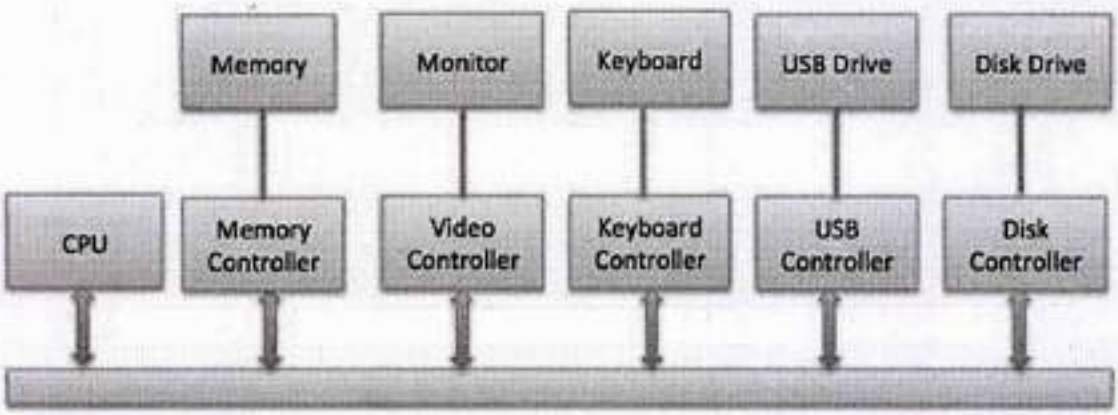
Device Controllers

Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices.

The Device Controller works like an interface between a device and a device driver. I/O units (Keyboard, mouse, printer, etc.) typically consist of a mechanical component and an electronic component where electronic component is called the device controller.

There is always a device controller and a device driver for each device to communicate with the Operating Systems. A device controller may be able to handle multiple devices. As an interface its main task is to convert serial bit stream to block of bytes, perform error correction as necessary.

Any device connected to the computer is connected by a plug and socket, and the socket is connected to a device controller. Following is a model for connecting the CPU, memory, controllers, and I/O devices where CPU and device controllers all use a common bus for communication.



Types of Buses and interfaces in Computer Architecture

Inside computers, there are many internal components. In order for these components to communicate with each other they make use of wires that are known as a ***'bus'*** .

A bus is a common pathway through which information flows from one computer component to another. This pathway is used for communication purpose and it is established between two or more computer components. We are going to check different **computer bus architectures** that are found in computers.

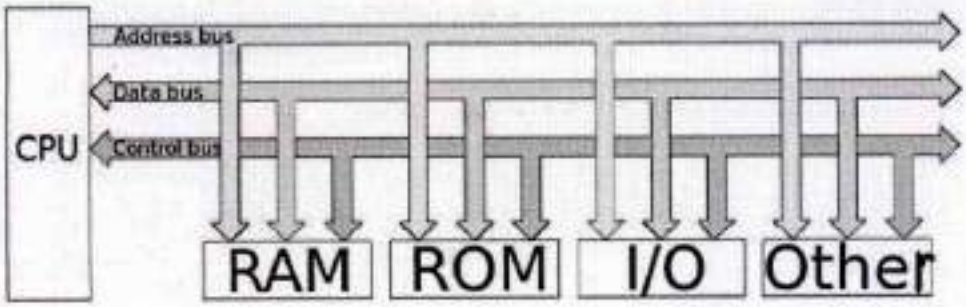
Different Types of Computer Buses

Question Paper Solution

Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....

Mid Term: I

Submitted By :.....Sanju Choudhary.....



The Computer Buses | Source

Functions of Buses in Computers

Summary of functions of buses in computers
1. **Data sharing** - All types of buses found in a computer transfer data between the computer peripherals connected to it.

The buses transfer or send data in either serial or parallel method of data transfer. This allows for the exchange of 1, 2, 4 or even 8 bytes of data at a time. (A byte is a group of 8 bits). Buses are classified depending on how many bits they can move at the same time, which means that we have 8-bit, 16-bit, 32-bit or even 64-bit buses.

2. Addressing - A bus has address lines, which match those of the processor. This allows data to be sent to or from specific memory locations.

3. Power - A bus supplies power to various peripherals connected to it.

4. Timing - The bus provides a **system clock** signal to synchronize the peripherals attached to it with the rest of the system.

The expansion bus facilitates easy connection of more or additional components and devices on a computer such as a TV card or sound card.

