**Bangladesh Civil Service – 36th (ICT)**
**Date: 2016 Time: 3 Hours. Full marks: 200**

**1.Write a Sorting Algorithm to sort "n" number always in O(nlogn) time**

**MergeSort(arr[], l, r)**
If r > l
1. Find the middle point to divide the array into two halves:
   middle m = l+ (r-l)/2
2. Call mergeSort for first half:
   Call mergeSort(arr, l, m)
3. Call mergeSort for second half:
   Call mergeSort(arr, m+1, r)
4. Merge the two halves sorted in step 2 and 3:
   Call merge(arr, l, m, r)

**2.What are the main functionality of "prority Queue"? How to implement these functions by using heap ?**
Priority Queue is an extension of queue with following properties.

Every item has a priority associated with it.
An element with high priority is dequeued before an element with low priority.
If two elements have the same priority, they are served according to their order in the queue.
In the below priority queue, element with maximum ASCII value will have the highest priority.A typical priority queue supports following operations.
insert(item, priority): Inserts an item with given priority.
getHighestPriority(): Returns the highest priority item.
deleteHighestPriority(): Removes the highest priority item.

**How to implement priority queue?**
Using Array: A simple implementation is to use array of following structure.

```
struct item {
  int item;
  int priority;
}
```
insert() operation can be implemented by adding an item at end of array in O(1) time.

getHighestPriority() operation can be implemented by linearly searching the highest priority item in array. This operation takes O(n) time.

deleteHighestPriority() operation can be implemented by first linearly searching an item, then removing the item by moving all subsequent items one position back.

We can also use Linked List, time complexity of all operations with linked list remains same as array. The advantage with linked list is deleteHighestPriority() can be more efficient as we don't have to move items.

Using Heaps:
Heap is generally preferred for priority queue implementation because heaps provide better performance compared arrays or linked list. In a Binary Heap, getHighestPriority() can be implemented in O(1) time, insert() can be implemented in O(Logn) time and deleteHighestPriority() can also be implemented in O(Logn) time.
With Fibonacci heap, insert() and getHighestPriority() can be implemented in O(1) amortized time and deleteHighestPriority() can be implemented in O(Logn) amortized time.

**3.What is Race Condition ? How to remove this ?**
A race condition occurs when two or more threads can access shared data and they try to change it at the same time. Because the thread scheduling algorithm can swap between threads at any time, you don't know the order in which the threads will attempt to access the shared data. Therefore, the result of the change in data is dependent on the thread scheduling algorithm, i.e. both threads are "racing" to access/change the data.

Problems often occur when one thread does a "check-then-act" (e.g. "check" if the value is X, then "act" to do something that depends on the value being X) and another thread does something to the value in between the "check" and the "act". E.g:

```
if (x == 5) // The "Check"
{
   y = x * 2; // The "Act"

   // If another thread changed x in between "if (x == 5)" and "y = x * 2" above,
   // y will not be equal to 10.
}
```
The point being, y could be 10, or it could be anything, depending on whether another thread changed x in between the check and act. You have no real way of knowing.

In order to prevent race conditions from occurring, you would typically put a lock around the shared data to ensure only one thread can access the data at a time. This would mean something like this:

```
// Obtain lock for x
if (x == 5)
{
   y = x * 2; // Now, nothing can change x until the lock is released.
              // Therefore y = 10
}
// release lock for x
```

**4.Describe "Huffman's" Algorithm.**
**Algorithm**

**huffmanCoding(string)**

**Input:** A string with different characters.

**Output:** The codes for each individual characters.

```
Begin
  define a node with character, frequency, left and right child of the node for Huffman tree.
  create a list 'freq' to store frequency of each character, initially, all are 0
  for each character c in the string do
    increase the frequency for character ch in freq list.
  done

  for all type of character ch do
    if the frequency of ch is non zero then
      add ch and its frequency as a node of priority queue Q.
  done

  while Q is not empty do
    remove item from Q and assign it to left child of node
    remove item from Q and assign to the right child of node
    traverse the node to find the assigned code
  done
End
```

**traverseNode(n: node, code)**

**Input:** The node n of the Huffman tree, and the code assigned from the previous call

**Output:** Code assigned with each character

```
if a left child of node n ≠φ then
  traverseNode(leftChild(n), code+'0')     //traverse through the left child
  traverseNode(rightChild(n), code+'1')    //traverse through the right child
else
  display the character and data of current node.
```

**5.How to convert a J-K Flip Flop to a T Flip Flop ?**
**J-K Flip-Flop:**
J-K flip-flop is the gated version of Sr flip-flop with an addition of extra input i.e. clock input. It prevents invalid output condition when both the inputs are at the same value.

**2. T Flip-Flop:**

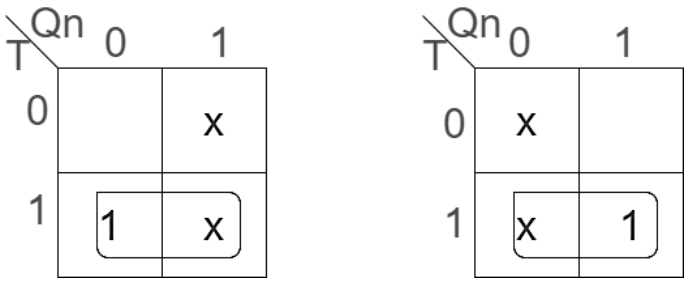
T flip-flop means Toggle flip-flop. It changes the output on each clock edge and gives an output which is half the frequency of the signal to the input.

**Conversion of J-K Flip-Flop into T Flip-Flop:**

- **Step-1:**
  Construct the characteristic table of T flip-flop and excitation table of J-K flip-flop.

| T | Qn | Qn+1 | J | K |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | x |
| 0 | 1 | 1 | x | 0 |
| 1 | 0 | 1 | 1 | x |
| 1 | 1 | 0 | x | 1 |

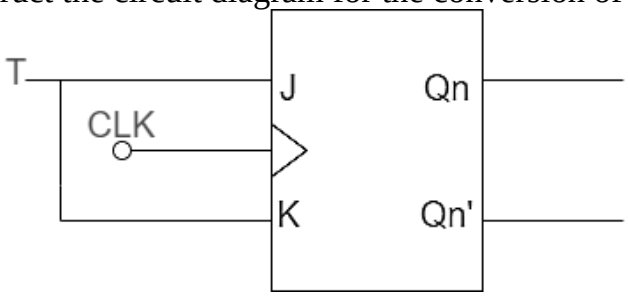
- **Step-2:**
  Using K map, find the boolean expression for J and K in terms of T.




  J = T

  K = T

- **Step-3:**
  Construct the circuit diagram for the conversion of J-K flip-flop into T flip-flop.




**6.Explain Early Binding and Late Binding with example ?**

Early binding : Method Overloading

Late Overloading: Virtual Function

**7.Explain Method overloading and Method overriding with example .**

See Previous Year.

