MIT COLLEGE OF ENGINEERING, PUNE

Department of Computer Engineering

LAB MANUAL

Subject

Computer Laboratory - II

Department of Computer Engineering

Lab Manual

Subject Name:

Computer Laboratory II

Subject Code:410447

Class: - B.E. **Branch: - COMPUTER**

Prepared By: - Nagesh Jadhav, Sambhaji Sarode

Year: - Fourth Year-Computer

Contributors: - Nagesh Jadhav, Sambhaji Sarode

Examinations: - Oral and Term Work

Required H/W and S/W: - 64 bit processor based machine and Linux with Java and C++

Institute Vision

To empower young generations for substantial contribution to economical, technological and social progress of the society.

Institute Mission

To be a globally-socially conscious institute of research and innovation with excellence in professional education to take up the challenges of change for benefit of the society.

Vision for Department of Computer Engineering

To empower young generations for substantial contribution in the field of computer and allied domains for the progress of society.

Mission for Department of Computer Engineering

To develop outstanding computer engineering professionals who will lead technology and knowledge based industries in the service of society.

410447: Computer Laboratory-II

PROGRAMME:COMPUTER ENGINEERING	DEGREE:BE YEAR: DEC 2014 – APRIL 2015
COURSE: Computer Laboratory-II	SEMESTER: IV TERMWORK/ORAL/~~PRACTICAL~~ MARKS (ORAL:50, TERMWORK : 50)
COURSE CODE: **410447**	COURSE TYPE:~~CORE~~/~~Elective~~/ Breadth
COURSE AREA/DOMAIN: Machine Intelligence and Computation	CONTACT HOURS: 4 hours/Week
CORRESPONDING LAB COURSE CODE (IF ANY):	LAB COURSE NAME: Computer Laboratory-II

SYLLABUS

Sr. No.	Assignment Name	Outcomes Mapped
	Group A	
1.	Implementation of any 2 uninformed search methods with some application	C707.1
2.	Write a program to perform profile translation-based proactive adaptation using context management in smartphones. Objective of this assignment is to automatically generate user's profile according to the scenarios using machine learning approaches. System should allow to keep user's full profile in user domain resulting into centralizing or exchanging the profile information with increase in the consistency of profile information	C707.1, C707.3
3.	Implement A* approach for any suitable application.	C707.1,
4.	Implementation of Unification algorithm	C707.1,
5.	Implement Naive Bayes to predict the work type for a person with following parameters: age: 30, Qualification: MTech, Experience: 8	C707.1, C707.3

6.	**Elective**-II B. A Pizza shop chain wants to automate dishes served with schemes or without scheme and delivered by the nearest shop in a chain. Use pervasive computing paradigm to develop a web-application using Embedded Java/ Python/ Scala so that the order be delivered to the customer within 10 minutes. Use XML/JSON to store the data.	C707.2, C707.3
Group B		**Group B**
7.	Write a program to build smart mobile app for context management. Objective of this assignment is to build a smart app form smart phone which can sense some parameters of the user and convert this parameters into some contextual information in order to do the context management.	C707.1, C707.3
8.	Write a program to build smart mobile app for user profiling. Objective of this assignment is to develop smart mobile app which can create user profiles based on their preferences so that smart recommendations cab be provided at run time.	C707.1, C707.3
9.	Implementation of MiniMax approach for TIC-TAC-TOE using Java/ Scala/ Python-Eclipse Use GUI Player X and Player O are using their mobiles for the play. Refresh the screen after the move for both the players	C707.1, C707.3
10.	Implementation of a simple NN for any suitable application (without tool)	C707.1, C707.3
11.	Implementation of any 2 uninformed search methods for a LPG company that wants to install a gas pipeline between five cities. The cost of pipeline installation is given in a table maintained using XML/JSON use C++/ Python/ Java/ Scala with Eclipse for the application. Calculate time and space complexities.	C707.1, C707.3
12.	Case study on LIBSVM -A Library for Support Vector Machines. (using LDA for dimensionality reduction).	C707.1, C707.2, C707.3
13.	Developing an book recommender (a book that the reader should read and is new) Expert system or (any other).	C707.1, C707.3
14.	Develop a POP for scheduling your higher studies exam. Assume suitable data like college submission schedule, college exams, Constraint that a paper publication is must to appear before the exam, a family function at home and so on.	C707.1
15.	Implement k-means for clustering data of children belonging to different age groups to perform some specific activities. Formulate the Feature vector for following parameters:	C707.1

	i. height ii. weight iii. age iv. IQ Formulate the data for 40 children to form 3 clusters.	
16.	**Elective**-II B2 In a rolling display program of news display on a smart TV or Computer Display the input strings are supplied by another computer connected through wireless networks. Develop necessary app using Scala/ Python/ Java/ C++. **Elective**-II B3 The BBB (Beagale Bone Black) is used in a Samrt CAR to rotate the steeper motor of a glass window by programmable angle, use model as a HOTSPOT device to transfer the Computer/Internet/Intranet page data of angle of rotation. Write a distributed application using JSON/ xml and Java/ Scala/ Python/ C++.	C707.1, C707.2, C707.3
Group C		**Group C**
17.	Case study- Smart Watch App Development with Tizen. Objective of this assignment is to design simple comic app with the Tizen SDK for Wearable and run it on the smart watch emulator that comes bundled with the IDE.	C707.1, C707.2, C707.3

Text Books :

1. Laboratory Manual generated by the Laboratory Teachers of the respective college, in the Term-work Format; to be assessed and approved by the BoS
2. Content in Digital Library

COURSE OBJECTIVES:

- To develop problem solving abilities for smart devices.
- To develop problem solving abilities of **pervasiveness**, embedded security and NLP.
- To study algorithmic examples in distributed, concurrent and parallel environments

COURSE OUTCOMES:

Sr. NO	DESCRIPTION	PO MAPPING
C707.1	Problem solving abilities for smart devices.	2,3,4,12
C707.2	Problem solving abilities of pervasiveness, embedded security and NLP.	2,3,12
C707.3	To solve problems for multi-core or distributed, concurrent/Parallel environments	2,3,12

COURSE PRE-REQUISITES:

C.CODE	COURSE NAME	DESCRIPTION	SEM
410447	Computer Laboratory-II		VII

GAPS IN THE SYLLABUS - TO MEET INDUSTRY/PROFESSIONAL REQUIREMENTS:

Sr. No.	Gap DESCRIPTION	Proposed Action	PO Mapping
1	Expert System	Prolog and Visual Prolog	3,5

TOPICS BEYOND CONTENT SYLLABUS/ADVANCED TOPICS/DESIGN:

Sr. No.	Content Description	PO Mapping

WEB SOURCE REFERENCES:

1. nptel.ac.in/courses/106106126/
2. http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-825-techniques-in-artificial-intelligence-sma-5504-fall-2002/lecture-notes/

DELIVERY/INSTRUCTIONAL METHODOLOGIES:

✓CHALK & TALK	✓LCD/SMART BOARDS	STUD. ASSIGNMENT	STUD. SEMINARS
✓WEB RESOURCES	ADD-ON COURSES		

ASSESSMENT METHODOLOGIES-DIRECT

✓ASSIGNMENTS	✓STUD. LAB PRACTICES	STUD. SEMINARS	SIMPLE QUESTIONS
TESTS/MODEL EXAMS	MINI/MAJOR PROJECTS	✓UNIV. EXAMINATION	CERTIFICATIONS
ADD-ON COURSES	IN TUTORIAL HOUR	OTHERS	

ASSESSMENT METHODOLOGIES-INDIRECT

√ASSESSMENT OF COURSE OUTCOMES (BY FEEDBACK, ONCE)	ASSESSMENT OF MINI/MAJOR PROJECTS BY EXT. EXPERTS	√STUDENT FEEDBACK ON FACULTY (TWICE)	OTHERS

Prepared By	**Approved By**	**Verified By**
Nagesh Jadhav Sambhaji Sarode	Prachi Joshi (Domain Head)	Prof. R.K.Bedi (Head of the Department)

PROBLEM STATEMENT :

Implementation of any 2 uninformed search methods with some application.

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

Uninformed Search :

An uninformed (a.k.a. blind, brute-force) search algorithm generates the search tree without using any domain specific knowledge. The two basic approaches differ as to whether you check for a goal when a node is *generated* or when it is *expanded*.

Breadth-First Search

Strategy: expand the shallowest unexpanded node. Implementation: The fringe is a FIFO queue.

- is complete (if *b* is finite)
- is optimal if all path costs are the same (because it always finds the shallowest node first)

Depth-First Search

Strategy: expand the deepest unexpanded node. Implementation: The fringe is a LIFO queue (stack)

- is not complete (because of infinite depth and loops)
- is not optimal

Depth-Limited Search

This is just a depth-first search with a cutoff depth. Here is the algorithm (for the test-at-expansion-time case)

```
fringe := [make_node(start_state, null, null)]
reached_limit := false
while fringe is not empty
   n := fringe.pop()
   if n.state is a goal state return n.actionListFromRoot()
   if n.depth == limit
      reached_limit := true
   else
      for each action a applicable to n.state
         fringe.push(make_node(succ(n.state, a), a, n))
```

return reached_limit ? cutoff : failure

- Won’t run forever unless *b* is infinite
- is not complete because a goal may be below the cutoff
- is not optimal

Depth-First Iterative Deepening

Algorithm:

```
for i in 1..infinity
   run DLS to level i
   if found a goal at level i, return it immediately
```

Uniform Cost Search

Strategy: Expand the lowest cost node. Implementation: the fringe is a priority queue: lowest cost node has the highest priority. In order to be optimal, must test at expansion, not generation, time.

Backwards Chaining

Run the search backwards from a goal state to a start state. Obiously this works best when the actions are reversible, and the set of goal states is small.

Bidirectional Search

Run a search forward from the start state and simulatneously. Motivation is that $bd2+bd2$ is much, much less than bd. Works best when the backwards search is feasible. Problem is space complexity: one of the trees has to be kept in memory so we can test membership for a node generated in the other tree.

ALGORITHM

Checking at generation time:

```
if start_state is a goal state return the empty action list
        fringe := [make_node(start_state, null, null)]
                while fringe is not empty
                        n := select and remove some node from the fringe
                        for each action a applicable to n.state
                        s := succ(n.state, a)
                        n' := make_node(s, a, n)
                        if s is a goal state, return n'.actionListFromRoot()
                        add n' to fringe
                return failure
```

Checking at expansion time:

```
fringe := [make_node(start_state, null, null)]
while fringe is not empty
   n := select and remove some node from the fringe
   if n.state is a goal state return n.actionListFromRoot()
   for each action a applicable to n.state
      add make_node(succ(n.state, a), a, n) to fringe
return failure
```

INPUT :

Goal state in (123_45678) format where '_' represents the blank tile .

EXPECTED OUTPUT :

Solution found at particular depth “d”.
Solution not found.

MATHEMATICAL MODEL :

Let S be the solution perspective .

S={s, e, i, o, f, DD, NDD, success, failure}

s = { Initial state of the 8-tile puzzel consisting of all tiles at particular places }

I = Input of the system → { I1, I2 }

where I1 = { Initial position of tiles }
I2 = { Final position of tiles }

o = Output of the system → { O1 , O2 }

where O1 = { Goal state reached }
O2 = { Goal state not reached }

f = Functions used → { f1, f2}

where f1 = { Depth first search }
f2 = { Breadth first search }

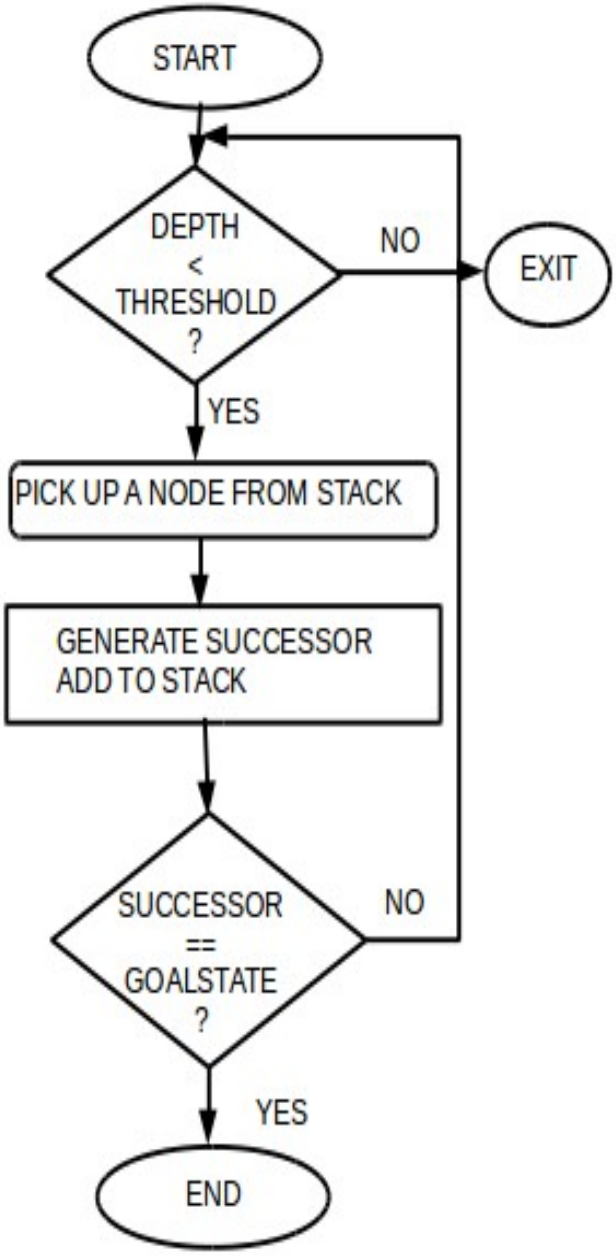
DD - Deterministic data → { Initial position of tiles }

NDD - Non deterministic data → { Number of moves }

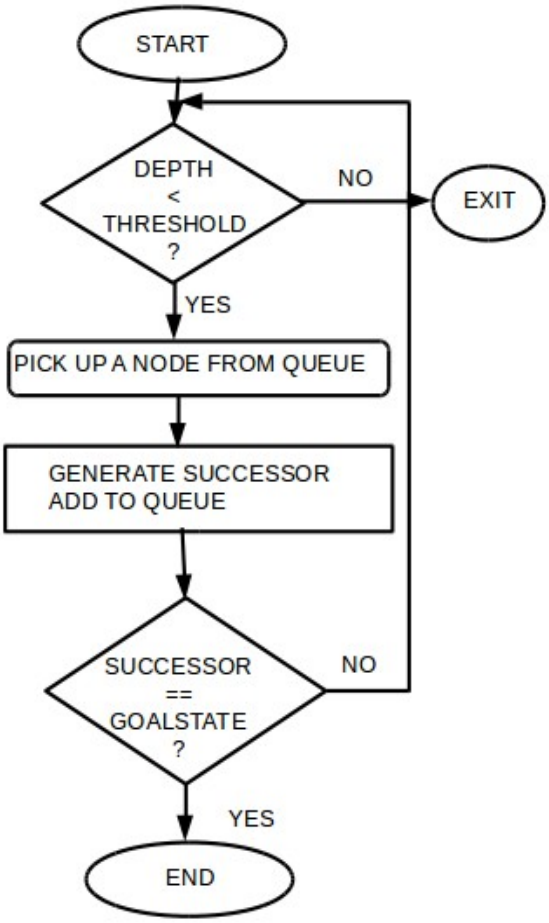
Success - Desired outcome generated → {Goal state reached }

Failure - Desired outcome not generated or forced exit due to system error.