Bus Terminologies

Computers have two major types of buses:
1. System bus:- This is the bus that connects the CPU to main memory on the motherboard. The system bus is also called the front-side bus, memory bus, local bus, or host bus.
2. A number of I/O Buses, (I/O is an acronym for input / output), connecting various peripheral devices to the CPU. These devices connect to the system bus via a 'bridge' implemented in the processors chipset. Other names for the I/O bus include "expansion bus", "external bus" or "host bus".

Expansion Bus Types

These are some of the common expansion bus types that have ever been used in computers:

- **ISA** - Industry Standard Architecture
- **EISA** - Extended Industry Standard Architecture

Question Paper Solution

Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....

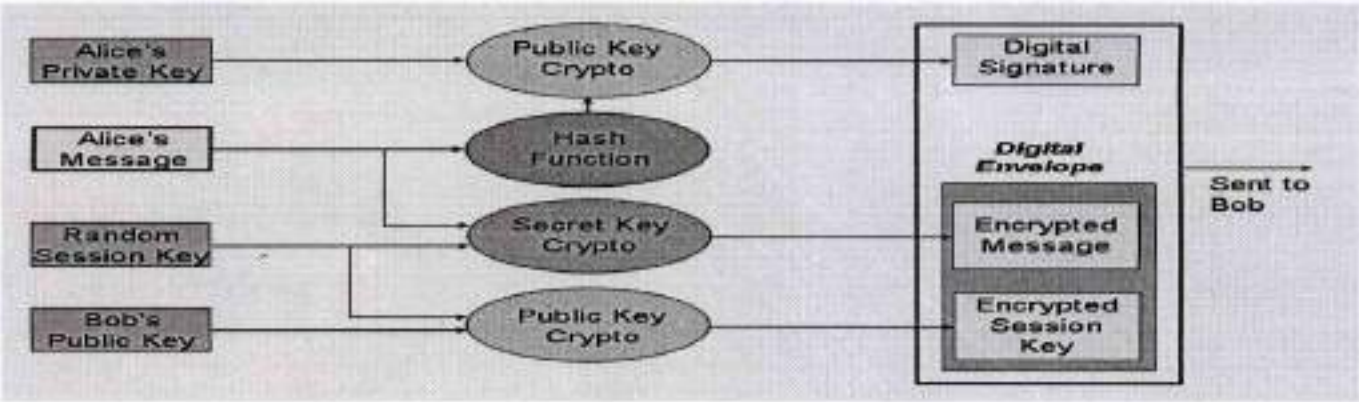
Mid Term: I

Submitted By :.....Sanju Choudhary.....

- **MCA** - Micro Channel Architecture
- **VESA** - Video Electronics Standards Association
- **PCI** - Peripheral Component Interconnect
- **PCMCIA** - Personal Computer Memory Card Industry Association (Also called PC bus)
- **AGP** - Accelerated Graphics Port
- **SCSI** - Small Computer Systems Interface.

Q3. What is Cryptography? Differentiate between symmetric and asymmetric encryption with an example.

Ans. Cryptography is associated with the process of converting ordinary plain text into unintelligible text and vice-versa. It is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it. Cryptography not only protects data from theft or alteration, but can also be used for user authentication.



Description: Earlier cryptography was effectively synonymous with encryption but nowadays cryptography is mainly based on mathematical theory and computer science practice.

Modern cryptography concerns with:

Confidentiality - Information cannot be understood by anyone

Integrity - Information cannot be altered.

Non-repudiation - Sender cannot deny his/her intentions in the transmission of the information at a later stage

Authentication - Sender and receiver can confirm each

Cryptography is used in many applications like banking transactions cards, computer passwords, and e- commerce transactions.

Three types of cryptographic techniques used in general.

1. Symmetric-key cryptography

2. Hash functions.

3. Public-key cryptography

Symmetric-key Cryptography: Both the sender and receiver share a single key. The sender uses this key to encrypt plaintext and send the cipher text to the receiver. On the other side the receiver applies the same key to decrypt the message and recover the plain text.

Public-Key Cryptography: This is the most revolutionary concept in the last 300-400 years. In Public-Key Cryptography two related keys (public and private key) are used. Public key may be freely distributed, while its paired private key, remains a secret. The public key is used for encryption and for decryption private key is used.

Hash Functions: No key is used in this algorithm. A fixed-length hash value is computed as per the plain text that makes it impossible for the contents of the plain text to be recovered. Hash functions are also used by many operating systems to encrypt passwords.

What is RAID system structure? Explain the role of boot block, bad bock, and partition control block management in Linux and windows os.