**8.Write Pseudocode to implement Enqueue and Dequeue operation array of C.**

**9.What is Encapsulation, inheritance and Polymorphism ?**

**10.What are the differences between Graph & Tree ? What is Hamiltonian Graph ?**
**11. Design a 4 bit binary adder using 1-Bit full adder circuit.**
**12.Distinguish between Latch and Flip Flop**

| SL No | Flip-flop | Latch |
|---|---|---|
| 1 | Flip-flop is a bistable device i.e., it has two stable states that are represented as 0 and 1. | Latch is also a bistable device whose states are also represented as 0 and 1. |
| 2 | It checks the inputs but changes the output only at times defined by the clock signal or any other control signal. | It checks the inputs continuously and responds to the changes in inputs immediately. |
| 3 | It is a edge triggered device | It is a level triggered device. |
| 4 | Gates like NOR, NOT, AND, NAND are building blocks of flip flops. | These are also made up of gates. |
| 5 | They are classified into asynchronous or synchronous flipflops. | There is no such classification in latches. |
| 6 | It forms the building blocks of many sequential circuits like counters. | These can be used for the designing of sequential circuits but are not generally preferred. |
| 7 | Flip-flop always have a clock signal. | latch doesn't have a clock signal |

**13.What are the differences between CISC and RISC architecture ?**
Solution:

| SL No | RISC | CISC |
|---|---|---|
| 1 | Focus on software | Focus on hardware |
| 2 | Transistors are used for more registers | Transistors are used for storing complex Instructions |
| 3 | Code size is large | Code size is small |
| 4 | A instruction execute in single clock cycle | Instruction take more than one clock cycle |
| 5 | A instruction fit in one word | Instruction are larger than size of one word |

**14.Describe the architecture of Intel 8086 Micrprocessor.**
The 8086 CPU is divided into two independent functional units:

- Bus Interface Unit (BIU)
- Execution Unit (EU)

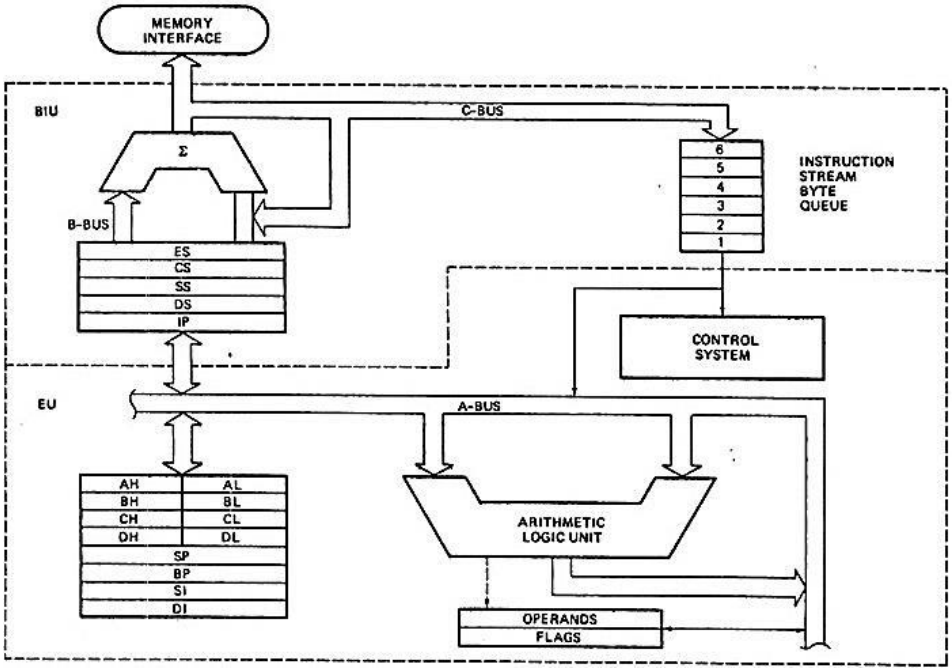



**Fig. 1: Block Diagram of Intel 8086**

**Features of 8086 Microprocessor:**

- Intel 8086 was launched in 1978.
- It was the first 16-bit microprocessor.
- This microprocessor had major improvement over the execution speed of 8085.
- It is available as 40-pin Dual-Inline-Package (DIP).
- It is available in three versions:
  - 8086 (5 MHz)
  - 8086-2 (8 MHz)
  - 8086-1 (10 MHz)
- It consists of 29,000 transistors.

**Bus Interface Unit (BIU)**

The function of BIU is to:

- Fetch the instruction or data from memory.
- Write the data to memory.
- Write the data to the port.
- Read data from the port.

**Instruction Queue**

1. To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory.
2. All six bytes are then held in first in first out 6 byte register called instruction queue.
3. Then all bytes have to be given to EU one by one.
4. This pre fetching operation of BIU may be in parallel with execution operation of EU, which improves the speed execution of the instruction.

**Execution Unit (EU)**

The functions of execution unit are:

- To tell BIU where to fetch the instructions or data from.
- To decode the instructions.
- To execute the instructions.

  The EU contains the control circuitry to perform various internal operations. A decoder in EU decodes the instruction fetched memory to generate different internal or external control signals required to perform the operation. EU has 16-bit ALU, which can perform arithmetic and logical operations on 8-bit as well as 16-bit.

**General Purpose Registers of 8086**

These registers can be used as 8-bit registers individually or can be used as 16-bit in pair to have AX, BX, CX, and DX.

1. **AX Register:** AX register is also known as accumulator register that stores operands for arithmetic operation like divided, rotate.
2. **BX Register:** This register is mainly used as a base register. It holds the starting base location of a memory region within a data segment.
3. **CX Register:** It is defined as a counter. It is primarily used in loop instruction to store loop counter.
4. **DX Register:** DX register is used to contain I/O port address for I/O instruction.

**Segment Registers**

Additional registers called segment registers generate memory address when combined with other in the microprocessor. In 8086 microprocessor, memory is divided into 4 segments as follow:

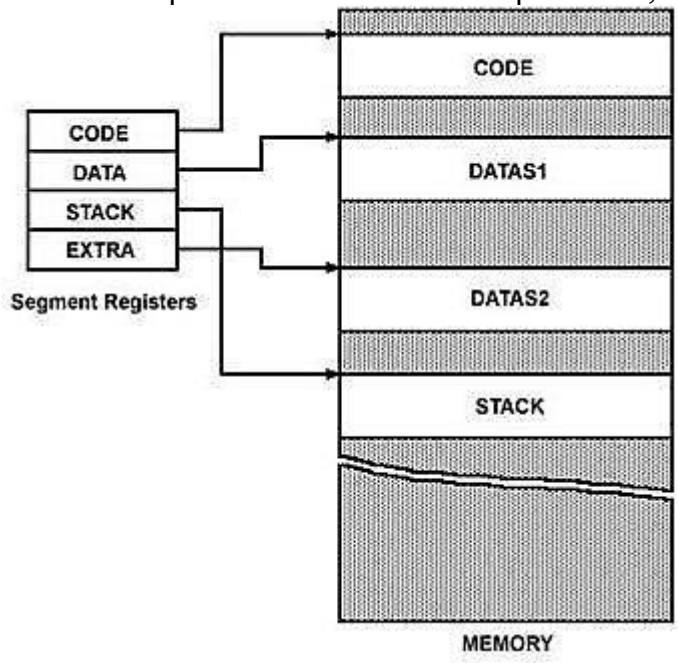



**Fig. 2: Memory Segments of 8086**

1. **Code Segment (CS):** The CS register is used for addressing a memory location in the Code Segment of the memory, where the executable program is stored.
2. **Data Segment (DS):** The DS contains most data used by program. Data are accessed in the Data Segment by an offset address or the content of other register that holds the offset address.
3. **Stack Segment (SS):** SS defined the area of memory used for the stack.
4. **Extra Segment (ES):** ES is additional data segment that is used by some of the string to hold the destination data.