FLOW CHART (DFS):



FLOW CHART (BFS):



TEST CASES : (DFS)

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1.	1_3425678	Solution found at depth 2	Solution found at depth 2	Correct
2.	_13425678	Solution found at depth 3	Solution found at depth 3	Correct
3.	1_3456728	Solution not found	Solution not found	Corrrect

TEST CASES : (BFS)

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1.	1234756_8	Solution Found At Depth 2	Solution Found At Level 2	Correct
2.	123_45678	Solution Found At Depth 2	Solution Found At Level 2	Correct
3.	1_4567823	No Solution	Solution Not Found Depth Exceeded	Corrrect

TIME AND SPACE COMPLEXITY :

Breadth First Search: Time complexity (worst case, goal at rightmost end of level *d*):

Test at generation: $1+b+b2+...+bd=O(bd)$

Space complexity is same as time complexity because every node has to stay in memory.

Depth First Search: Time complexity (worst case: solution is at m): $O(b^m)$ — regardless whether we test at generation or expansion

Space complexity is $O(bm)$ — linear space . Only the nodes on the current path are stored .

CONCLUSION :

Hence the implementation of two uninformed search techniques ie BFS and DFS has been successfully implemented.

OUTCOMES ACHIEVED

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities of pervasiveness,embedded security and NLP.	
To solve problems for multicore or distributed,concurrent/Parallel environments	

FAQs

1. What is blind search?
2. Differentiate between informed search and uninformed search techniques?
3. State and explain problem solving agent.
4. Explain drawback of Uniformed Search.

ASSIGNMENT NO-2

PROBLEM STATEMENT :

Write a program to perform profile translation-based proactive adaptation using context management in smartphones. Objective of this assignment is to automatically generates user's profile according to the scenarios using machine learning approaches. System should allow to keep user's full profile in user domain resulting into centralizing or exchanging the profile information with increase in the consistency of profile information.

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

Mobile technology and Internet is becoming an integral part of our daily life. Various transactions like shopping, ticket booking and banking transactions have been done on the fly. The technology like Smartphone adds portability for these activities. To manage information and applications on Smartphone, user must provide credentials or profiles to service provider with their details filled by logging onto different websites. To this purpose, user's profile resides in control of multiple service providers. Due to this, duplication of data occurs which will leads to a data inconsistency. To overcome these issues, this paper proposes Profile Translation based Proactive Adaption using Context Management (PTPACM) in Smartphones which automatically generates user's profile according to the scenarios. Proposed system allows keeping user's full profile in user domain resulting into centralizing or exchanging the profile information with increase in the consistency of profile information.

Advances in mobile technologies and Internet access make users life very comfortable and convenient to do the work very intelligently. Traditionally Internet uses client-server model. Client requests for some pages and server responds to client or client may give his information to server if required. This is a reactive model. To give more ease at the client end we can use concept of proactivity. Proactive service can be defined as giving response without explicit request. In proactive systems, user is provided with different suggestions according to the different situations. So proactivity means that the system pushes recommendations to the user when current situations seem appropriate .

PTPACM SYSTEM :

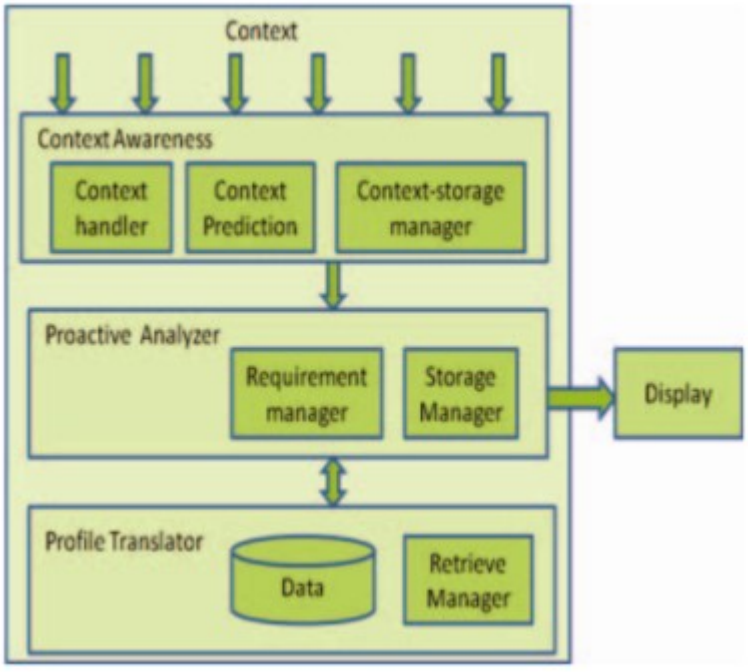
With the increasing use of web apps present in Smartphone it has become tedious job for the users to provide the detailed profile to service provider. In current scenario the user's profile is maintained by service providers. When the particular website requires the information of user, whole profile is provided by the user. But this is actually not relevant because all the information is not needed. Accordingly, only the specified information of the usage should be provided. A system is created to avoid this which is android based and generates specific profile for specific website. As if the website relates to shopping, the profile would

contain only the name, postal address and contact or online reservation will require name, age etc. So according to the requirements, different profiles of user are generated. These will be the abstract views of actual profile of user. And these abstract views are provided to the service provider .

CONTEXT AWARENESS :

To recognize what profile is needed by service provider we use the concept of context awareness. Context in our terms can be referred as all the web apps in a Smartphone also all different kinds of websites that user visits through these Apps. Recognizing which website/webapp or which action of user needs generation of such profiles is referred as context awareness. Being context aware, the system recognizes need of an abstract views of profile which is provided proactively i.e. before user requests. When user tries to open an app which requires profile then this will be recognized and while that site loads, profile will be generated automatically. To generate these views proactively we should know the credentials needed for that website. Generating the profile according to the context determined is the foremost task of the application. The personalization of data is to be done at user side itself. The profile database is to be stored in the Smartphone. This is android based context aware proactive system which manages user's whole profile information and gives abstract view of profile according to the current context requirements .

ARCHITECTURE DIAGRAM :



ANDROID TOOLS USED :

1. android.webkit.WebView : A View that displays web pages. This class is the basis upon which you can roll your own web browser or simply display some online content within your Activity. It uses the WebKit rendering engine to display web pages and includes methods to navigate forward and backward through a history, zoom in and out, perform text.

searches and more.

2. <intent-filter > : Specifies the types of intents that an activity, service, or broadcast receiver can respond to. An intent filter declares the capabilities of its parent component — what an activity or service can do and what types of broadcasts a receiver can handle. It opens the component to receiving intents of the advertised type, while filtering out those that are not meaningful for the component.There are three Intent characteristics you can filter on: the *action*, *data*, and *categories*. For each of these characteristics you can provide multiple possible matching values.

ALGORITHM :

Let, C → **context**
A → **User's Action**

Sense function will sense user's action and will return current context.This context is passed to the **ContextHandler**. ContextHandler will handle this context. Handling of context is done by using stored context information.

1. **C → Sense(A) :** Users action are continously sensed and returned as a context .
2. **ContextHandler (C) :** The context is passed to ContextHandler .
3. All the TextField tags are fetched from the website using a parser .
4. Based on the context the attributes are fetched from the users saved profile .
5. The fetched tags are compared with the attributes fetched from available profile .
6. Matched attributes are automatically filled .

INPUT :

User's profile including Name , Phone no. , Email , Username , Password .
WebPage requiring a signup for further surfing or service access .

EXPECTED OUTPUT :

All the attributes in the saved profile are filled automatically without any human intervention .

MATHEMATICAL MODEL :

Let P be the solution perspective .

P ={ S , E , I , O, F }

S = { Initial state of the system consisting of a interactive form filling for profile generation .
}

I = Input of the system → { I1 , I2 }
where I1 = { User's attributes based on the fields in the form . }
I2 = { A website requiring SignUp for further service access . }

O = Output of the system → { O1 }
where O1 = { Automatic Filling of fields from users Profile }

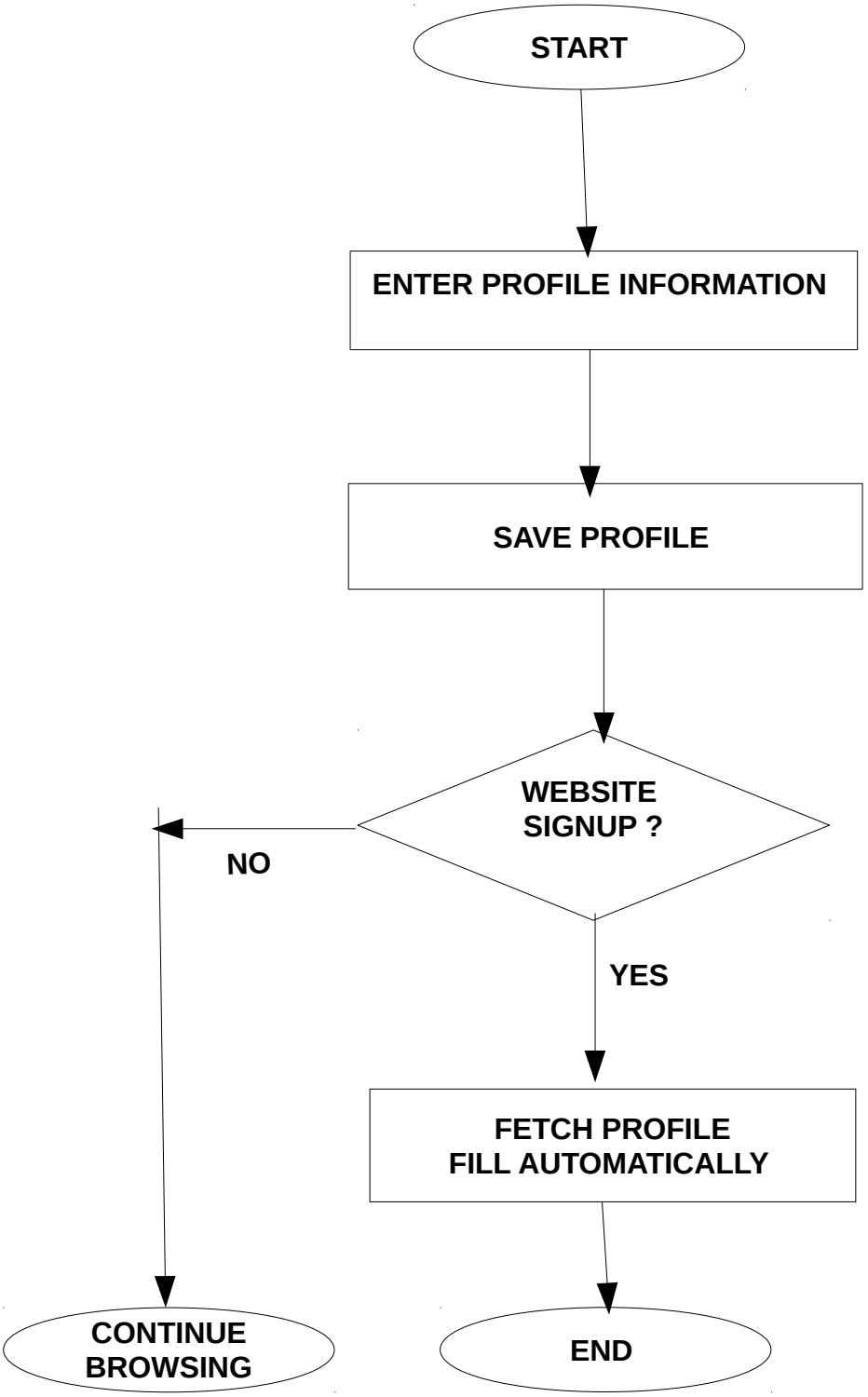
F = Functions used → { f1 , f2 , f3 }
where f1 = { **IntentFilter()** for filtering websites i.e Context Management }
f2 = { **WebView()** for viewing websites in Android Activity }
f3 = { **SharedPreferences()** for saving preferences }

E = End state of the system which fills all the available fields from user's profile into the WebPage .