Ans. RAID (redundant array of independent disks)

RAID (*redundant array of independent disks*; originally *redundant array of inexpensive disks*) provides a way of storing the same data in different places (thus, redundantly) on multiple hard disks (though not all RAID levels provide redundancy). By placing data on multiple disks, input/output (I/O) operations can overlap in a balanced way, improving performance. Since multiple disks increase the mean time between failures (MTBF),

storing data redundantly also increases fault tolerance.

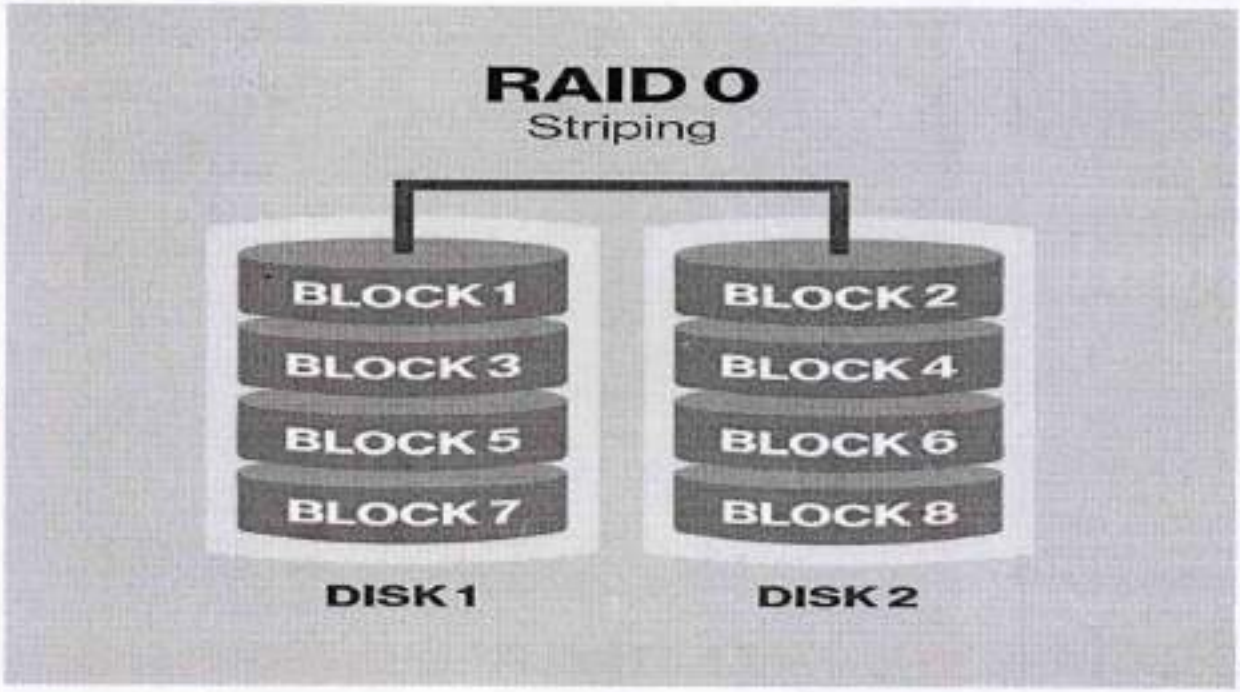
RAID arrays appear to the operating system (OS) as a single logical hard disk. RAID employs the technique of disk mirroring or disk striping, which involves partitioning each drive's storage space into units ranging from a sector (512 bytes) up to several megabytes. The stripes of all the disks are interleaved and addressed in order.

In a single-user system where large records, such as medical or other scientific images, are stored, the stripes are typically set up to be small (perhaps 512 bytes) so that a single record spans all disks and can be accessed quickly by reading all disks at the same time.

In a multi-user system, better performance requires establishing a stripe wide enough to hold the typical or maximum size record. This allows overlapped disk I/O across drives.

Standard RAID levels

RAID 0: This configuration has striping but no redundancy of data. It offers the best performance but no fault-tolerance.