**Flag Registers of 8086**

Flag register in EU is of 16-bit and is shown in fig. 3:

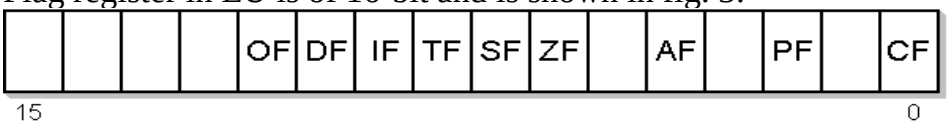



**Fig. 3: Flag Register of 8086**

Flags Register determines the current state of the processor. They are modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program. 8086 has 9 flags and they are divided into two categories:

1. Conditional Flags
2. Control Flags

**Conditional Flags**

Conditional flags represent result of last arithmetic or logical instruction executed. Conditional flags are as follows:

- **Carry Flag (CF):** This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.
- **Auxiliary Flag (AF):** If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AF flag is set i.e. carry given by D3 bit to D4 is AF flag. This is not a general-purpose flag, it is used internally by the processor to perform Binary to BCD conversion.
- **Parity Flag (PF):** This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity Flag is reset.
- **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.
- **Sign Flag (SF):** In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.
- **Overflow Flag (OF):** It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine.

**Control Flags**

Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

1. **Trap Flag (TP):**

a. It is used for single step control.
b. It allows user to execute one instruction of a program at a time for debugging.
c. When trap flag is set, program can be run in single step mode.

2. **Interrupt Flag (IF):**

a. It is an interrupt enable/disable flag.
b. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.
c. It can be set by executing instruction sit and can be cleared by executing CLI instruction.

3. **Direction Flag (DF):**

a. It is used in string operation.
b. If it is set, string bytes are accessed from higher memory address to lower memory address.
c. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

**15.What are the differences between C++ and Java Programming Language ?**

| **Index** | | |
|---|---|---|
| **Platform-independent** | C++ is platform-dependent. | Java is platform-independent. |
| **Mainly used for** | C++ is mainly used for system programming. | Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications. |
| **Design Goal** | C++ was designed for systems and applications programming. It was an extension of C programming language. | Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience. |
| **Goto** | C++ supports the goto statement. | Java doesn't support the goto statement. |
| **Multiple inheritance** | C++ supports multiple inheritance. | Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java |
| **Operator Overloading** | C++ supports operator overloading. | Java doesn't support operator overloading. |
| **Pointers** | C++ supports pointers. You can write pointer program in C++. | Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java. |
| **Compiler and Interpreter** | C++ uses compiler only. C++ is compiled and run using the compiler | Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The |

| | | |
|---|---|---|
| | which converts source code into machine code so, C++ is platform | interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why |
| **Call by Value and Call by reference** | C++ supports both call by value and call by reference. | Java supports call by value only. There is no call by reference in java. |

**16.How many permutations are possible from 26 Alphabet letters but they don't have match with car ,dog and byte ?**
**17.Explain how Parallel Processing does work?**

Parallel processing can be described as a class of techniques which enables the system to achieve simultaneous data-processing tasks to increase the computational speed of a computer system.

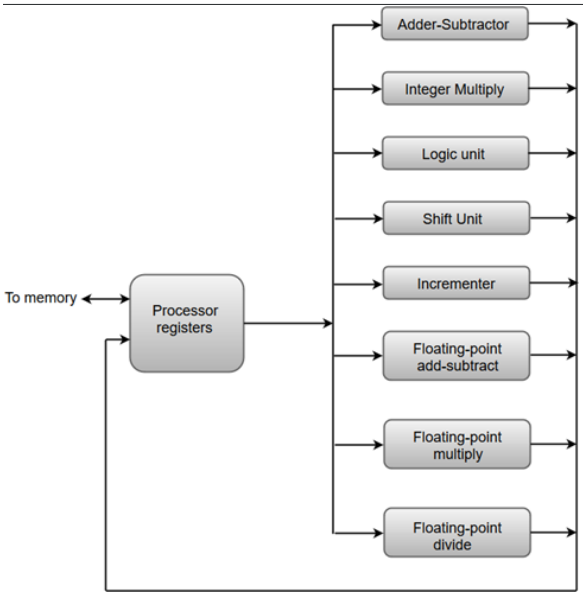
A parallel processing system can carry out simultaneous data-processing to achieve faster execution time. For instance, while an instruction is being processed in the ALU component of the CPU, the next instruction can be read from memory.

The primary purpose of parallel processing is to enhance the computer processing capability and increase its throughput, i.e. the amount of processing that can be accomplished during a given interval of time.

A parallel processing system can be achieved by having a multiplicity of functional units that perform identical or different operations simultaneously. The data can be distributed among various multiple functional units.

The following diagram shows one possible way of separating the execution unit into eight functional units operating in parallel.

The operation performed in each functional unit is indicated in each block if the diagram:




- The adder and integer multiplier performs the arithmetic operation with integer numbers.
- The floating-point operations are separated into three circuits operating in parallel.
- The logic, shift, and increment operations can be performed concurrently on different data. All units are independent of each other, so one number can be shifted while another number is being incremented.

**18.What is interrupt Controller?**

An Interrupt Control is usually used in Micro controllers to generate interrupts signals which tells the CPU to pause its current task and start executing another set of predefined activities.

**19.Write a Program of basic calculator operation function using switch statement (+.-,*,/) using any language C/C++**

**20.Write down the differences between propositional logic and Predicative logic?**

| SL | Propositional logic | Predicate logic |
|---|---|---|
| 1 | Propositional logic is the logic that deals with a collection of declarative statements which have a truth value, true or false | Predicate logic is an expression consisting of variables with a specified domain. It consists of objects, relations and functions between the objects |
| 2 | It is the basic and most widely used logic. Also known as Boolean logic. | It is an extension of propositional logic covering predicates and quantification. |
| 3 | A proposition has a specific truth value, either true or false. | A predicate's truth value depends on the variables' value. |
| 4 | Scope analysis is not done in propositional logic. | Predicate logic helps analyze the scope of the subject over the predicate. There are three quantifiers : Universal Quantifier (∀) depicts for all, Existential Quantifier (∃) |

| | | |
|---|---|---|
| | | depicting there exists some and Uniqueness Quantifier (∃!) depicting exactly one. |
| **5** | Propositions are combined with Logical Operators or Logical Connectives like Negation(¬), Disjunction(∧), Conjunction(∨), Exclusive OR(⊕), Implication(⇒), Bi-Conditional or Double Implication(⇔). | Predicate Logic adds by introducing quantifiers to the existing proposition. |
| **6** | It is a more generalized representation. | It is a more specialized representation. |
| **7** | It cannot deal with sets of entities. | It can deal with set of entities with the help of quantifiers. |

**21.How can a optical mouse work ?**

An optical mouse works in a completely different way. It shines a bright light down onto your desk from an LED (light-emitting diode) mounted on the bottom of the mouse. The light bounces straight back up off the desk into a photocell (photoelectric cell), also mounted under the mouse, a short distance from the LED. The photocell has a lens in front of it that magnifies the reflected light, so the mouse can respond more precisely to your hand movements. As you push the mouse around your desk, the pattern of reflected light changes, and the chip inside the mouse uses this to figure out how you're moving your hand.