TEST CASES :

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1	GMAIL SIGNUP	FILL NAME , PHONE AND EMAIL FILLED	NAME , PHONE, EMAIL AND PASSWORD SET	Correct
2	AMAZON SIGNUP	NAME AND EMAIL FILLED	NAME EMAIL FILLED	Correct

FLOWCHART :



CONCLUSION :

Hence we have successfully created an Android Application to read context of the user and providing proactivity in Smartphone, using profile translation architecture.

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	√
To solve problems for multicore or distributed,concurrent/Parallel environments	

FAQs

1. What is profile context management in smart devices?

ASSIGNMENT NO-3

PROBLEM STATEMENT :

Implement A* approach for any suitable application.

OBJECTIVES :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

A* is a computer algorithm that is widely used in pathfinding and graph traversal, the process of plotting an efficiently traversable path between multiple points, called nodes. It is *complete* and will always find a solution if one exists. It evaluates nodes by combining g(n), the cost to reach the node, and h(n), the cost to get from the node to the goal:

$$\mathbf{f(n) = g(n) + h(n) .}$$

Since g(n) gives the path cost from the start node to node n, and h(n) is the estimated cost of the cheapest path from n to the goal, we have
f (n) = estimated cost of the cheapest solution through n .

Thus, if we are trying to find the cheapest solution, a reasonable thing to try first is the node with the lowest value of g(n) + h(n). The algorithm is identical to UNIFORM-COST-SEARCH except that A∗ uses g + h instead of g.

Optimality of A*:

The tree-search version of A ∗ is optimal if h(n) is admissible, while the graph-search version is optimal if h(n) is consistent.

Conditions for optimality: Admissibility and consistency

The first condition we require for optimality is that h(n) be an admissible heuristic. An admissible heuristic is one that never overestimates the cost to reach the goal. Because g(n) is the actual cost to reach n along the current path, and f (n) = g(n) + h(n), we have as an immediate consequence that f (n) never overestimates the true cost of a solution along the current path through n.

A second, slightly stronger condition called consistency (or sometimes monotonicity) is required only for applications of A ∗ to graph search. A heuristic h(n) is consistent if, for every node n and every successor n of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n plus the estimated cost of reaching the goal from n : h(n) ≤ c(n, a, n) + h(n) .

ALGORITHM :

First, we create lists open and closed. open is similar to bfs's queue, and closed is similar to the coloring technique we used before. However, we have to make a list because coloring doesn't store cost information.
So we pop the element with the least f off the open list, and generate its successors. If one of its

successors is the goal, we're golden and can stop the search.
For each successor, calculate its cost and add it to the open list. At the end of this iteration, put the node we popped off the open list on the closed list.

PSEUDO CODE:

```
1:  // A*
2:  initialize the open list
3:  initialize the closed list
-   put the starting node on the open list (you can leave its f at zero)
4:
5:  while the open list is not empty
6:      find the node with the least f on the open list, call it "q"
7:      pop q off the open list
8:      generate q's 8 successors and set their parents to q
9:      for each successor
10:         if successor is the goal, stop the search
11:         successor.g = q.g + distance between successor and q
12:         successor.h = distance from goal to successor
-           successor.f = successor.g + successor.h
13:
-           if a node with the same position as successor is in the OPEN list \
14:             which has a lower f than successor, skip this successor
-           if a node with the same position as successor is in the CLOSED list \
15:             which has a lower f than successor, skip this successor
16:         otherwise, add the node to the open list
17:     end
18:     push q on the closed list
    end
```

INPUT :

Goal State and Initial state.

EXPECTED OUTPUT :

Solution found in “n” steps.

MATHEMATICAL MODEL :

Let S be the solution perspective .
S={s, e, i, o, f, DD, NDD, success, failure}

s = { Initial state of the board consisting of all tiles at particular places }

I = Input of the system → { I1, I2 }

where I1 = { Initial position of tiles }
I2 = { Final position of tiles }

o = Output of the system → { O1 , O2 }

where O1 = { Goal state reached }
O2 = { Goal state not reached }

f = Functions used → { f1}

where f1 = { minimum f(n) ,where f(n) = g(n) + h(n)
where h(n) = no. of unmatched tiles }

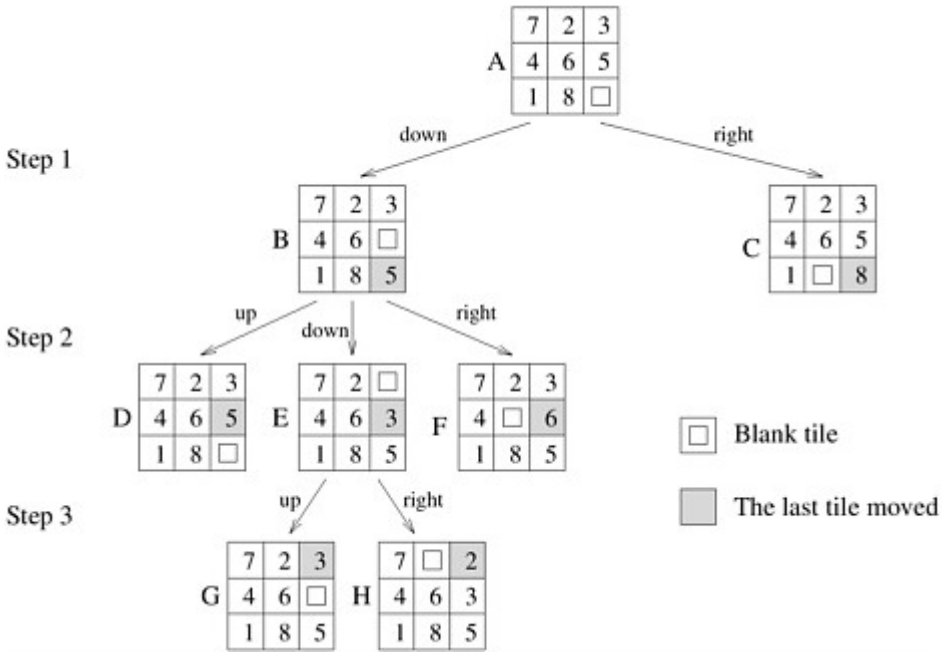
DD - Deterministic data → { Initial position of tiles , heuristic values }

NDD - Non deterministic data → { Number of moves }

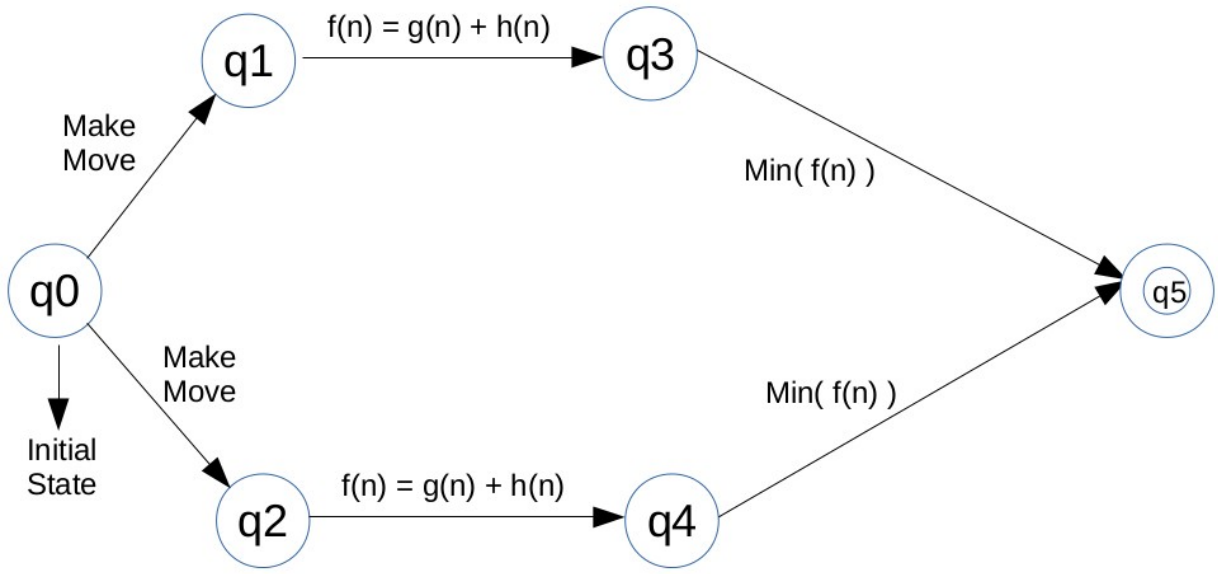
Success - Desired outcome generated → {Goal state reached }

Failure - Desired outcome not generated or forced exit due to system error.

GAME TREE :



STATE DIAGRAM :



TEST-CASES

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1.	1_3428657	Solution found in 9 steps	Solution found in 9 steps	Correct
2	1_3425678	Solution found in 9 steps	Solution found in 9 steps	Correct
3.	1234567890_1342	Invalid input	Wrong input .Enter in correct fashion	Correct

SPACE AND TIME COMPLEXITIES :

The time complexity of A* depends on the heuristic. In the worst case of an unbounded search space, the number of nodes expanded is exponential in the length of the solution (the shortest path) *d*: O(b), where *b* is the branching factor (average number of successors per state).This assumes that a goal state exists at all, and is reachable from the start state; if it is not, and the state space is infinite, the algorithm will not terminate. The time complexity is polynomial when the search space is a tree, there is a single goal state, and the heuristic function *h* meets the following condition:

h(x) – h*(x) = O(logh*(x))

where h^* is the optimal heuristic, the exact cost to get from x to the goal. In other words, the error of h will not grow faster than the logarithm of the "perfect heuristic" h^* that returns the true distance from x to the goal.

CONCLUSION :

Hence we have successfully implemented the A-star algorithm to solve the n-queens problem.

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	
To solve problems for multicore or distributed,concurrent/Parallel environments	

FAQs

1. What is Greedy's best first search? Explain.
2. What is heustic function? What is Manhatton distance?
3. What are drawbacks of A* algorithm?
4. Explain admissible heuristic .
5. What is SMA* and IDA* approach?
6. Explain RBFS algorithm.

ASSIGNMENT NO-4

PROBLEM STATEMENT :

Implementation of Unification algorithm

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

Lified inference rules require finding substitutions that make different logical expressions look identical. This process is called unification and is a key component of all first-order inference algorithms. The UNIFY algorithm takes two sentences and returns a unifier for them if one exists.

A **unification algorithm** should compute for a given problem a complete, and minimal substitution set, that is, a set covering all its solutions, and containing no redundant members.

UNIFY (p q)= 0 where SunsT(0,p)= SuBsT(0, q) .

Let us look at some examples of how UNIFY should behave. Suppose we have a query Askilars(Knows (John , x)): whom does John know? Answers to this query can be found by finding all sentences in the knowledge base that unify with Knows(John, x). Here are the results of unification with four different sentences that might be in the knowledge base:

UNIFY (Knows (John x). Knows (John, Jane)) = {x / Jane}
UNIFY (Knows (John , x). Knows (y, Bill)) = {x / Bill, y / John}
UNIFY (Knows(John , x), Knows (y , Mother (y))) = {y/John ,x / mother(John)}
UNIFY (Knows(John, x). Knows (x , Elizabeth)) = fail.

ALGORITHM :

1: **Procedure** Unify(*t1,t2*)
2: **Inputs**
3: *t1,t2*: atoms **Output**
4: most general unifier of *t1* and *t2* if it exists or ⊥ otherwise
5: **Local**
6: *E*: a set of equality statements
7: *S*: substitution
8: $E \leftarrow \{t1=t2\}$
9: $S=\{\}$
10: **while** ($E\neq\{\}$)
11: select and remove $x=y$ from *E*
12: **if** (*y* is not identical to *x*) **then**
13: **f** (*x* is a variable) **then**
14: replace *x* with *y* everywhere in *E* and *S*
15: $S \leftarrow \{x/y\} \cup S$
16: **else if** (*y* is a variable) **then**
17: replace *y* with *x* everywhere in *E* and *S*
18: $S \leftarrow \{y/x\} \cup S$
19: **else if** (*x* is *f(x1,...,xn)* and *y* is *f(y1,...,yn)*) **then**

```
20:                 E←E∪{x1=y1,...,xn=yn}
21:         else
22:                 return ⊥
23:         return S
```

INPUT :

Values of Expression E1 and E2 to be unified .

EXPECTED OUTPUT :

Uninfication output or Cannot apply unification.

MATHEMATICAL MODEL :

Let S be the solution perspective .