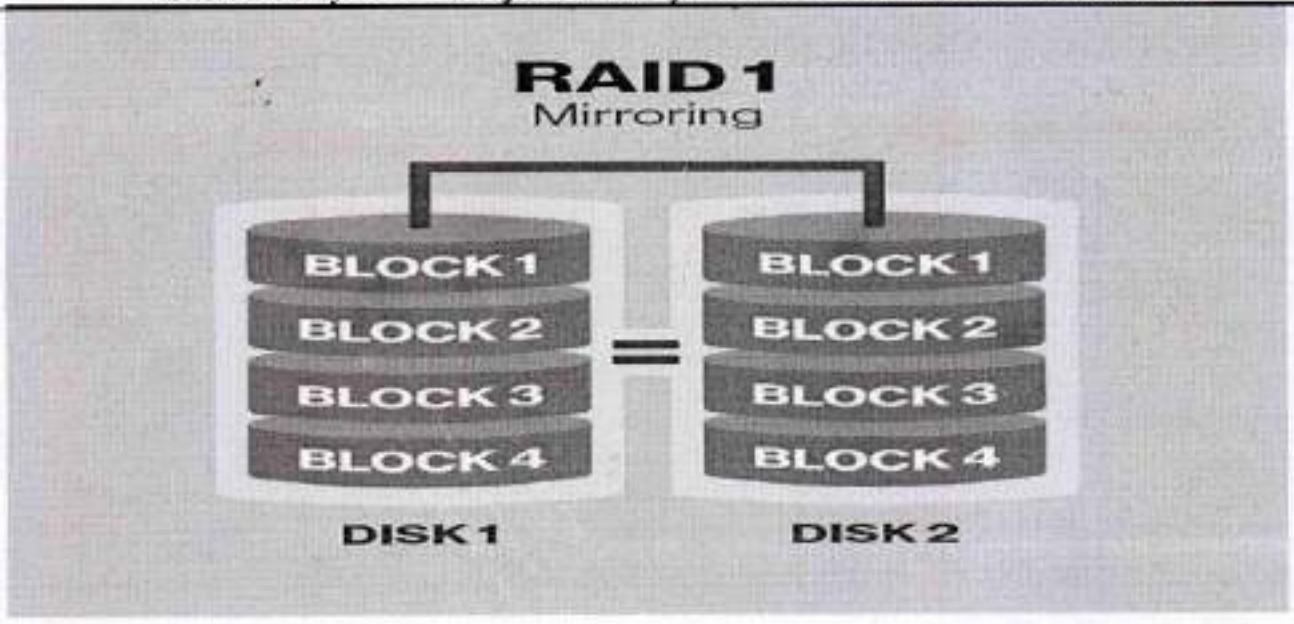
RAID 1: Also known as *disk mirroring*, this configuration consists of at least two drives that duplicate the storage of data. There is no striping. Read performance is improved since either disk can be read at the same time. Write performance is the same as for single disk storage.

Question Paper Solution

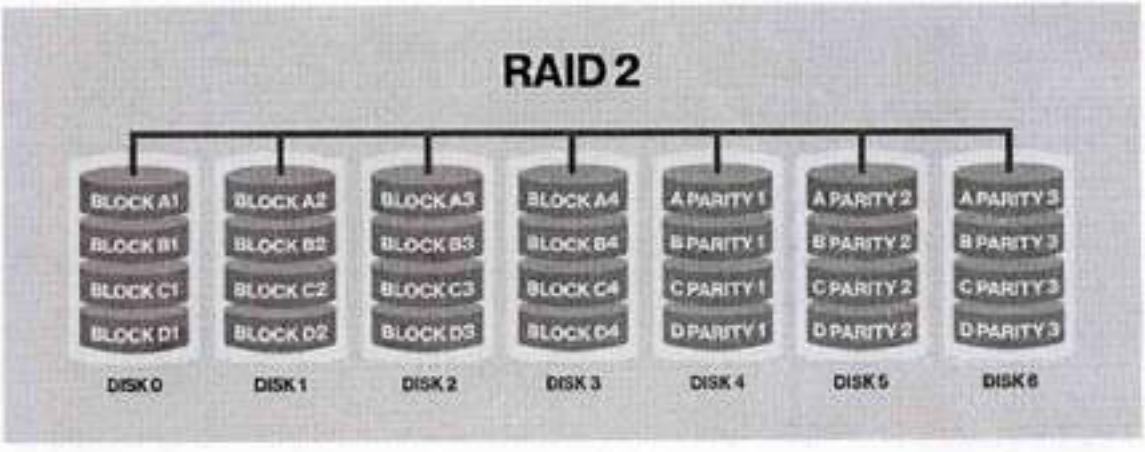
Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....

Mid Term: I

Submitted By :.....Sanju Choudhary.....