Some optical mice have two LEDs. The first one shines light down onto the desk. The light from that is picked up by the photocell. The second LED lights up a red plastic strip along the back of the mouse so you can see it's working. Most optical mice also have a wheel at the front so you can scroll pages on-screen much faster. You can click the wheel too, so it functions like the third (center) button on a conventional ball mouse.

**22.Explain the functionality of a push down automata**

A pushdown automaton is a way to implement a context-free grammar in a similar way we design DFA for a regular grammar. A DFA can remember a finite amount of information, but a PDA can remember an infinite amount of information.

Basically a pushdown automaton is −

**"Finite state machine" + "a stack"**

A pushdown automaton has three components −

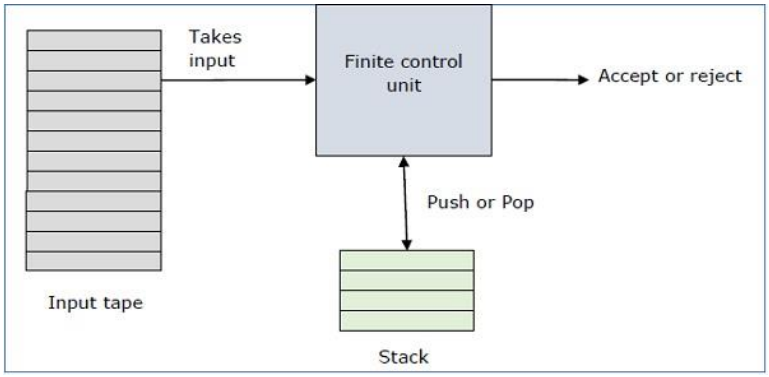
- an input tape,
- a control unit, and
- a stack with infinite size.

The stack head scans the top symbol of the stack.

A stack does two operations −

- **Push** − a new symbol is added at the top.
- **Pop** − the top symbol is read and removed.

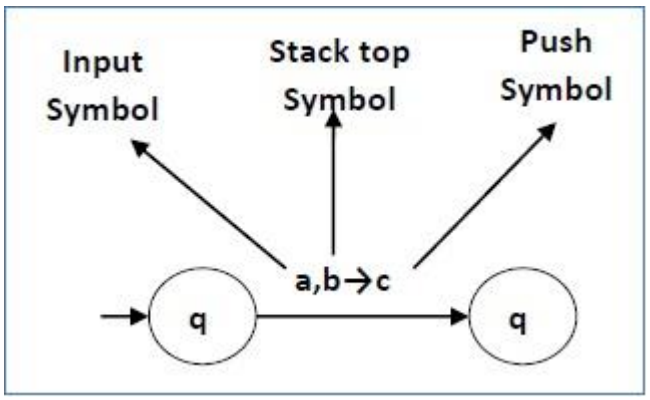
A PDA may or may not read an input symbol, but it has to read the top of the stack in every transition.




A PDA can be formally described as a 7-tuple (Q, ∑, S, δ, $q_0$, I, F) −

- **Q** is the finite number of states
- ∑ is input alphabet
- **S** is stack symbols
- **δ** is the transition function: Q × (∑ ∪ {ε}) × S × Q × S*
- **$q_0$** is the initial state ($q_0$ ∈ Q)
- **I** is the initial stack top symbol (I ∈ S)
- **F** is a set of accepting states (F ∈ Q)

The following diagram shows a transition in a PDA from a state $q_1$ to state $q_2$, labeled as a,b → c −




This means at state **$q_1$**, if we encounter an input string **'a'** and top symbol of the stack is **'b'**, then we pop **'b'**, push **'c'** on top of the stack and move to state **$q_2$**.

**23.How does protected memory Management work ?**

**Memory protection** is a way to control memory access rights on a computer, and is a part of most modern instruction set architectures and operating systems. The main purpose of memory protection is to prevent a process from accessing memory that has not been allocated to it. This prevents a bug or malware within a process from affecting other processes, or the operating system itself. Protection may encompass all accesses to a specified area of memory, write accesses, or attempts to execute the contents of the area. An attempt to access unauthorized[a] memory results in a hardware fault, e.g., a segmentation fault, storage violation exception, generally causing abnormal termination of the offending process. Memory protection for computer security includes additional techniques such as address space layout randomization and executable space protection.

**24.Make a binary search tree using following numbers :28/25/30/15/100/105**
**Try Yourself**

**25.What is Big-O ,Big-Omega ,Big theta ?**

**Big-O:**

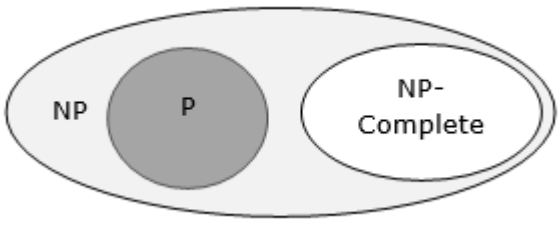
It is like <=
rate of growth of an algorithm is less than or equal to a specific value

**Big-Omega:**
It is like >=
rate of growth is greater than or equal to a specified value

**Big theta:**
It is like ==
meaning the rate of growth is equal to a specified value

**26.Explain NP-hard and NP complete with example.**

A problem is in the class NPC if it is in NP and is as **hard** as any problem in NP. A problem is **NP-hard** if all problems in NP are polynomial time reducible to it, even though it may not be in NP itself.




If a polynomial time algorithm exists for any of these problems, all problems in NP would be polynomial time solvable. These problems are called **NP-complete**. The phenomenon of NP-completeness is important for both theoretical and practical reasons.

**27.What is minimum spanning tree ?**
A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. That is, it is a spanning tree whose sum of edge weights is as small as possible. More generally, any edge-weighted undirected graph (not necessarily connected) has a minimum spanning forest, which is a union of the minimum spanning trees for its connected components.

**28.Write down about Divide and conquer algorithm.What are the differences between divide and conquer and dynamic programming.**

**Divide and Conquer**

Divide and Conquer works by dividing the problem into sub-problems, conquer each sub-problem recursively and combine these solutions.

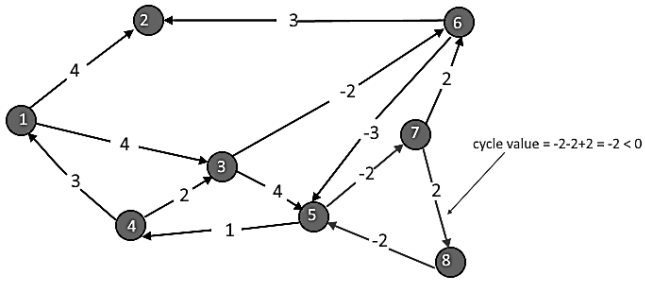
**Dynamic Programming**

Dynamic Programming is a technique for solving problems with overlapping subproblems. Each sub-problem is solved only once and the result of each sub-problem is stored in a table ( generally implemented as an array or a hash table) for future references. These sub-solutions may be used to obtain the original solution and the technique of storing the sub-problem solutions is known as memoization.

You may think of DP = recursion + re-use

A classic example to understand the difference would be to see both these approaches towards obtaining the nth fibonacci number. Check this material from MIT.