S={s, e, i, o, f, DD, NDD, success, failure}

s = { Initial Expressions E1 and E2 }

I = Input of the system → { I1, I2 }
where I1 = { Element 1 }
I2 = { Element 2 }

o = Output of the system → { O1 , O2 }
where O1 = { Unification output }
O2 = { Cannot apply unification }

f = Functions used → { f1}
where f1 = { Unification algorithm }

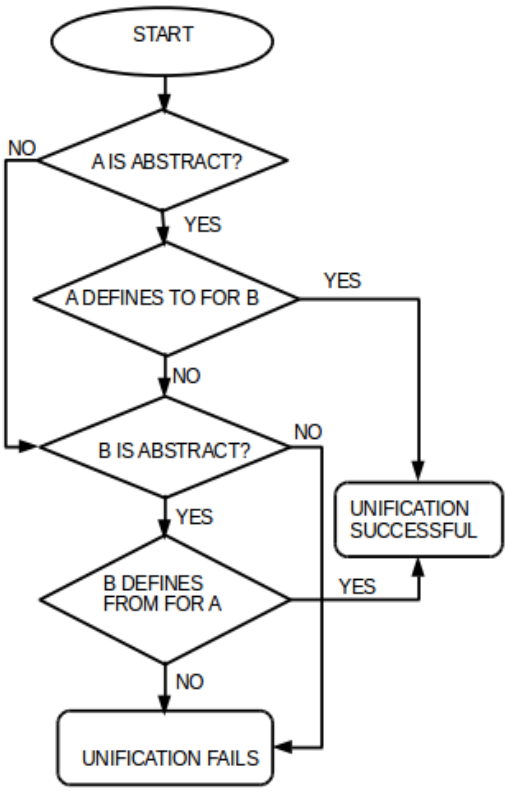
DD - Deterministic data → { }

NDD - Non deterministic data → { Element E1, E2 and Output }

Success - Desired outcome generated → {Unification result }

Failure - Desired outcome not generated or forced exit due to system error.

FLOWCHART :



TEST CASES :

TEST CASE	**INPUT**	**EXPECTED OUTPUT**	**OUTPUT ACHIEVED**	**REMARKS**
1.	E1 f(x,y) E2 g(x,y)	CANNOT APPLY UNIFICATION	PREDICATES NOT MATCHING	Correct
2.	E1 knows(x,y) E2 knows(knows(x), y)	CANNOT APPLY UNIFICATION	CANNOT APPLY UNIFICATION	Correct
3.	E1 knows(john,x) E2 knows(john,jane)	X / jane	jane / x	Correct
4.	E1 knows(john,x) E2 knows(y,bill)	y / john bill / x	y / john bill / x	Correct
5.	E1 knows(john,x) knows(y,mother(y))	y / john mother(john) / x	y / john mother(john) / x	Correc

SPACE AND TIME COMPLEXITIES :

The worst-time complexity of the unification algorithm is EXPTIME. Let N = N(D). There are at most O ((2^ N)^2) cases. Hence complexity of unification is O (2^N) ∈ EXPTIME.

CONCLUSION :

Hence we have successfully implemented the unification algorithm.

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	
To solve problems for multicore or distributed,concurrent/Parallel environments	

FAQs

1. What is logical agent? Explain.
2. What is resolution ?
3. Explain resolution in prospostional logic and predicate logic.
4. Explain steps of converting FOL into CNF.
5. What is ontology and what is knowledge Engineering?

ASSIGNMENT NO-5

PROBLEM STATEMENT :

Implement Naive Bayes to predict the work type for a person with following parameters: age: 30,Qualification: MTech, Experience: 8
Following table provides the details of the available data:

Work Type	Age	Qualification	Experience
Consultancy	30	Ph.D.	9
Service	21	MTech.	1
Research	26	MTech.	2
Service	28	BTech.	10
Consultancy	40	MTech.	14
Research	35	Ph.D.	10
Research	27	BTech.	6
Service	32	MTech.	9
Consultancy	45	Btech.	17
Research	36	Ph.D.	7

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

The naïve Bayesian classifier, or simple Bayesian classifier, works as follows:

1. Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X = (x 1 , x 2 , . . . , x n), depicting n measurements made on the tuple from n attributes, respectively, A 1 , A 2 , . . . , A n .

2. Suppose that there are m classes, C 1 , C 2 , . . . , C m . Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the naïve Bayesian classifier predicts that tuple X belongs to the class C i if and only if

$$P(C_i|X) > P(C_j|X) \text{ for } 1 \leq j \leq m, j = i.$$

Thus we maximize P(C i |X). The class C i for which P(C i |X) is maximized is called the maximum posteriori hypothesis. By Bayes' theorem,

$$P(C_i|X) = [P(X|C_i)P(C_i)]/P(X)$$

As P(X) is constant for all classes, only P(X|C i)P(C i) need be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is,

P(C 1) = P(C 2) = · · · = P(C m), and we would therefore maximize P(X|C i). Otherwise, we maximize P(X|C i)P(C i). Note that the class prior probabilities may be estimated by P(C i) = |C i , D |/|D|, where |C i , D | is the number of training tuples of class C i in D.

Given data sets with many attributes, it would be extremely computationally expensive to compute P(X|C i). In order to reduce computation in evaluating P(X|C i), the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

$$\mathbf{P(X|C\,i) = \prod_{k=1}^{n} P(x\,k|C\,i)}$$
$$\mathbf{= P(x\,1|C\,i) \times P(x\,2|C\,i) \times \cdots \times P(x\,n|C\,i).}$$

We can easily estimate the probabilities P(x 1 |C i), P(x 2 |C i), . . . , P(x n |C i) from the training tuples. Recall that here x k refers to the value of attribute A k for tuple X. For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute P(X|C i), we consider the following:

(a) If A k is categorical, then P(x k |C i) is the number of tuples of class C i in D having the value x k for A k , divided by |C i , D |, the number of tuples of class C i in D.

(b) If A k is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined by

so that,

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$
$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}).$$

These equations may appear daunting, but hold on! We need to compute μ C i and σ C i , which are the mean (i.e., average) and standard deviation, respectively, of the values of attribute A k for training tuples of class C i . We then plug these two quantities into Equation (6.13), together with x k , in order to estimate P(x k |C i).

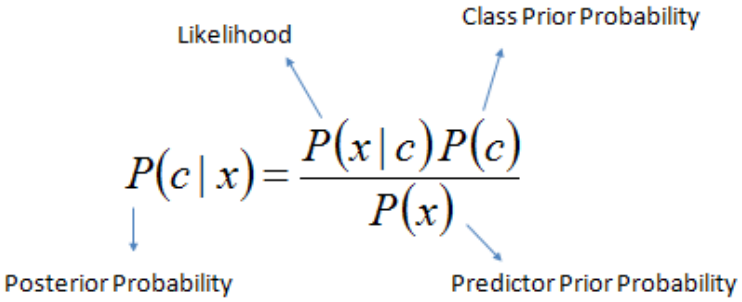
In order to predict the class label of X, P(X|C i)P(C i) is evaluated for each class C i . The classifier predicts that the class label of tuple X is the class C i if and only if

P(X|C i)P(C i) > P(X|C j)P(C j) for 1 ≤ j ≤ m, j = i.

In other words, the predicted class label is the class C i for which P(X|C i)P(C i) is the maximum.

ALGORITHM :

Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- $P(c|x)$ is the posterior probability of *class* (*target*) given *predictor* (*attribute*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*

INPUT :
Enter age , Qualification and Experience

EXPECTED OUTPUT :

Predicted worktype

MATHMODEL

D : Set of tuples
- Each Tuple is an 'n' dimensional attribute vector
- X : (x1,x2,x3,.... xn)

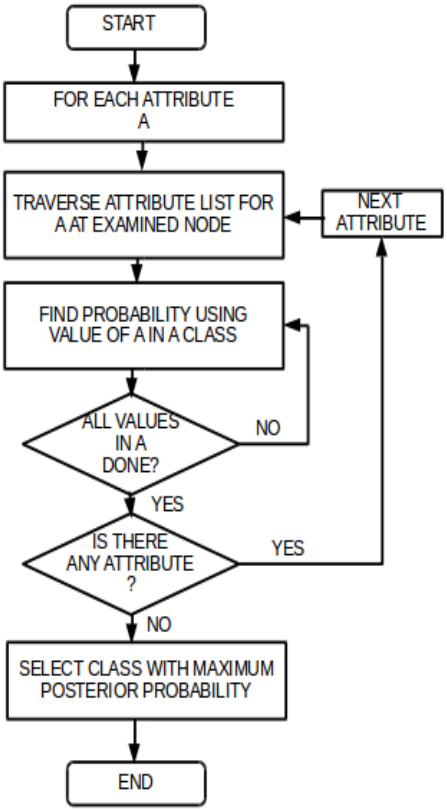
Let there be 'm' Classes : C1,C2,C3...Cm
Naïve Bayes classifier predicts X belongs to Class Ci iff
- P (Ci/X) > P(Cj/X) for 1<= j <= m , j <> i

Maximum Posteriori Hypothesis
- P(Ci/X) = P(X/Ci) P(Ci) / P(X)
- Maximize P(X/Ci) P(Ci) as P(X) is constant

With many attributes, it is computationally expensive to evaluate P(X/Ci).
Naïve Assumption of "class conditional independence"

- $P (X / . Ci) = \prod_{k=1}^{n} P (x k / Ci)$
- P(X/Ci) = P(x1/Ci) * P(x2/Ci) *...* P(xn/ Ci)

FLOWCHART :



TEST CASES :

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1.	ENTER AGE :30 ENTER QUALIFICATION :M.Tech ENTER EXPERIENCE :8	Research	Research	Correct
2.	ENTER AGE :40 ENTER QUALIFICATION :M.Tech ENTER EXPERIENCE :15	Consultancy	Consultancy	Correct
3.	ENTER AGE :40 ENTER QUALIFICATION :P.hD ENTER EXPERIENCE :8	Research	Research	Correct

SPACE AND TIME COMPLEXITIES :

In Baye's classification we use three nested loops which iterate over the entire length of the dataset. Hence time complexity is $O(N^3)$.

CONCLUSION :

Hence we have successfully implemented the naive bayes algorithm to predict the work type of the person.

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	
To solve problems for multicore or distributed,concurrent/Parallel environments	√

FAQs

1. What is prior and posterior probability?
2. What is Bayes theorem?
3. What is classification? How it is different from clustering?
4. What is belief network?

ASSIGNMENT NO-6

PROBLEM STATEMENT :

A Pizza shop chain wants to automate dishes served with schemes or without scheme and delivered by the nearest shop in a chain. Use pervasive computing paradime to develop a web-application using Embedded Java/ Pythone/ Scala so that the order be delivered to the customer within 10 minutes. Use XML/JSON to store the data.

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

Web Application :

In computing, a web application or web app is a client-server software application in which the client (or user interface) runs in a web browser. Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions, wikis and many other functions.

Web Applications are usually broken into logical chunks called "tiers", where every tier is assigned a role. Traditional applications consist only of 1 tier, which resides on the client machine, but web applications lend themselves to an n-tiered approach by nature. Though many variations are possible, the most common structure is the three-tiered application. In its most common form, the three tiers are called *presentation*, *application* and *storage*, in this order. A web browser is the first tier (presentation), an engine using some dynamic Web content technology (such as ASP, CGI, ColdFusion, Dart, JSP/Java, Node.js, PHP, Python or Ruby on Rails) is the middle tier (application logic), and a database is the third tier (storage).The web browser sends requests to the middle tier, which services them by making queries and updates against the database and generates a user interface.

JSP (JavaServer Pages) And Servelet :

JavaServer Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications.Java Server Page (JSP) is a technology for controlling the content or appearance of Web pages through the use of servlets, small programs that are specified in the Web page and run on the Web server to modify the Web page before it is sent to the user who requested it.

Generating dynamic content in Web applications is important when the content must reflect the most current and available data and personalized information. One of the main advantages of JavaServer Pages is the ability to generate dynamic content .

JSP DIRECTIVES :

JSP directives provide directions and instructions to the container, telling it how to handle certain aspects of JSP processing. A JSP directive affects the overall structure of the servlet class. It usually has the following form :

<%@ directive attribute="value" %>

Directives can have a number of attributes which you can list down as key-value pairs and separated by commas.The blanks between the @ symbol and the directive name, and between the last attribute and the closing %>, are optional.