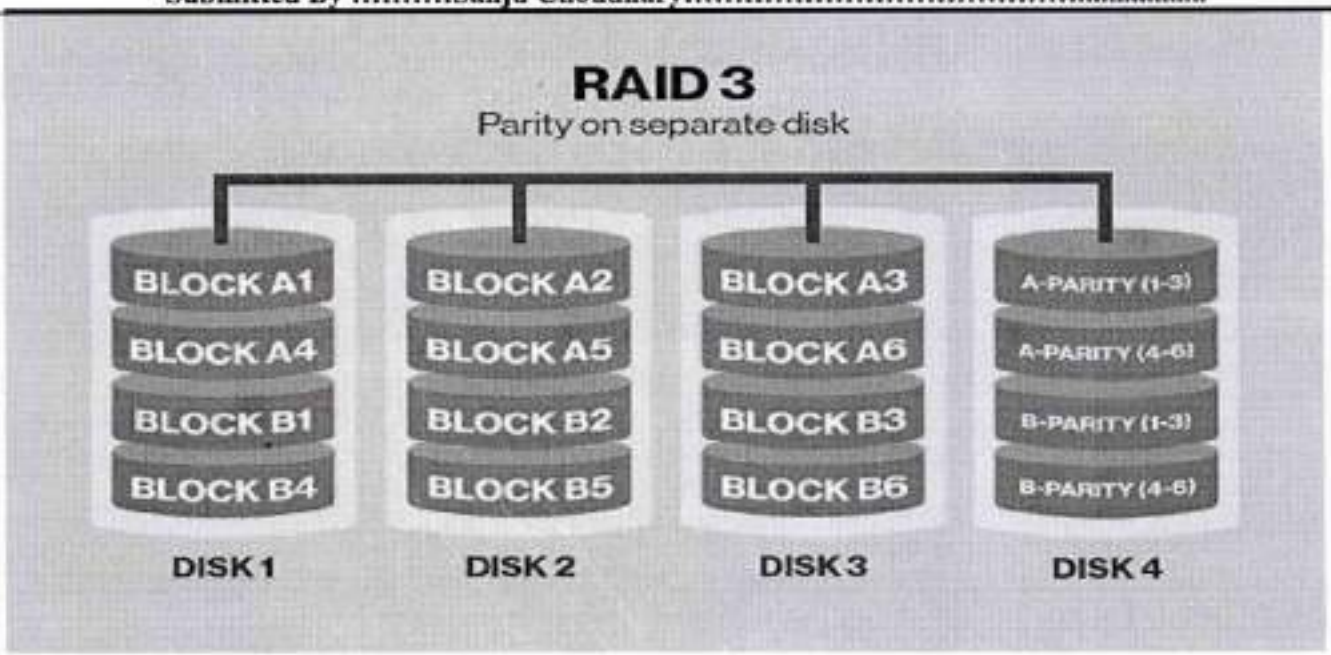
RAID 2: This configuration uses striping across disks with some disks storing error checking and correcting (ECC) information. It has no advantage over RAID 3 and is no longer used.



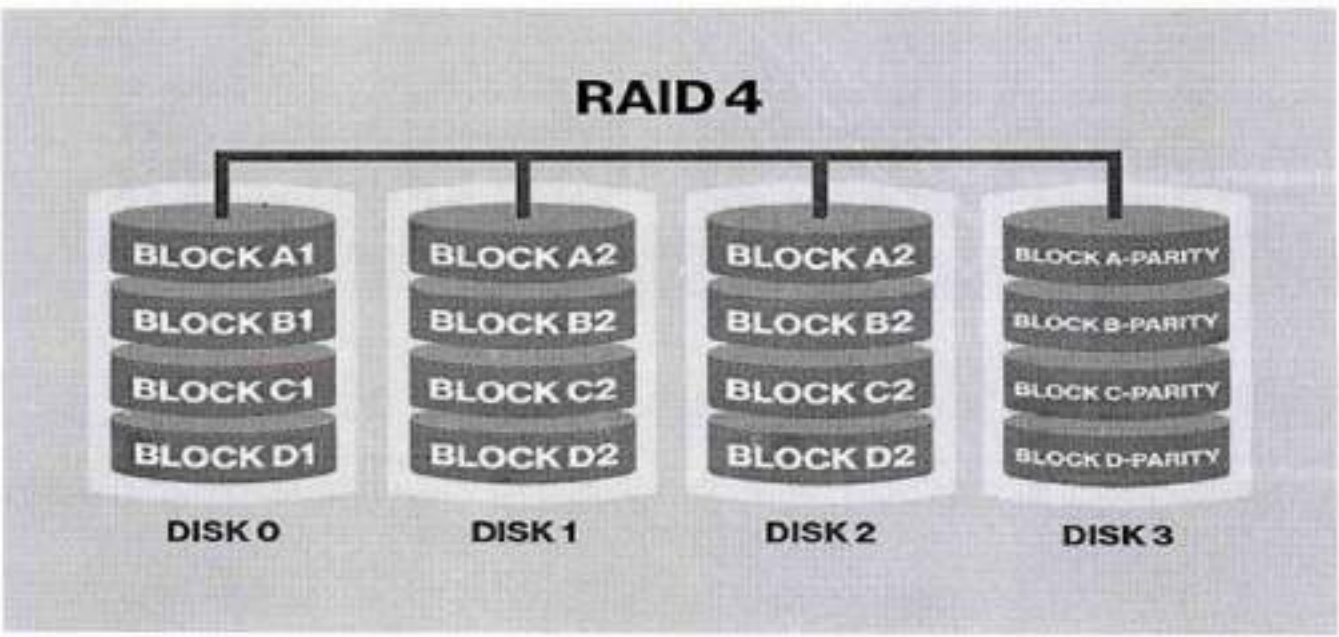
RAID 3: This technique uses striping and dedicates one drive to storing parity information. The embedded ECC information is used to detect errors. Data recovery is accomplished by calculating the exclusive OR (XOR) of the information recorded on the other drives. Since an I/O operation addresses all drives at the same time, RAID 3 cannot overlap I/O. For this reason, RAID 3 is best for single-user systems with long record applications.