**29.Explain how to Bellman Ford Algorithm can identify a negative cycle of a graph.**
We are given a directed graph. We need compute whether the graph has negative cycle or not. A negative cycle is one in which the overall sum of the cycle comes negative.




Negative weights are found in various applications of graphs. For example, instead of paying cost for a path, we may get some advantage if we follow the path.
Examples:

Input : 4 4

0 1 1

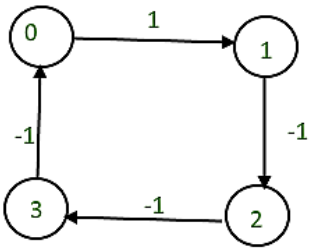
1 2 -1

2 3 -1

3 0 -1

Output : Yes

The graph contains a negative cycle.



The idea is to use Bellman Ford Algorithm.

Below is algorithm find if there is a negative weight cycle reachable from given source.

**1)** Initialize distances from source to all vertices as infinite and distance to source itself as 0. Create an array dist[] of size |V| with all values as infinite except dist[src] where src is source vertex.
**2)** This step calculates shortest distances. Do following |V|-1 times where |V| is the number of vertices in given graph.
…..**a)** Do following for each edge u-v
……If dist[v] > dist[u] + weight of edge uv, then update dist[v]
…….dist[v] = dist[u] + weight of edge uv
**3)** This step reports if there is a negative weight cycle in graph. Do following for each edge u-v
……If dist[v] > dist[u] + weight of edge uv, then "Graph contains negative weight cycle"

The idea of step 3 is, step 2 guarantees shortest distances if graph doesn't contain negative weight cycle. If we iterate through all edges one more time and get a shorter path for any vertex, then there is a negative weight cycle.

**30.Describe agile Model**
Agile Model
The meaning of Agile is swift or versatile."**Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning. The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.
Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Phases of Agile Model:
Following are the phases in the Agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback

**31.Explain each component of Unit testing Environment.**

The Unit Testing Techniques are mainly categorized into three parts which are Black box testing that involves testing of user interface along with input and output, White box testing that involves

testing the functional behaviour of the software application and Gray box testing that is used to execute test suites, test methods, test cases and performing risk analysis.

Code coverage techniques used in Unit Testing are listed below:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Condition Coverage
- Finite State Machine Coverage

**32.What are the reasons of failure of a Software Project ?**

Not Enough Time. ...
Insufficient Budget. ...
Poor Communication. ...
Never Reviewing Project Progress. ...
Inadequate Testing. ...
Testing in the Production Environment. ...
Lack of Quality Assurance. ...
Not Conforming to Industry Standards.

**33.Describe the disadvantages of prototype technique**

**Prototype Model: Prototype** is a working model of software with some limited functionality. The prototype Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered. It offers a small scale facsimile of the end product and is used for obtaining customer feedback as described below:

The Prototyping Model is one of the most popularly used Software Development Life Cycle Models (SDLC models). This model is used when the customers do not know the exact project requirements beforehand. In this model, a prototype of the end product is first developed, tested and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

.

**Following is a stepwise approach explained to design a software prototype.**

**Basic Requirement Identification**

This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.

**Developing the initial Prototype**

The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed. While, the workarounds are used to give the same look and feel to the customer in the prototype developed.

**Review of the Prototype**

The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.

**Revise and Enhance the Prototype**

The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like – time and budget constraints and technical feasibility of the actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until the customer expectations are met.

Prototypes can have horizontal or vertical dimensions. A Horizontal prototype displays the user interface for the product and gives a broader view of the entire system, without concentrating on internal functions. A Vertical prototype on the other side is a detailed elaboration of a specific function or a sub system in the product.

The purpose of both horizontal and vertical prototype is different. Horizontal prototypes are used to get more information on the user interface level and the business requirements. It can even be presented in the sales demos to get business in the market. Vertical prototypes are technical in nature and are used to get details of the exact functioning of the sub systems. For example, database requirements, interaction and data processing loads in a given sub system.

**35.Explain Banker's Algorithm for Deadlock handling.**

Banker's Algorithm in Operating System

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue.

**Why Banker's algorithm is named so?**

Banker's algorithm is named so because it is used in banking system to check whether loan can be sanctioned to a person or not. Suppose there are n number of account holders in a bank and the total sum of their money is S. If a person applies for a loan then the bank first subtracts the loan amount from the total money that bank has and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders comes to withdraw their money then the bank can easily do it.

In other words, the bank would never allocate its money in such a way that it can no longer satisfy the needs of all its customers. The bank would try to be in safe state always.

Following **Data structures** are used to implement the Banker's Algorithm:

Let **'n'** be the number of processes in the system and **'m'** be the number of resources types.

**Available :**

- It is a 1-d array of size **'m'** indicating the number of available resources of each type.
- Available[ j ] = k means there are **'k'** instances of resource type $\mathbf{R_j}$

**Max :**

- It is a 2-d array of size '**n*m'** that defines the maximum demand of each process in a system.
- Max[ i, j ] = k means process $\mathbf{P_i}$ may request at most **'k'** instances of resource type $\mathbf{R_j}$.

**Allocation :**

- It is a 2-d array of size **'n*m'** that defines the number of resources of each type currently allocated to each process.
- Allocation[ i, j ] = k means process $\mathbf{P_i}$ is currently allocated **'k'** instances of resource type $\mathbf{R_j}$

**Need :**

- It is a 2-d array of size **'n*m'** that indicates the remaining resource need of each process.
- Need [ i, j ] = k means process $\mathbf{P_i}$ currently need **'k'** instances of resource type $\mathbf{R_j}$

for its execution.

- Need [ i, j ] = Max [ i, j ] – Allocation [ i, j ]

$Allocation_i$ specifies the resources currently allocated to process $P_i$ and $Need_i$ specifies the additional resources that process $P_i$ may still request to complete its task.
Banker's algorithm consists of Safety algorithm and Resource request algorithm

**Safety Algorithm**

The algorithm for finding out whether or not a system is in a safe state can be described as follows:
1) Let Work and Finish be vectors of length 'm' and 'n' respectively.
Initialize: Work = Available
Finish[i] = false; for i=1, 2, 3, 4….n
2) Find an i such that both
a) Finish[i] = false
b) $Need_i$ <= Work
if no such i exists goto step (4)
3) Work = Work + Allocation[i]
Finish[i] = true
goto step (2)
4) if Finish [i] = true for all i
then the system is in a safe state

**Resource-Request Algorithm**

Let $Request_i$ be the request array for process $P_i$. $Request_i$ [j] = k means process $P_i$ wants k instances of resource type $R_j$. When a request for resources is made by process $P_i$, the following actions are taken:
1) If $Request_i$ <= $Need_i$
Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.
2) If $Request_i$ <= Available
Goto step (3); otherwise, $P_i$ must wait, since the resources are not available.
3) Have the system pretend to have allocated the requested resources to process Pi by modifying the state as
follows:

Available = Available – Requesti
$Allocation_i = Allocation_i + Request_i$
$Need_i = Need_i – Request_i$