Directive	Description
<%@ page ... %>	Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
<%@ include ... %>	Includes a file during the translation phase.
<%@ taglib ... %>	Declares a tag library, containing custom actions, used in the page

page Directive:

The **page** directive is used to provide instructions to the container that pertain to the current JSP page. You may code page directives anywhere in your JSP page. By convention, page directives are coded at the top of the JSP page.Following is the basic syntax of page directive:

<%@ page attribute="value" %>

include Directive:

The **include** directive is used to includes a file during the translation phase. This directive tells the container to merge the content of other external files with the current JSP during the translation phase. You may code *include* directives anywhere in your JSP page.The general usage form of this directive is as follows:

<%@ include file="relative url" >

taglib Directive:

The JavaServer Pages API allows you to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behavior.The **taglib** directive declares that your JSP page uses a set of custom tags, identifies the location

of the library, and provides a means for identifying the custom tags in your JSP page.The taglib directive follows the following syntax:

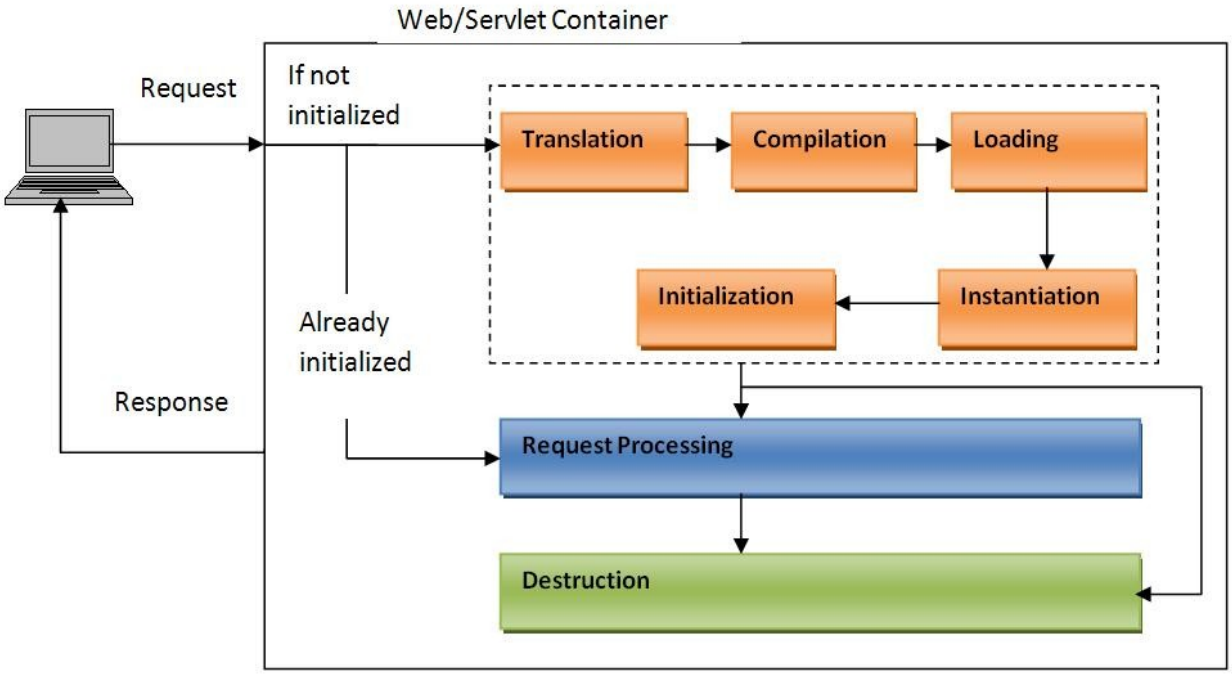
<%@ taglib uri="uri" prefix="prefixOfTag" >

JAVA SERVLET :

Servlets are programs that run on a Web or application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.Many client requests can be satisfied by prebuilt documents, and the server would handle these requests without invoking servlets. In many cases, however, a static result is not sufficient, and a page needs to be generated for each request.

A servlet is a Java programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For such applications, Java Servlet technology defines HTTP-specific servlet classes.The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets. All servlets must implement the Servlet interface, which defines life-cycle methods.

LIFE CYCLE OF A SERVLET :



1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

3) init method is invoked

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet.The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.

4) service method is invoked

The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once.

5) destroy method is invoked

The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc.This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.After the destroy() method is called, the servlet object is marked for garbage collection.

ALGORITHM :

1. Put all the pizza shop in the XML file .
2. Open Pizza order WebPage invoking the WebApp .
3. Enter city and corresponding area .
4. Check whether the pizza shop can deliver the order .
5. Depending upon the availability give appropriate response .

INPUT :

Enter area and city for ordering pizza .

EXPECTED OUTPUT :

Appropriate response whether pizza can be delivered within ten minutes or not .

MATHEMATICAL MODEL :

Let P be the solution perspective .

P ={ S , E , I , O, F }

S = { Initial state of the Webapp showing pizza order WebPage (UserInterface). }

I = Input of the system → { I1 , I2 }
where I1 = { Enter city }
I2 = { Enter area }

O = Output of the system → { O1 }
where O1 = { Appropriate response whether pizza can be delivered or not }

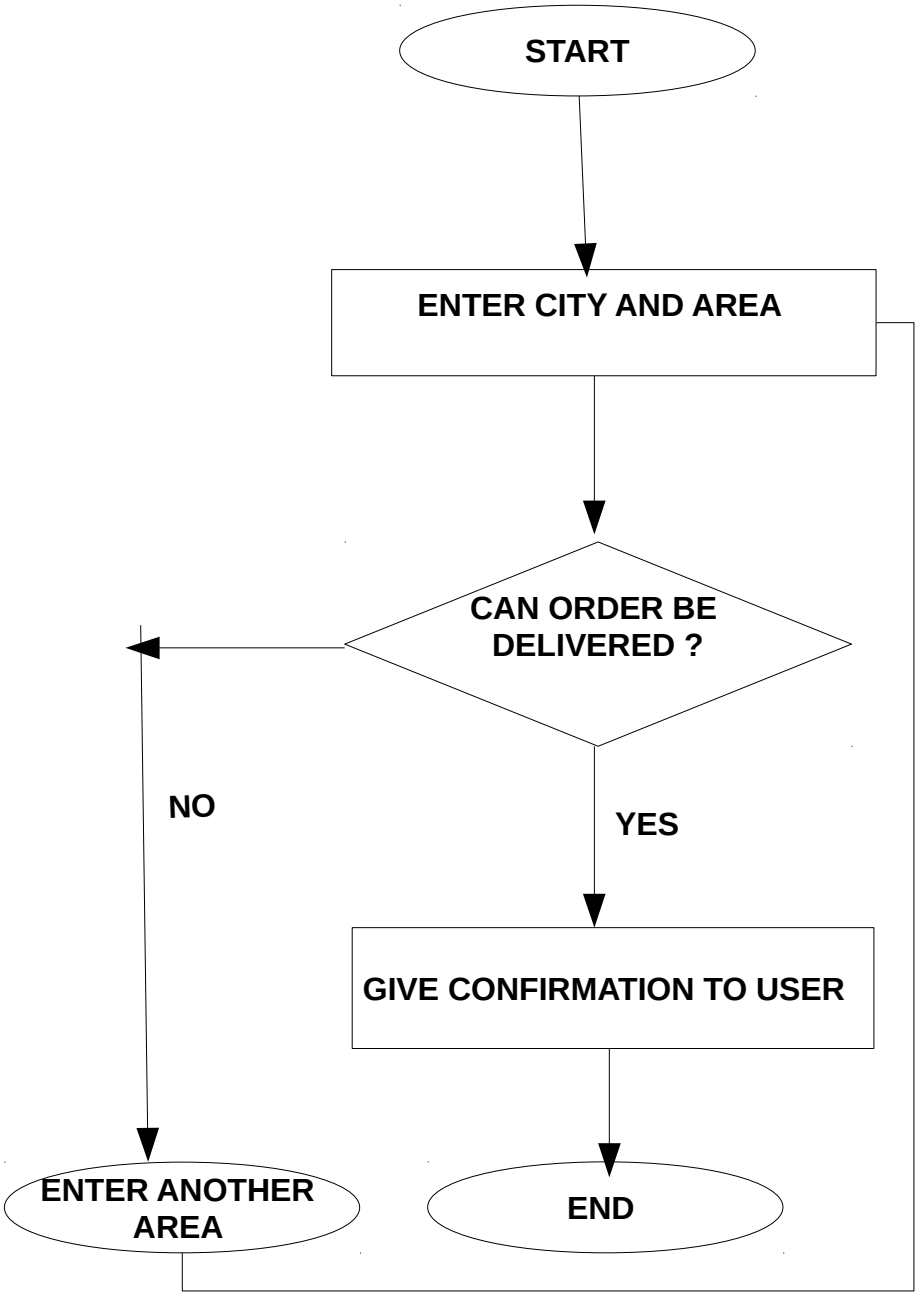
F = Functions used → { f1 , f2 , f3 }
where f1 = { Extracting pizza shop information from Xml File }
f2 = { Input area and city }
f3 = { Calculating whether pizza can be delivered within time limit }

E = End state of the system which tells the user whether his/her pizza order can be delivered to his area within ten minutes .

TEST CASES :

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1	CITY , AREA	ORDER DELIVERABLE	ORDER WILL BE DELIVERED WITHIN 10 MINUTES	Correct
2	CITY , AREA	ORDER CANNOT BE DELIVERED	ORDER CANNOT BE DELIVERED TO YPUR AREA WITHIN TEN MINUTES	Correct

FLOWCHART :



CONCLUSION :

Hence we have successfully implemented a JSP Servelet based WebApp which delivers pizza to the user from the nearest shop available .

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	√
To solve problems for multicore or distributed,concurrent/Parallel environments	√

FAQs

1. What is pervasive computing?
2. Explain differnt paradigm in pervasive computing

ASSIGNMENT NO- B1, B2

PROBLEM STATEMENT :

Write a program to perform profile translation-based proactive adaptation using context management in smartphones. Objective of this assignment is to automatically generates user's profile according to the scenarios using machine learning approaches. System should allow to keep user's full profile in user domain resulting into centralizing or exchanging the profile information with increase in the consistency of profile information.

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

Mobile technology and Internet is becoming an integral part of our daily life. Various transactions like shopping, ticket booking and banking transactions have been done on the fly. The technology like Smartphone adds portability for these activities. To manage information and applications on Smartphone, user must provide credentials or profiles to service provider with their details filled by logging onto different websites. To this purpose, user's profile resides in control of multiple service providers. Due to this, duplication of data occurs which will leads to a data inconsistency. To overcome these issues, this paper proposes Profile Translation based Proactive Adaption using Context Management (PTPACM) in Smartphones which automatically generates user's profile according to the scenarios. Proposed system allows keeping user's full profile in user domain resulting into centralizing or exchanging the profile information with increase in the consistency of profile information.

Advances in mobile technologies and Internet access make users life very comfortable and convenient to do the work very intelligently. Traditionally Internet uses client-server model. Client requests for some pages and server responds to client or client may give his information to server if required. This is a reactive model. To give more ease at the client end we can use concept of proactivity. Proactive service can be defined as giving response without explicit request. In proactive systems, user is provided with different suggestions according to the different situations. So proactivity means that the system pushes recommendations to the user when current situations seem appropriate .

PTPACM SYSTEM :

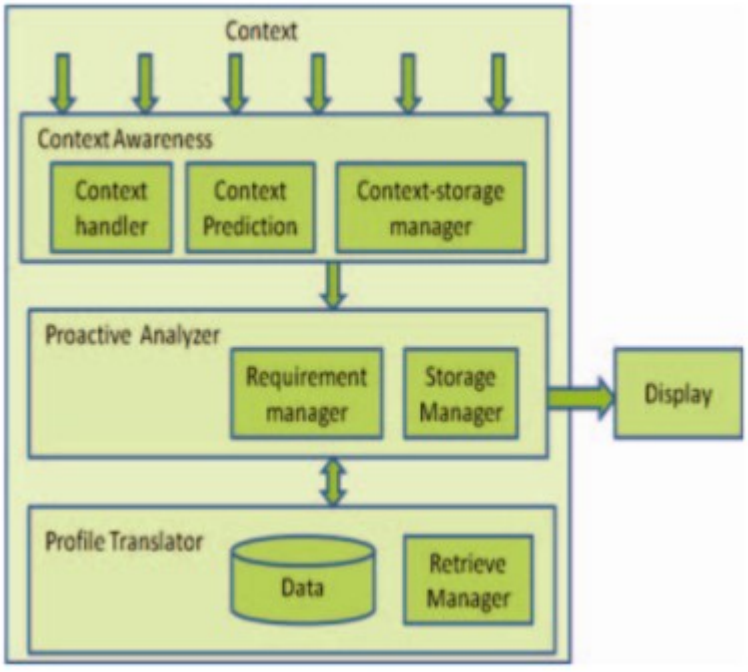
With the increasing use of web apps present in Smartphone it has become tedious job for the users to provide the detailed profile to service provider. In current scenario the user's profile is maintained by service providers. When the particular website requires the information of user, whole profile is provided by the user. But this is actually not relevant because all the information is not needed. Accordingly, only the specified information of the usage should be provided. A system is created to avoid this which is android based and generates specific profile for specific website. As if the website relates to shopping, the profile would

contain only the name, postal address and contact or online reservation will require name, age etc. So according to the requirements, different profiles of user are generated. These will be the abstract views of actual profile of user. And these abstract views are provided to the service provider .

CONTEXT AWARENESS :

To recognize what profile is needed by service provider we use the concept of context awareness. Context in our terms can be referred as all the web apps in a Smartphone also all different kinds of websites that user visits through these Apps. Recognizing which website/webapp or which action of user needs generation of such profiles is referred as context awareness. Being context aware, the system recognizes need of an abstract views of profile which is provided proactively i.e. before user requests. When user tries to open an app which requires profile then this will be recognized and while that site loads, profile will be generated automatically. To generate these views proactively we should know the credentials needed for that website. Generating the profile according to the context determined is the foremost task of the application. The personalization of data is to be done at user side itself. The profile database is to be stored in the Smartphone. This is android based context aware proactive system which manages user's whole profile information and gives abstract view of profile according to the current context requirements .

ARCHITECTURE DIAGRAM :



ANDROID TOOLS USED :

1. android.webkit.WebView : A View that displays web pages. This class is the basis upon which you can roll your own web browser or simply display some online content within your Activity. It uses the WebKit rendering engine to display web pages and includes methods to navigate forward and backward through a history, zoom in and out, perform text.

searches and more.

2. <intent-filter > : Specifies the types of intents that an activity, service, or broadcast receiver can respond to. An intent filter declares the capabilities of its parent component — what an activity or service can do and what types of broadcasts a receiver can handle. It opens the component to receiving intents of the advertised type, while filtering out those that are not meaningful for the component.There are three Intent characteristics you can filter on: the *action*, *data*, and *categories*. For each of these characteristics you can provide multiple possible matching values.

ALGORITHM :

Let, C → **context**
A → **User's Action**

Sense function will sense user's action and will return current context.This context is passed to the **ContextHandler**. ContextHandler will handle this context. Handling of context is done by using stored context information.

1. **C → Sense(A) :** Users action are continously sensed and returned as a context .
2. **ContextHandler (C) :** The context is passed to ContextHandler .
3. All the TextField tags are fetched from the website using a parser .
4. Based on the context the attributes are fetched from the users saved profile .
5. The fetched tags are compared with the attributes fetched from available profile .
6. Matched attributes are automatically filled .