Question Paper Solution

Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....
Mid Term: I
Submitted By :.....Sanju Choudhary.....



RAID 4: This level uses large stripes, which means you can read records from any single drive. This allows you to use overlapped I/O for read operations. Since all write operations have to update the parity drive, no I/O overlapping is possible. RAID 4 offers no advantage over RAID 5.



RAID 5: This level is based on block-level striping with parity. The parity information is striped across each drive, allowing the array to function even if one drive were to fail. The array's architecture allows read and write operations to span multiple drives. This results in performance that is usually better than that of a single drive, but not as high as that of a RAID 0 array. RAID 5 requires at least three disks, but it is often recommended to use at least five disks for performance reasons.

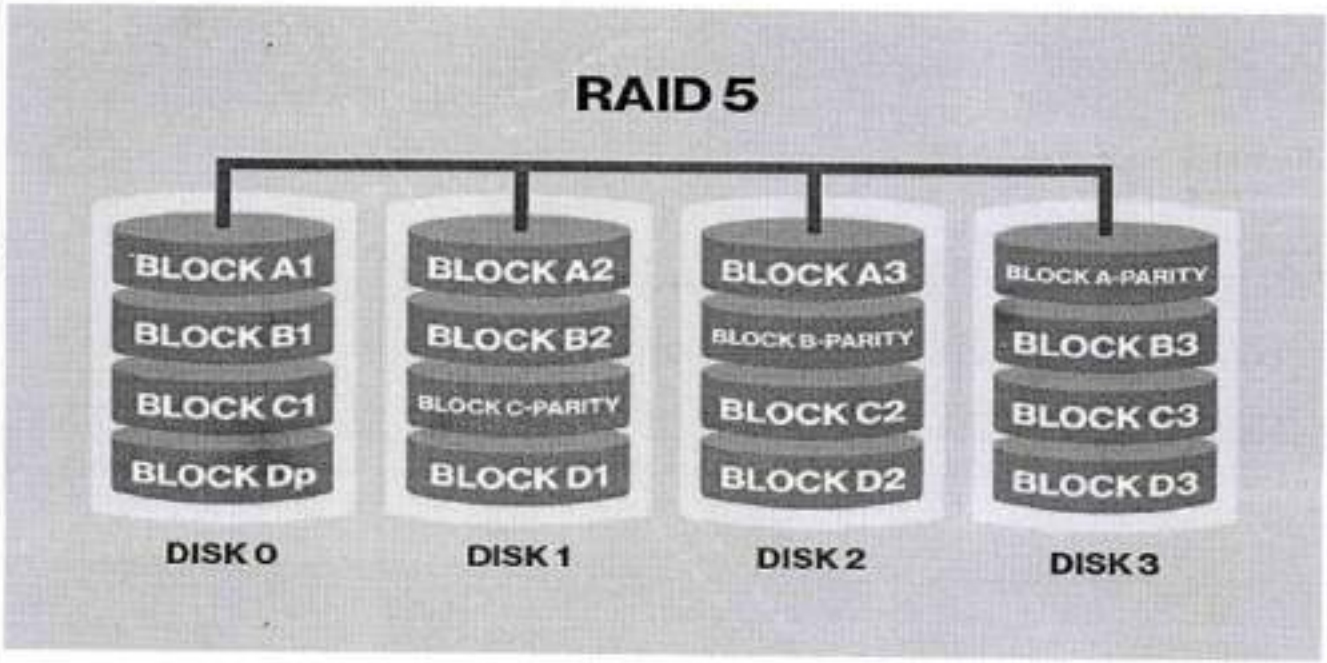
Question Paper Solution

Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....