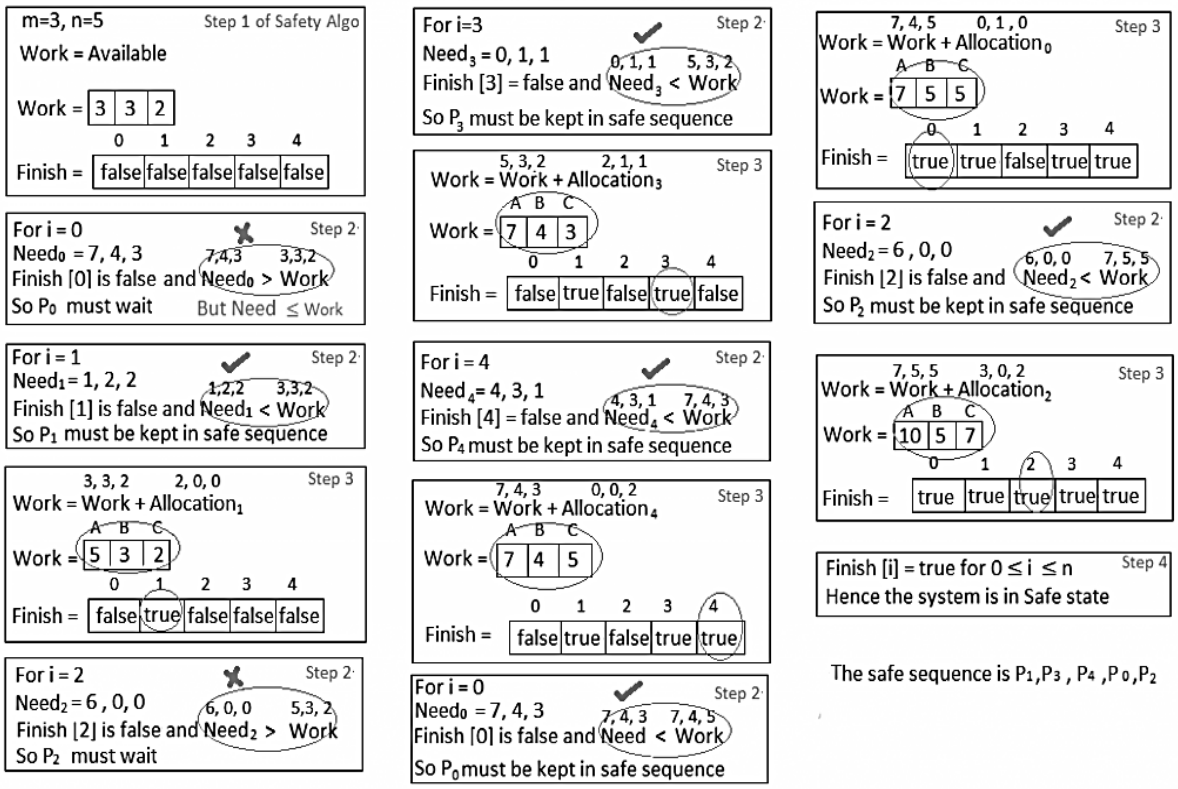
**Example:**
**Considering a system with five processes $P_0$ through $P_4$ and three resources of type A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time $t_0$ following snapshot of the system has been taken:**

| Process | Allocation | Max | Available |
|---|---|---|---|
| | A B C | A B C | A B C |
| $P_0$ | 0 1 0 | 7 5 3 | 3 3 2 |
| $P_1$ | 2 0 0 | 3 2 2 | |
| $P_2$ | 3 0 2 | 9 0 2 | |
| $P_3$ | 2 1 1 | 2 2 2 | |
| $P_4$ | 0 0 2 | 4 3 3 | |

**Question1. What will be the content of the Need matrix?**
Need [i, j] = Max [i, j] – Allocation [i, j]
So, the content of Need Matrix is:

| Process | Need | | |
|---|---|---|---|
| | A | B | C |
| $P_0$ | 7 | 4 | 3 |
| $P_1$ | 1 | 2 | 2 |
| $P_2$ | 6 | 0 | 0 |
| $P_3$ | 0 | 1 | 1 |
| $P_4$ | 4 | 3 | 1 |

**Question2. Is the system in a safe state? If Yes, then what is the safe sequence?**
Applying the Safety algorithm on the given system,

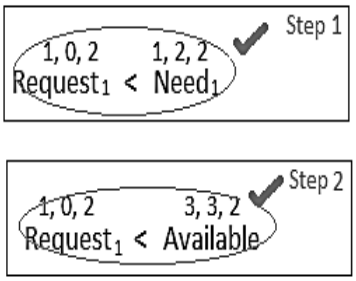



**Question3. What will happen if process $P_1$ requests one additional instance of resource**

A B C

$Request_1$ = 1, 0, 2

To decide whether the request is granted we use Resource Request algorithm




Step 3

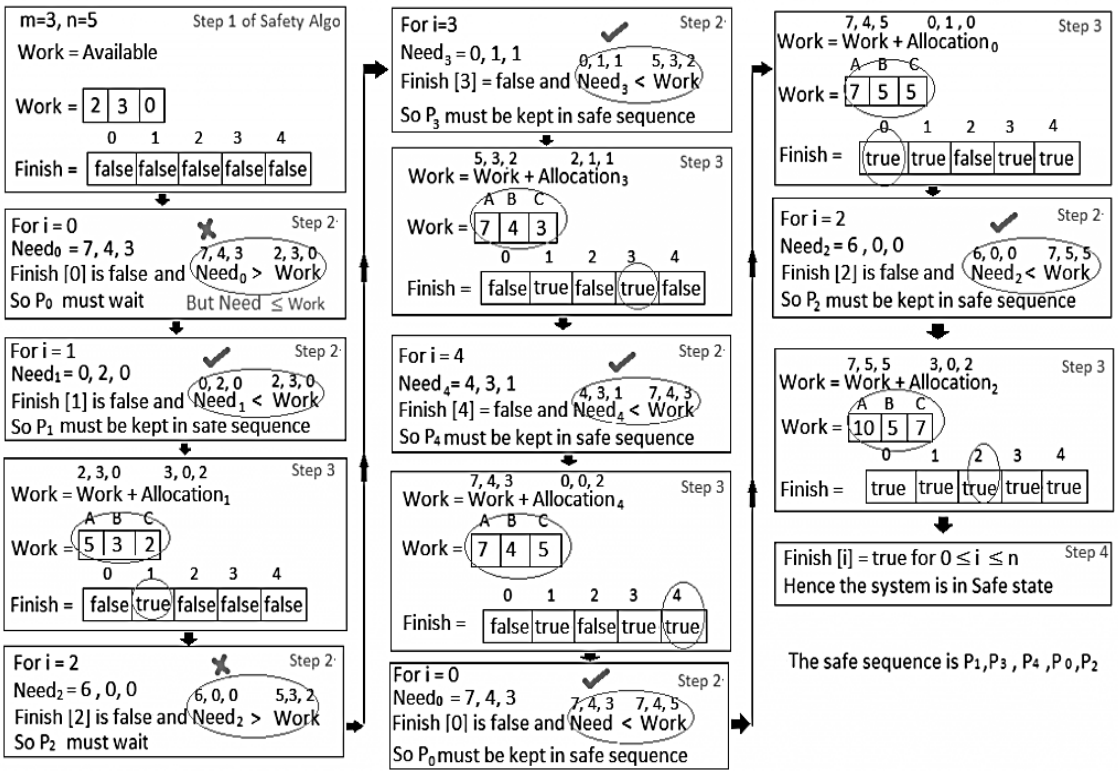
$Available = Available - Request_1$

$Allocation_1 = Allocation_1 + Request_1$

$Need_1 = Need_1 - Request_1$

| Process | Allocation | Need | Available |
|---|---|---|---|
| | A B C | A B C | A B C |
| $P_0$ | 0 1 0 | 7 4 3 | 2 3 0 |
| $P_1$ | 3 0 2 | 0 2 0 | |
| $P_2$ | 3 0 2 | 6 0 0 | |
| $P_3$ | 2 1 1 | 0 1 1 | |
| $P_4$ | 0 0 2 | 4 3 1 | |

**A and two instances of resource type C?**




We must determine whether this new system state is safe. To do so, we again execute Safety algorithm on the above data structures.