INPUT :

User's profile including Name , Phone no. , Email , Username , Password .
WebPage requiring a signup for further surfing or service access .

EXPECTED OUTPUT :

All the attributes in the saved profile are filled automatically without any human intervention .

MATHEMATICAL MODEL :

Let P be the solution perspective .

P ={ S , E , I , O, F }

S = { Initial state of the system consisting of a interactive form filling for profile generation .
}

I = Input of the system → { I1 , I2 }
where I1 = { User's attributes based on the fields in the form . }
I2 = { A website requiring SignUp for further service access . }

O = Output of the system → { O1 }
where O1 = { Automatic Filling of fields from users Profile }

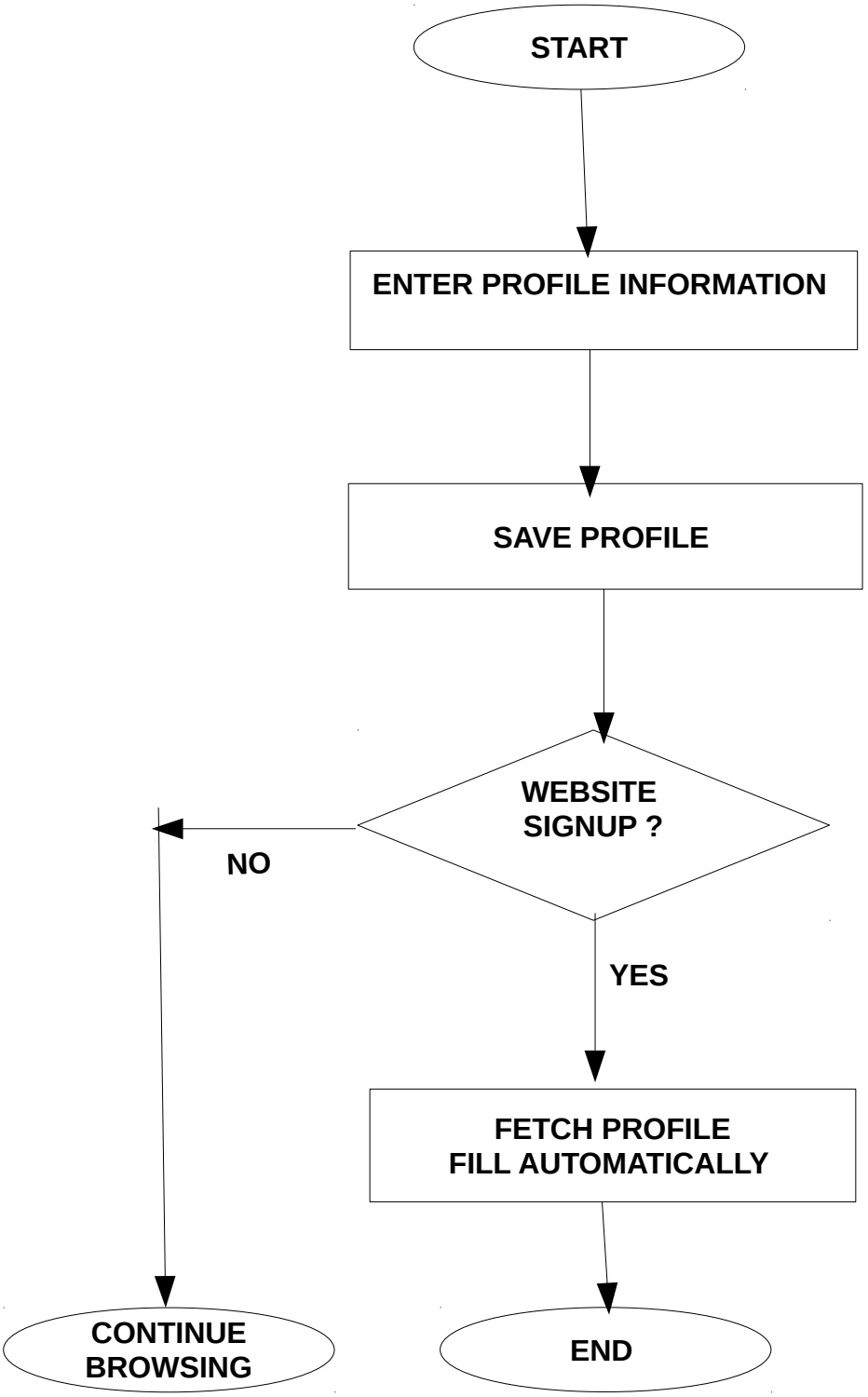
F = Functions used → { f1 , f2 , f3 }
where f1 = { **IntentFilter()** for filtering websites i.e Context Management }
f2 = { **WebView()** for viewing websites in Android Activity }
f3 = { **SharedPreferences()** for saving preferences }

E = End state of the system which fills all the available fields from user's profile into the WebPage .

TEST CASES :

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1	GMAIL SIGNUP	FILL NAME , PHONE AND EMAIL FILLED	NAME , PHONE, EMAIL AND PASSWORD SET	Correct
2	AMAZON SIGNUP	NAME AND EMAIL FILLED	NAME EMAIL FILLED	Correct

FLOWCHART :



CONCLUSION :

Hence we have successfully created an Android Application to read context of the user and providing proactivity in Smartphone, using profile translation architecture.

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	√
To solve problems for multicore or distributed,concurrent/Parallel environments	

ASSIGNMENT NO-B3

PROBLEM STATEMENT :

Implementation of MiniMax approach for TIC-TAC-TOE using Java/ Scala/ Python-Eclipse Use GUI. Player X and Player O are using their mobiles for the play. Refresh the screen after the move for both the players.

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

Given a game tree, the optimal strategy can be determined from the minimax value of each node, which we write as MiNimAx(n). The minimax value of a node is the utility (for MAX) of being in the corresponding state, assuming that both players play optimally from there to the end of the game. Obviously, the minimax value of a terminal state is just its utility.

Furthermore, given a choice, MAX prefers to move to a state of maximum value, whereas MIN prefers a state of minimum value. So we have the following:

MINIMAX(s) =

{ UTILITY (S) if TERMINAL-TEST(s) }
{ $\max_a \varepsilon$ Actions(s) MINIMAX(RESULT(s,a)) if PLAYER(S)-----MAX }
{ $\max_a \varepsilon$ Actions(s) MINIMAX(RESULT(s,a)) if PLAYER(S)-----MAX }

The minimax algorithm computes the minimax decision from the current state. It uses a simple recursive computation of the minimax values of each successor state, directly implementing the defining equations. The recursion proceeds all the way down to the leaves of the tree, and then the minimax values are backed up through the tree as the recursion unwinds.

For two player games, the minimax algorithm is such a tactic, which uses the fact that the two players are working towards opposite goals to make predictions about which future states will be reached as the game progresses, and then proceeds accordingly to optimize its chance of victory. The theory behind minimax is that the algorithm's opponent will be trying to minimize whatever value the algorithm is trying to maximize (hence, "minimax"). Thus, the computer should make the move which leaves its opponent capable of doing the least damage.

In the ideal case (i.e., the computer has infinite time and infinite storage capacity), the computer will investigate every game outcome possible from the game's current state (as well as the paths taken to reach those states) and assign those outcomes values. For a win-or-lose game like chess or tic-tac-toe, there are only three possible values-win, lose, or draw (often assigned numeric values of 1, -1, and 0 respectively)--but for other games like backgammon or poker with a "score" to maximize, the scores themselves are used. Then, starting from the bottom, the computer evaluates which possible outcome is best for the computer's opponent. It then assumes that, if that game stage is reached, its opponent will make the move which leads to the outcome best for the opponent (and

worse for the computer). Thus, it can "know" what its opponent will do, and have a concrete idea of what the final game state will be if that second-to-last position is in fact reached. Then, the computer can treat that position of a terminal node of that value, even though it is not actually a terminal value. This process can then be repeated a level higher, and so on. Ultimately, each option that the computer currently has available can be a assigned a value, as if it were a terminal state, and the computer simply picks the highest value and takes that action.

Describing a Perfect Game of Tic Tac Toe

To begin, let's start by defining what it means to play a perfect game of tic tac toe:

If I play perfectly, every time I play I will either win the game, or I will draw the game. Furthermore if I play against another perfect player, I will always draw the game.

How might we describe these situations quantitatively? Let's assign a score to the "end game conditions:"

- Player wins.
- Player loses.
- The game is a draw.

A description for the algorithm, assuming X is the "turn taking player," would look something like:

- If the game is over, return the score from X's perspective.
- Otherwise get a list of new game states for every possible move
- Create a scores list
- For each of these states add the minimax result of that state to the scores list
- If it's X's turn, return the maximum score from the scores list
- If it's O's turn, return the minimum score from the scores list

ALGORITHM :

```
function MINIMAX (N) :
begin
if N is deep enough then
        return the estimated score of this leaf
else
        Let N1,N2,..... Nm  be the successors of N;
        if N is a Min node then :
                return min{MINIMAX(N1),.....MINIMAX(Nm)}
        else
                return min{MINIMAX(N1),.....MINIMAX(Nm)}
end MINIMAX;
```

INPUT :

Input position on the tic-tac-toe board.

EXPECTED OUTPUT :

Human wins Or Computer wins Or Draw .

MATHEMATICAL MODEL :

Let S be the solution perspective .

S={s, e, i, o, f, DD, NDD, success, failure}

s = { Initial state of TIC-TAC-TOE containing blank in all tiles }

I = Input of the system → { I1}
where I1 = { Position on the tic-tac-toe board }

o = Output of the system → { O1 , O2 . O3}
where O1 = { Human wins}
O2 = { Computer wins }
O3 = { Game tie }

f = Functions used → { f1}
where f1 = { Minimax algorithm }
f2 = { Tkinter algorithm }

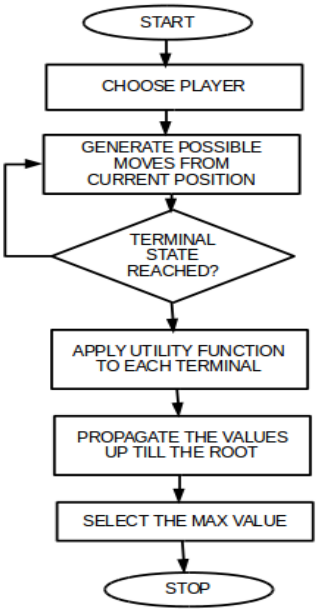
DD - Deterministic data → { }

NDD - Non deterministic data → { Input position, Outcome }

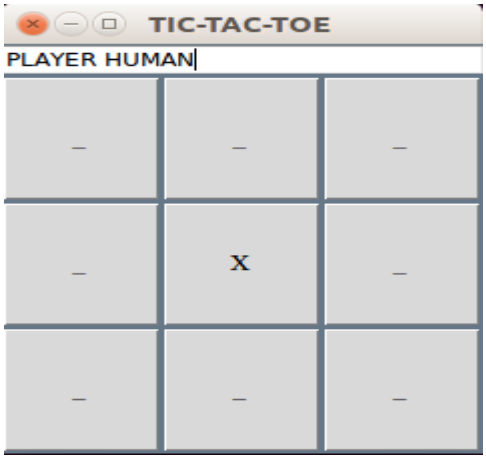
Success - Desired outcome generated → {Unification result }

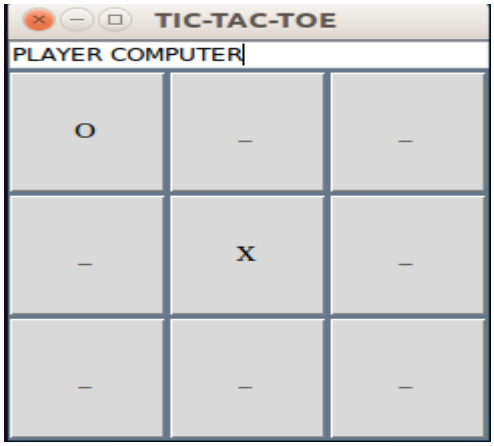
Failure - Desired outcome not generated or forced exit due to system error.