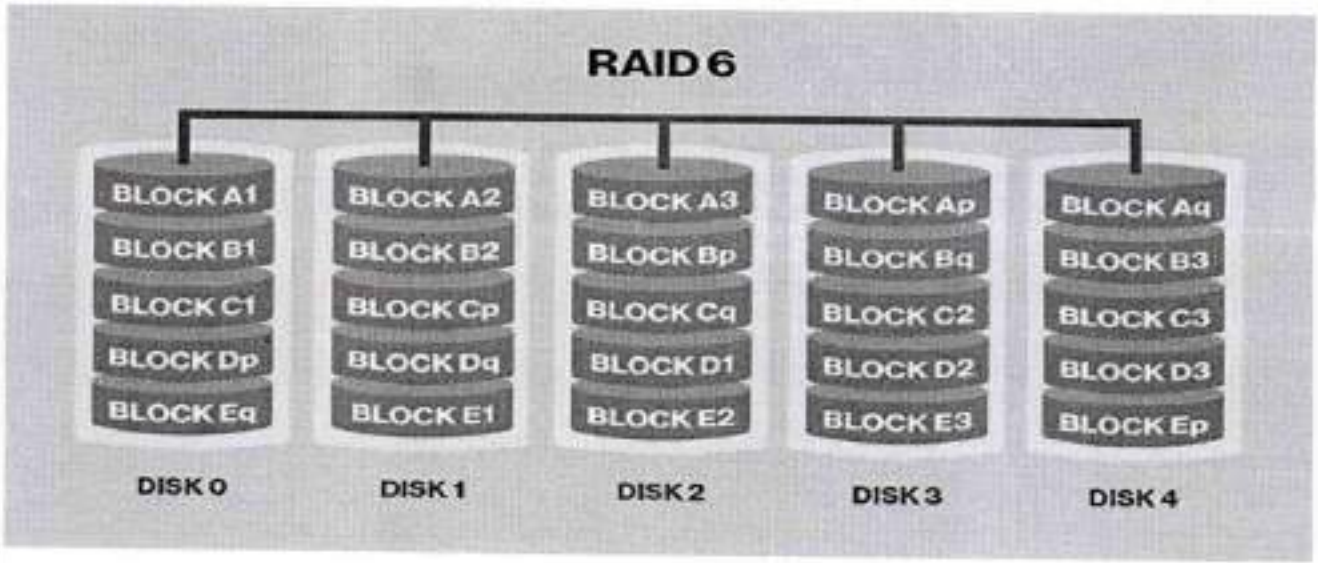
Mid Term: I

Submitted By :.....Sanju Choudhary.....

RAID 5 arrays are generally considered to be a poor choice for use on write-intensive systems because of the performance impact associated with writing parity information. When a disk does fail, it can take a long time to rebuild a RAID 5 array. Performance is usually degraded during the rebuild time and the array is vulnerable to an additional disk failure until the rebuild is complete.



RAID 6: This technique is similar to RAID 5 but includes a second parity scheme that is distributed across the drives in the array. The use of additional parity allows the array to continue to function even if two disks fail simultaneously. However, this extra protection comes at a cost. RAID 6 arrays have a higher cost per gigabyte (GB) and often have slower write performance than RAID 5 arrays.