Hence the new system state is safe, so we can immediately grant the request for process **$P_1$**

**36.Explain primary key and Foreign key**
**Solution:**
Foreign key: A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables. It acts as a cross-reference between tables because it references the primary key of another table, thereby establishing a link between them.

**37.Explain Swapping & Memory protection system**
**Swapping**. **Swapping** is a mechanism in which a process can be **swapped** temporarily out of main memory (or move) to secondary storage (disk) and make that memory available to other processes. At some later time, the system **swaps** back the process from the secondary storage to main memory.

**Memory protection** is a way to control memory access rights on a computer, and is a part of most modern instruction set architectures and operating systems. The main purpose of memory protection is to prevent a process from accessing memory that has not been allocated to it. This prevents a bug or malware within a process from affecting other processes, or the operating system itself. Protection may encompass all accesses to a specified area of memory, write accesses, or attempts to execute the contents of the area. An attempt to access unauthorized[a] memory results in a hardware fault, e.g., a segmentation fault, storage violation exception, generally causing abnormal termination of

the offending process. Memory protection for computer security includes additional techniques such as address space layout randomization and executable space protection.

**38.Distinguish between paging & Segmentation**

| SL | Key | Paging | Segmentation |
|---|---|---|---|
| **1** | Memory Size | In Paging, a process address space is broken into fixed sized blocks called pages. | In Segmentation, a process address space is broken in varying sized blocks called sections. |
| **2** | Accountability | Operating System divides the memory into pages. | Compiler is responsible to calculate the segment size, the virtual address and actual address. |
| **3** | Size | Page size is determined by available memory. | Section size is determined by the user. |
| **4** | Speed | Paging technique is faster in terms of memory access. | Segmentation is slower than paging. |
| **5** | Fragmentation | Paging can cause internal fragmentation as some pages may go underutilized. | Segmentation can cause external fragmentation as some memory block may not be used at all. |
| **6** | Logical Address | During paging, a logical address is divided into page number and page offset. | During segmentation, a logical address is divided into section number and section offset. |
| **7** | Table | During paging, a logical address is divided into page number and page offset. | During segmentation, a logical address is divided into section number and section offset. |
| **8** | Data Storage | Page table stores the page data. | Segmentation table stores the segmentation data. |

**39.Explain Naïve Bayasian Model with example.**

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit("Yes") or unfit("No") for plaing golf.

Here is a tabular representation of our dataset.

| SL | Outlook | Temperature | Humidity | Windy | Play Golf |
|---|---|---|---|---|---|
| 0 | Rainy | Hot | High | False | No |

| | | | | | |
|---|---|---|---|---|---|
| 1 | Rainy | Hot | High | True | No |
| 2 | Overcast | Hot | High | False | Yes |
| 3 | Sunny | Mild | High | False | Yes |
| 4 | Sunny | Cool | Normal | False | Yes |
| 5 | Sunny | Cool | Normal | True | No |
| 6 | Overcast | Cool | Normal | True | Yes |
| 7 | Rainy | Mild | High | False | No |
| 8 | Rainy | Cool | Normal | False | Yes |
| 9 | Sunny | Mild | Normal | False | Yes |
| 10 | Rainy | Mild | Normal | True | Yes |
| 11 | Overcast | Mild | High | True | Yes |
| 12 | Overcast | Hot | Normal | False | Yes |
| 13 | Sunny | Mild | High | True | No |

The dataset is divided into two parts, namely, feature matrix and the response vector.

- Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of dependent features. In above dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
- Response vector contains the value of class variable(prediction or output) for each row of feature matrix. In above dataset, the class variable name is ‘Play golf’.

Assumption:
The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

contribution to the outcome.

With relation to our dataset, this concept can be understood as:

- We assume that no pair of features are dependent. For example, the temperature being ‘Hot’ has nothing to do with the humidity or the outlook being ‘Rainy’ has no effect on the winds. Hence, the features are assumed to be independent.
- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can’t predict the outcome accuratey. None of the attributes is irrelevant and assumed to be contributing equally to the outcome.

Note: The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Now, before moving to the formula for Naive Bayes, it is important to know about Bayes' theorem.

**Bayes' Theorem**

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation

where A and B are events and P(B) ? 0.

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as evidence.
- P(A) is the priori of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

where, y is class variable and X is a dependent feature vector (of size n) where:

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

X = (Rainy, Hot, High, False)
y = No
So basically, P(y|X) here means, the probability of "Not playing golf" given that the weather conditions are "Rainy outlook", "Temperature is hot", "high humidity" and "no wind".

Naive assumption

Now, its time to put a naive assumption to the Bayes' theorem, which is, independence among the features. So now, we split evidence into the independent parts.
Now, if any two events A and B are independent, then,

P(A,B) = P(A)P(B)
Hence, we reach to the result:

which can be expressed as:

Now, as the denominator remains constant for a given input, we can remove that term:

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability. This can be expressed mathematically as:

So, finally, we are left with the task of calculating P(y) and $P(x_i \mid y)$.
Please note that P(y) is also called class probability and $P(x_i \mid y)$ is called conditional probability.
The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.
Let us try to apply the above formula manually on our weather dataset. For this, we need to do some precomputations on our dataset.

We need to find $P(x_i \mid y_j)$ for each $x_i$ in X and $y_j$ in y. All these calculations have been demonstrated in the tables below:

**Outlook**

| | Yes | No | P(yes) | P(no) |
|---|---|---|---|---|
| Sunny | 2 | 3 | 2/9 | 3/5 |
| Overcast | 4 | 0 | 4/9 | 0/5 |
| Rainy | 3 | 2 | 3/9 | 2/5 |
| **Total** | 9 | 5 | 100% | 100% |

**Temperature**

| | Yes | No | P(yes) | P(no) |
|---|---|---|---|---|
| Hot | 2 | 2 | 2/9 | 2/5 |
| Mild | 4 | 2 | 4/9 | 2/5 |
| Cool | 3 | 1 | 3/9 | 1/5 |
| **Total** | 9 | 5 | 100% | 100% |

**Humidity**

| | Yes | No | P(yes) | P(no) |
|---|---|---|---|---|
| High | 3 | 4 | 3/9 | 4/5 |
| Normal | 6 | 1 | 6/9 | 1/5 |
| **Total** | 9 | 5 | 100% | 100% |

**Wind**

| | Yes | No | P(yes) | P(no) |
|---|---|---|---|---|
| False | 6 | 2 | 6/9 | 2/5 |
| True | 3 | 3 | 3/9 | 3/5 |
| **Total** | 9 | 5 | 100% | 100% |

| Play | | P(Yes)/P(No) |
|---|---|---|
| Yes | 9 | 9/14 |
| No | 5 | 5/14 |
| **Total** | 14 | 100% |

So, in the figure above, we have calculated $P(x_i \mid y_j)$ for each $x_i$ in X and $y_j$ in y manually in the tables 1-4. For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.
Also, we need to find class probabilities (P(y)) which has been calculated in the table 5. For example, P(play golf = Yes) = 9/14.