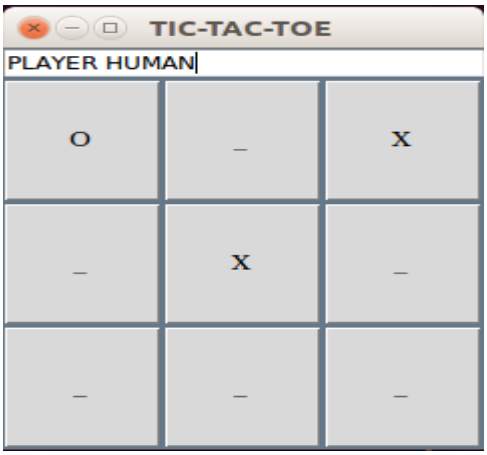
FLOWCHART :

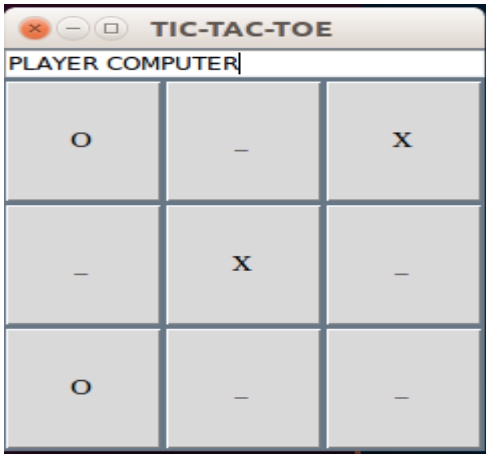


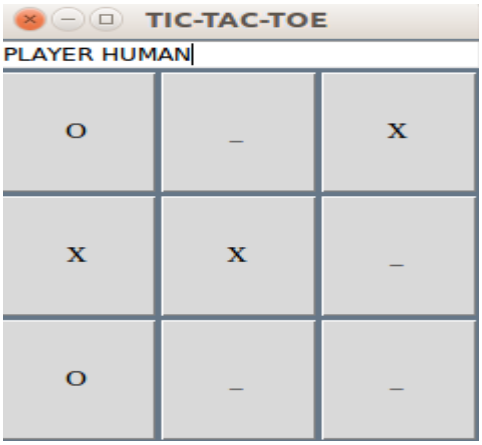
TEST CASES (DRAW CONDITION) :

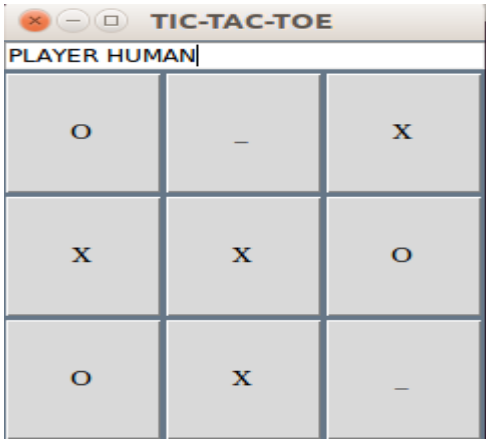


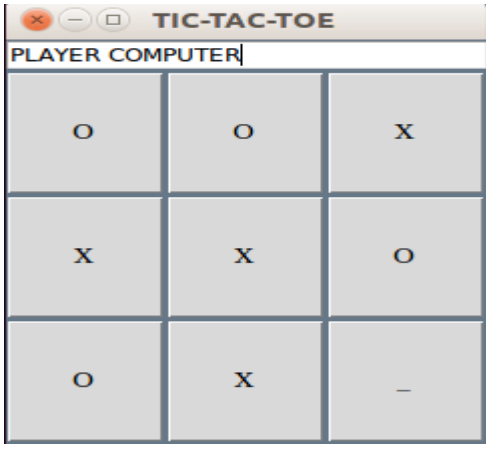


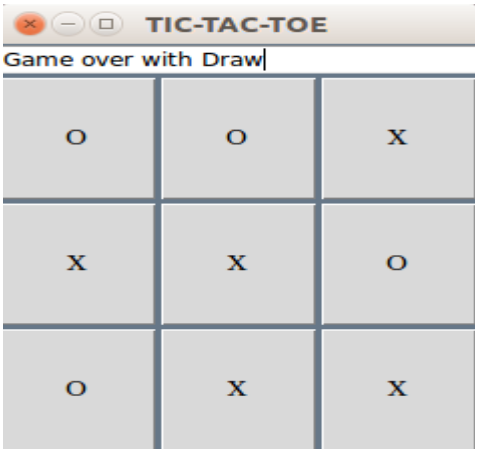




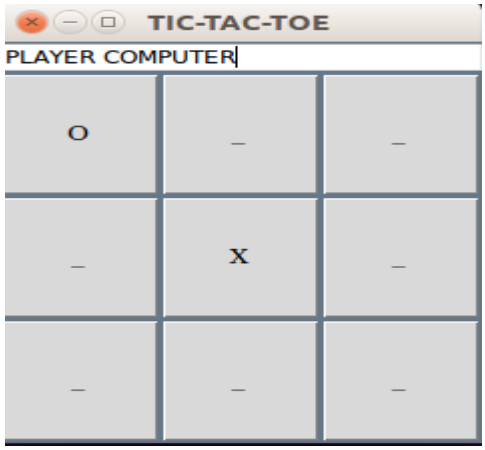


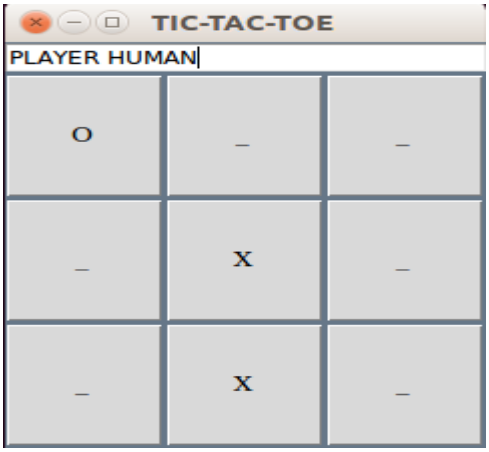


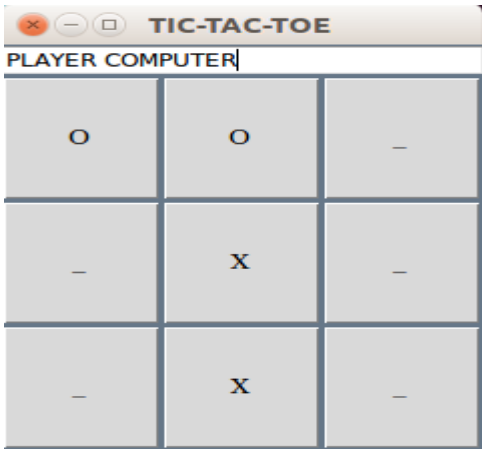


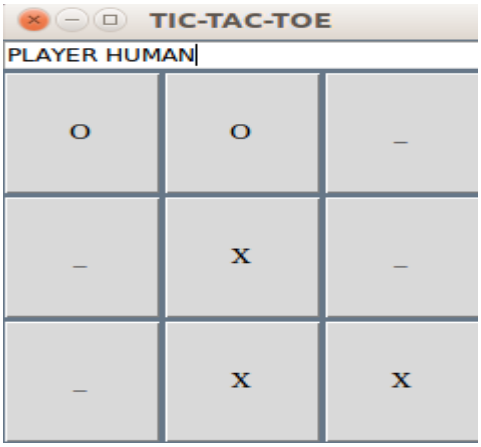


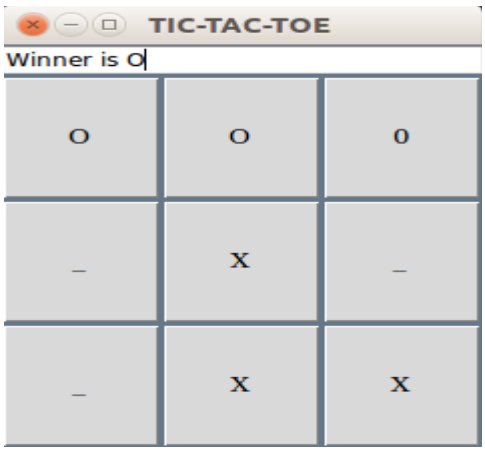
TEST CASES (WINNING CONDITION) :











SPACE AND TIME COMPLEXITIES :

If the maximum depth of the tree is m and there are b legal moves at each point, then the time complexity of the minimax algorithm is O(bm). The space complexity is O(bm.) for an algorithm that generates all actions at once, or O(m) for an algorithm that generates actions one at a time. For real games, of course, the time cost is totally impractical, but this algorithm serves as the basis for the mathematical analysis of games and for more practical algorithms.

CONCLUSION :

Hence we have successfully implemented the minimax algorithm for the tic-tac-toe game.

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	
To solve problems for multicore or distributed,concurrent/Parallel environments	√

FAQs

1. What is adversarial search?
2. What is zero sum game, static evalution function and terminal nodes?
3. What is alpha-beta prunning?
4. What is horizon effect and waiting for quiscense?

ASSIGNMENT NO-B4

PROBLEM STATEMENT :

Implementation of a simple NN for any suitable application (without tool)

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

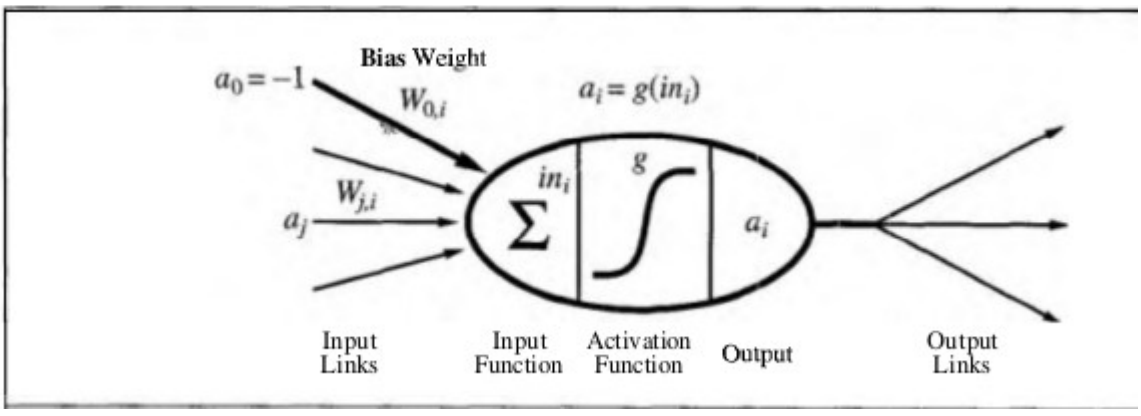
Neural Networks :

A neuron is a cell in the brain whose principal function is the collection, processing, and dissemination of electrical signals.The brain's information-processing capacity is thought to emerge primarily from networks of such neurons. For this reason, some of the earliest A1 work aimed to create artificial neural networks.

Neural networks are composed of nodes or units connected by directed links. A link from unit j to unit i serves to propagate the activation **aj** from **j** to **i.** Each link also has a numeric weight W(j,i) associated with it, which determines the strength and sign of the connection. Each unit i first computes a weighted sum of its inputs:

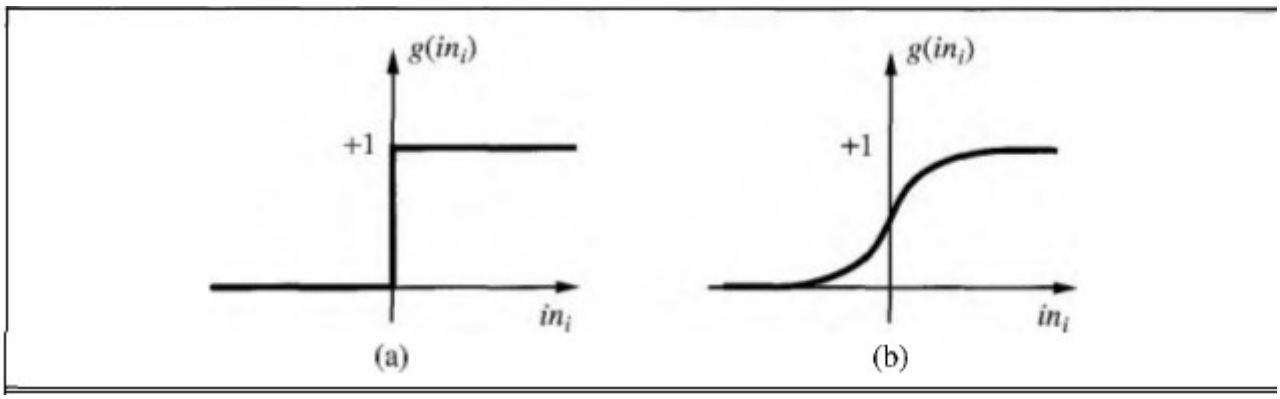
$$s = \sum_{i=0}^{n} w_i \cdot x_i$$

Then it applies an activation function to this sum to derive the output .The below figure shows the basic diagram of a perceptron .



The activation function g is designed to meet two desiderata. First, we want the unit to be "active" (near +1) when the **"right"** inputs are given, and "inactive" (near 0) when the **"wrong"** inputs are given. Second, the activation needs to be nonlinear, otherwise the entire neural network collapses into a simple linear function. Two choices for g are possible : the threshold function and the sigmoid function (also known as the logistic function). The

sigmoid function has the advantage of being differentiable, which we is important for the weight-learning algorithm. Notice that both functions have a threshold (either hard or soft) at zero; the bias weight sets the actual threshold for the unit, in the sense that the unit is activated when the weighted sum of "real" inputs (i.e., excluding the bias input) exceeds W(0,i)

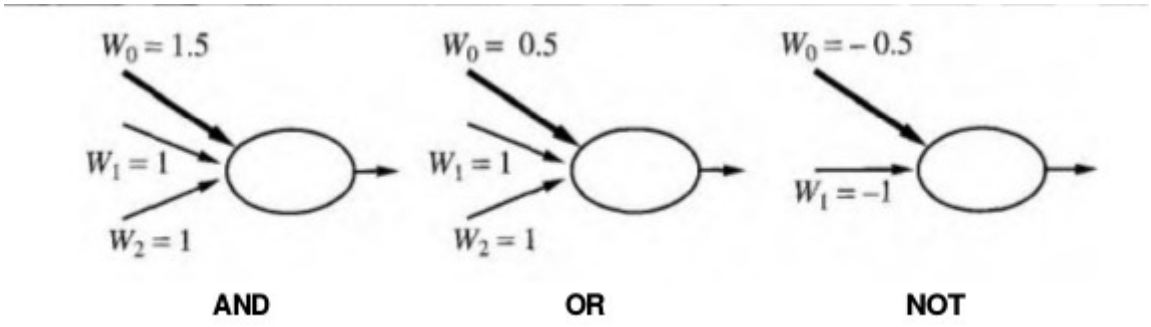


The above figure shows the two functions used to decide whether or not to fire a perceptron .

a. The threshold activation function, which outputs 1 when the input is positive and 0 otherwise.

b. The sigmoid function **$1/(1\ e^{-x})$**.