Nested RAID levels

Question Paper Solution

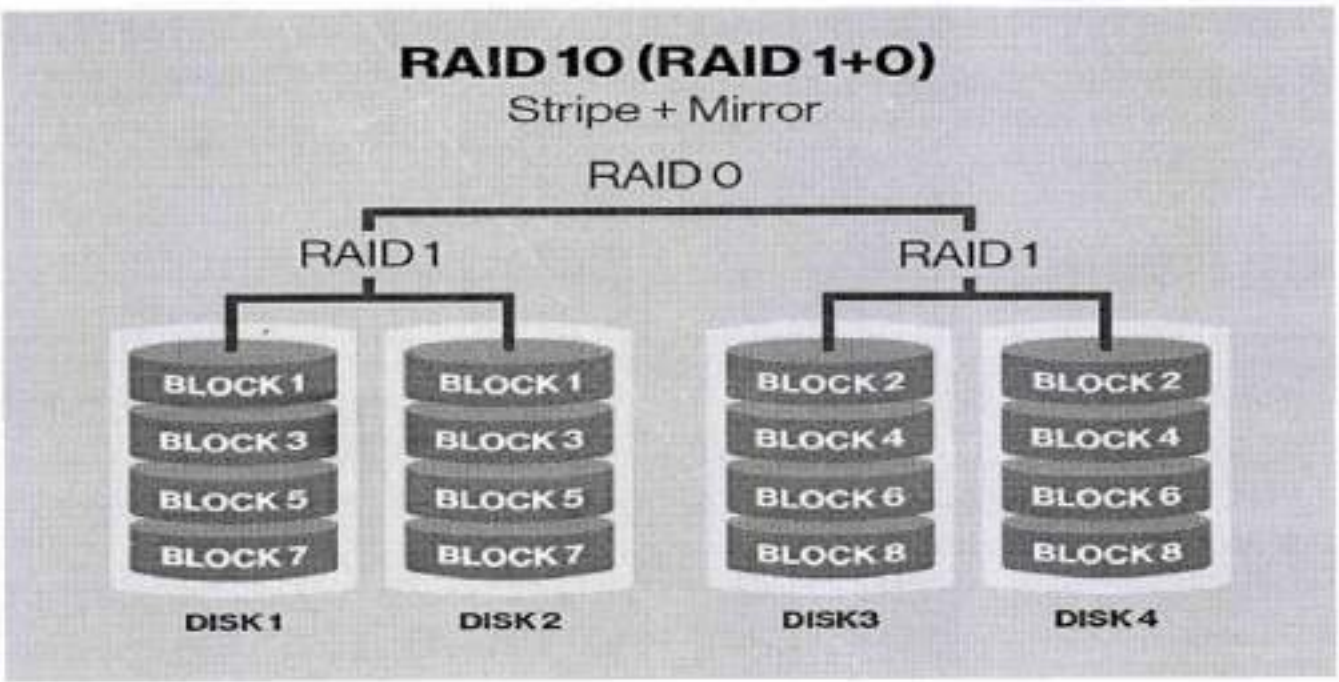
Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....

Mid Term: I

Submitted By :.....Sanju Choudhary.....

Some RAID levels are referred to as *nested RAID* because they are based on a combination of RAID levels. Here are some examples of nested RAID levels.

RAID 10 (RAID 1+0): Combining RAID 1 and RAID 0, this level is often referred to as RAID 10, which offers higher performance than RAID 1 but at a much higher cost. In RAID 1+0, the data is mirrored and the mirrors are striped.



RAID 01 (RAID 0+1): RAID 0+1 is very similar to RAID 1+0, except the data organization method is slightly different. Rather than creating a mirror and then stripping the mirror, RAID 0+1 creates a stripe set and then mirrors the stripe set.

RAID 03 (RAID 0+3 also known as RAID 53 or RAID 5+3): This level uses striping (in RAID 0 style) for RAID 3's virtual disk blocks. This offers higher performance than RAID 3, but at a much higher cost.

RAID 50 (RAID 5+0): This configuration combines RAID 5 distributed parity with RAID 0 striping to improve RAID 5 performance without reducing data protection.

Non-standard RAID levels

RAID 7: This RAID level is based on RAID 3 and RAID 4, but adds caching to the mix. It includes a real-time embedded OS as a controller, caching via a high-speed bus and other characteristics of a standalone computer. It is a non-standard, trademarked RAID level owned by the now defunct Storage Computer Corp.

Adaptive RAID: Adaptive RAID lets the RAID controller decide how to store the parity on the disks. It will choose between RAID 3 and RAID 5, depending on which RAID set type will perform better with the type of data being written to the disks.

Question Paper Solution

Branch : .....IT..... Semester: .....VI... Subject: .....ATOS.....

Mid Term: I

Submitted By :.....Sanju Choudhary.....

RAID S (also known as Parity RAID): This is an alternate, proprietary method for striped parity RAID from EMC Symmetrix that is no longer in use on current equipment. It appears to be similar to RAID 5 with some performance enhancements, as well as the enhancements that come from having a high-speed disk cache on the disk array.