So now, we are done with our pre-computations and the classifier is ready!

Let us test it on a new set of features (let us call it today):

today = (Sunny, Hot, Normal, False)
So, probability of playing golf is given by:

and probability to not play golf is given by:

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:and Now, since

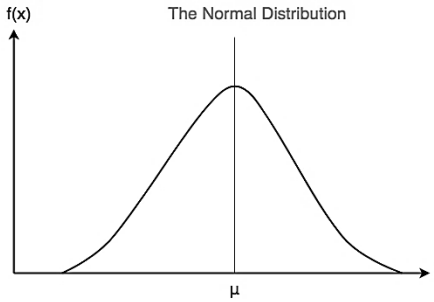
These numbers can be converted into a probability by making the sum equal to 1 (normalization):and Since

So, prediction that golf would be played is ‘Yes’.

The method that we discussed above is applicable for discrete data. In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The

different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.
Now, we discuss one of such classifiers here.

Gaussian Naive Bayes classifier
In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:




The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

Now, we look at an implementation of Gaussian Naive Bayes classifier using scikit-learn.

```
# load the iris dataset
from sklearn.datasets import load_iris
iris = load_iris()

# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target

# splitting X and y into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

# training the model on training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
y_pred = gnb.predict(X_test)
```

```
# comparing actual response values (y_test) with predicted response values (y_pred)
from sklearn import metrics
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test,
y_pred)*100)
```

Output:

Gaussian Naive Bayes model accuracy(in %): 95.0

**40.Describe the procedure of Genetic Algorithm.**

**Five phases are considered in a genetic algorithm:**

Initial population.
Fitness function.
Selection.
Crossover.
Mutation.

**41.What are the DDL and DML?**

Data Definition Language
Data Manipulation Language

**40.What is database normalization? Why 1NF,2NF,3NF and BCNF.**

**Normalization** is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly. Let's discuss about anomalies first then we will discuss normal forms with examples.

There are three types of anomalies that occur when the database is not normalized. These are – Insertion, update and deletion anomaly. Let's take an example to understand this.

To solve this anomalies we need 1NF,2NF,3NF and BCNF.

**42.How hash indexing & B+ Tree indexing?**

A **B+ Tree** is primarily utilized for implementing dynamic indexing on multiple levels. Compared to B- Tree, the B+ Tree stores the data pointers only at the leaf nodes of the Tree, which makes search more process more accurate and faster.

In this B+ Tree tutorial, you will learn:

**Rules for B+ Tree**

Here are essential rules for B+ Tree.

- Leaves are used to store data records.
- It stored in the internal nodes of the Tree.
- If a target key value is less than the internal node, then the point just to its left side is followed.
- If a target key value is greater than or equal to the internal node, then the point just to its right side is followed.

- The root has a minimum of two children.

Hashing There are many possibilities for representing the dictionary and one of the best methods for representing is hashing. Hashing is a type of a solution which can be used in almost all situations. Hashing is a technique which uses less key comparisons and searches the element in O(n) time in the worst case and in an average case it will be done in O(1) time. This method generally used the hash functions to map the keys into a table, which is called a hash table.

**1) Hash table**

Hash table is a type of data structure which is used for storing and accessing data very quickly. Insertion of data in a table is based on a key value. Hence every entry in the hash table is defined with some key. By using this key data can be searched in the hash table by few key comparisons and then searching time is dependent upon the size of the hash table.

**2) Hash function**

Hash function is a function which is applied on a key by which it produces an integer, which can be used as an address of hash table. Hence one can use the same hash function for accessing the data from the hash table. In this the integer returned by the hash function is called hash key.

Types of hash function

There are various types of hash function which are used to place the data in a hash table,

**43.What is SQL? Write down two example of SQL.**

**SQL means Standared Query Language.**

Select * from table name

Select * from table name where id = “2” ;

**45.What is VPN ? How it works?**

**A virtual private network**, or **VPN**, is an encrypted connection over the **Internet** from **a** device to **a network**

Mean Virtual Private Network.VPN is used to access a network securedly.

When we connect with VPN then we connec with VPS server that is located in USA/CANADA . If we type our address then it goes to local ISP then local ISP sends request to access to VPS server .So virtual server give us the access to searched network.Suppose in our country government block facebook.com then we can visit this website using VPN.

i) You can change your device ip easily.
ii) For access blocked website or apps.
iii) Visit for special website
iv) Transfer key securely
v) Keep safe privacy
vi) For surbey flip CPA or affiliate program

**46.What is Digital Signature ? How public key and private key encryption does?**

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software...

**47.What are the differences between FDM and TDM ?**

| SL | Key | | |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **1** | Definition | TDM stands for Time Division Multiplexing. | FDM stands for Frequency Division Multiplexing. |
| **2** | Signal | TDM works well with both analog as well as digital signals. | FDM works only with analog signal. |
| **3** | Conflict | TDM has low conflict. | FDM has high conflict. |
| **4** | Wiring | Wiring or Chip of TDM is simpler. | Wiring or Chip of FDM is complex. |
| **5** | Efficiency | TDM is efficient. | FDM is quiet inefficient |
| **6** | Sharing | Time is shared in TDM | Frequency is shared in FDM |

**48.How DNS does work ?**

EVERY computer has an IP address .DNS is used to domain name into IP address .

When we enter islamicrokomarishop.com to a web browser ..it converts into ip address.

The DNS server with search through its database to find a matching ip address. For that domain name. and when it finds it will resolve that domain name to ip address. To that website.and once that is done then yopur computer is able to communicate with yahoo web server and retriev the webpage .so DNS works basically like a phone book ,when you want to find a number ,you don't lookyou don't look up the name first then it will give the number

**49.Describe DoS and Brute Force attack.**

In computing, a **denial-of-service attack** (**DoS attack**) is a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. Denial of service is typically accomplished by flooding the targeted machine or resource with superfluous requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled.
In a **distributed denial-of-service attack** (**DDoS attack**), the incoming traffic flooding the victim originates from many different sources. This effectively makes it impossible to stop the attack simply by blocking a single source.
A DoS or DDoS attack is analogous to a group of people crowding the entry door of a shop, making it hard for legitimate customers to enter, thus disrupting trade.

Criminal perpetrators of DoS attacks often target sites or services hosted on high-profile web servers such as banks or credit card payment gateway.
Revenge, blackmail and activism can motivate these attacks.

**Brute Force attack**

A brute force attack uses trial-and-error to guess login info, encryption keys, or find a hidden web page. Hackers work through all possible combinations hoping to guess correctly.

These attacks are done by ‘brute force’ meaning they use excessive forceful attempts to try and ‘force’ their way into your private account(s).

This is an old attack method, but it's still effective and popular with hackers. Because depending on the length and complexity of the password, cracking it can take anywhere from a few seconds to many years.