We can get a feel for the operation of individual units by comparing them with logic gates. One of the original motivations for the design of individual units was their ability to represent basic Boolean functions. The below figure shows how the Boolean functions AND, OR, and NOT can be represented by threshold units with suitable weights. This is important because it means we can use these units to build a network to compute any Boolean function of the inputs.



Network structures :

There are two main categories of neural network structures: acyclic or feed-forward networks and cyclic or recurrent networks. A feed-forward network represents a function of its current input ; thus, it has no internal state other than the weights themselves. A recurrent network, on the other hand, feeds its outputs back into its own inputs. This means that the activation levels of the network form a dynamical system that may reach a stable state or exhibit oscillations or even chaotic behavior. Moreover, the response of the network to a given

input depends on its initial state, which may depend on previous inputs. Hence, recurrent networks (unlike feed-forward networks) can support short-term memory. This makes them more interesting as models of the brain, but also more difficult to understand.

A neural network can be used for classification or regression. For Boolean classification with continuous outputs (e.g., with sigmoid units), it is traditional to have a single output unit, with a value over 0.5 interpreted as one class and a value bellow 0.5 as the other. For k-way classifier , one could divide the single output unit's range intlo k portions, but ittis more common to have k separate output units, with the value of each one representing the relative likelihood of that class given the current input.Feed-forward networks are usually arranged in layers, such that each unit receives input only from units in the immediately preceding layer. In the next two subsections, we will look at single layer networks, which have no hidden units, 2nd miultilayer networks, which have one or more layers of hidden units.

ALGORITHM :

function PERCEPTRON-LEARNING(*examples*, *network*) **returns** a perceptron hypothesis
inputs: *examples*, a set of examples, each with input $\mathbf{x} = x_1, \ldots, x_n$ and output y
network, a perceptron with weights W_j, $j = 0 \ldots n$, and activation function g

repeat
for each e **in** *examples* **do**
$in \leftarrow \sum_{j=0}^{n} W_j x_j[e]$
$Err \leftarrow y[e] - g(in)$
$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j[e]$
until some stopping criterion is satisfied
return NEURAL-NET-HYPOTHESIS(*network*)

BASIC LOGIC OF PROGRAM :

First we get a random input set from the training data. Then we calculate the dot product (sometimes also called scalar product or inner product) of the input and weight vectors. This is our (scalar) result, which we can compare to the expected value. If the expected value is bigger, we need to increase the weights, if it's smaller, we need to decrease them. This correction factor is calculated in the last line, where the error is multiplied with the learning rate (`eta`) and the input vector (`x`). It is then added to the weights vector, in order to improve the results in the next iteration.

PSEUDO-CODE :

Begin :

```
input  =  get_random_data ( dataset )
calculated_output = DotProduct ( input * weightVector  )
compare ( calculated_output , expected_output )
```

if similar :
    Training done
    Print output
else :
    Continue Training

End

INPUT :

Table containing expected input and output values .

EXPECTED OUTPUT :

Number of iterations taken to train the perceptron and the actual weights which give the correct result .

MATHEMATICAL MODEL :

Let P be the solution perspective .

P ={ S , E , I , O, F }

S = { Initial state of the perceptron containing expected input and output values for OR and AND GATE
}

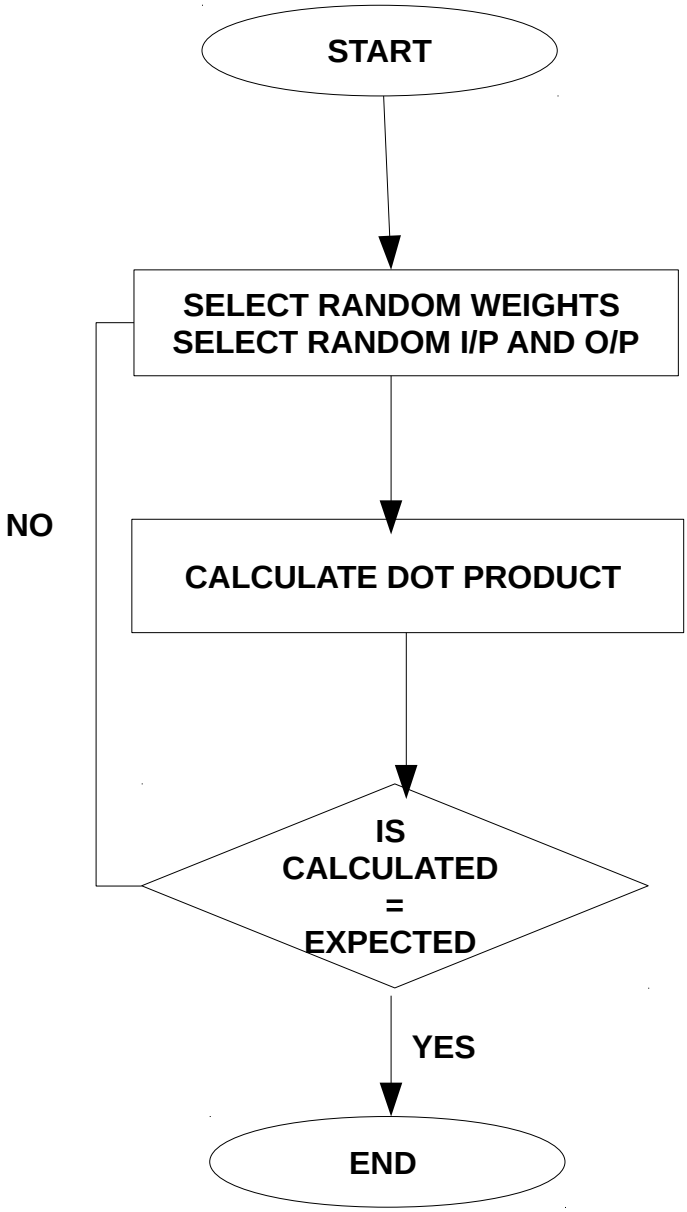
I = Input of the system → { I1 , I2 }
    where I1 = { Initial expected input and output }
        I2 = { ETA Rate }

O = Output of the system → { O1 , O2 }
    where O1 = { Final weights calculated }
        O2 = { No. of iterations }

F = Functions used → { f1 , f2 }
    where f1 = { Dot product of vectors }
        f2 = { Random weight generator }
        f2 = { Random choice selector }

E = End state of the system which shows that the perceptron has learned completely .

FLOWCHART :



TEST CASES :

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1 OR GATE	ETA = 0.2	FAST LEARNING	ITRATIONS = 9	Correct
2 OR GATE	ETA = 0.001	SLOW LEARNING	ITRATIONS = 1359	Correct
3 AND GATE	ETA = 0.1	SLOWER LEARNING	ITRATIONS = 40	Correct

TIME AND SPACE COMPLEXITY :

- TIME COMPLEXITY :

  O(N), N =No of samples in training data .

- SPACE COMPLEXITY :

  O(n), since the training data and error list varies with number of samples. The space consumed by other variables is much less as compared to the training data and error list and is also constant.

CONCLUSION :

Hence we have successfully implemented a perceptron on OR and AND gate .

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	
To solve problems for multicore or distributed,concurrent/Parallel environments	√

FAQs

1. What is neural network?
2. What is machine learning?
3. Explain different machine learning mechanism.
4. Explain Learning agent.

ASSIGNMENT NO-B5

1. Problem Statement: Implementation of any 2 uninformed search methods for a LPG company that wants to install a gas pipeline between five cities. The cost of pipeline installation is given in a table maintained using XML/JSON use C++/ Python/ Java/ Scala with Eclipse for the application. Calculate time and space complexities.

2. Objective:

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

3. Theory:

Strategies that know whether one non-goal state is "more promising" than another are called **informed search** or **heuristic search** strategies; they will be covered in Chapter 4. All search strategies are distinguished by the order in which nodes are expanded.

Depth-First Search:

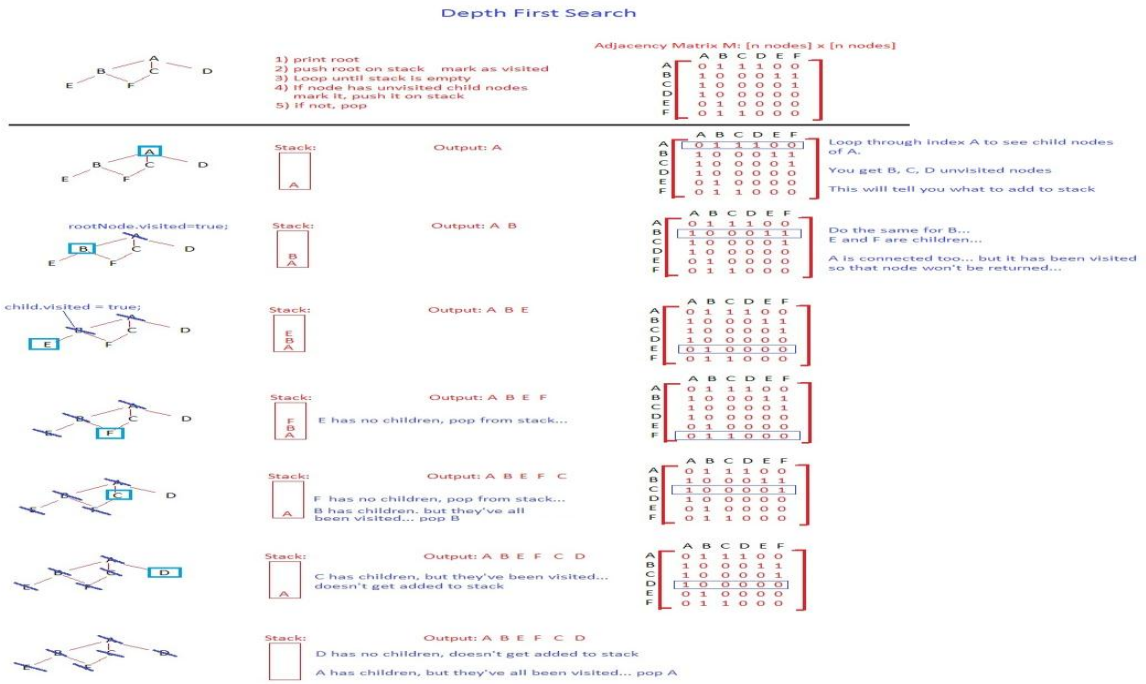
Depth-first search always expands the *deepest* node in the current fringe of the search tree. The progress of the search is illustrated in Figure 3.12. The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors. As those nodes are expanded, they are dropped from the fringe, so then the search "backs up" to the next shallowest node that still has unexplored successors.

This strategy can be implemented by TREE-SEARCH with a last-in-first-out (LIFO) queue, also known as a stack.

Breadth-First Search:

Breadth-first search is a simple strategy in which the root node is expanded first, then all the successors of the root node are expanded next, then their successors, and so on. In general, all the nodes are expanded at a given depth in the search tree before any nodes at the next level are expanded.

Breadth-first search can be implemented by calling TREE-SEARCH with an empty fringe that is a first-in-first-out (FIFO) queue, assuring that the nodes that are visited first will be expanded first. In other words, calling TREE-SEARCH results in a breadth-first search.

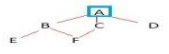


Depth First Search

1) print root
2) push root on stack mark as visited
3) Loop until stack is empty
4) If node has unvisited child nodes
   mark it, push it on stack
5) If not, pop

Adjacency Matrix M: [n nodes] x [n nodes]

	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	0	1
D	1	0	0	0	0	0
E	0	1	0	0	0	0
F	0	1	1	0	0	0



Stack: A

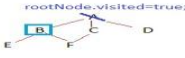
Output: A

Loop through index A to see child nodes of A.

You get B, C, D unvisited nodes

This will tell you what to add to stack

rootNode.visited=true;



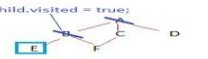
Stack: B A

Output: A B

Do the same for B...
E and F are children...

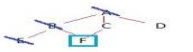
A is connected too... but it has been visited so that node won't be returned...

child.visited = true;



Stack: E B A

Output: A B E



Stack: F B A

Output: A B E F

E has no children, pop from stack...

Stack: A

Output: A B E F C

F has no children, pop from stack...
B has children. but they've all been visited... pop B

Stack: A

Output: A B E F C D

C has children, but they've been visited... doesn't get added to stack

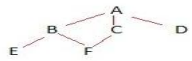
Stack:

Output: A B E F C D

D has no children, doesn't get added to stack

A has children, but they've all been visited... pop A

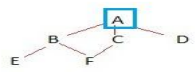
Breadth First Search



1) Print root node
2) Push root into Queue - mark as visited
3) Loop until Queue is empty
4) Remove node from queue, examine it
5) If examined node has unvisited children... mark them as visited...
   Print node
6) Insert unvisited nodes into Queue

Adjacency Matrix M: [n nodes] x [n nodes]

	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	0	1
D	1	0	0	0	0	0
E	0	1	0	0	0	0
F	0	1	1	0	0	0



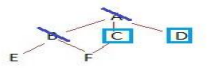
Queue: A

Output:

A has 3 unvisited children nodes

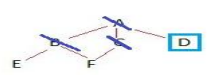
Queue: B C D

Output: A



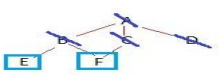
Queue: C D

Output: A B



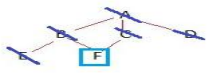
Queue: D

Output: A B C



Queue: E F

Output: A B C D

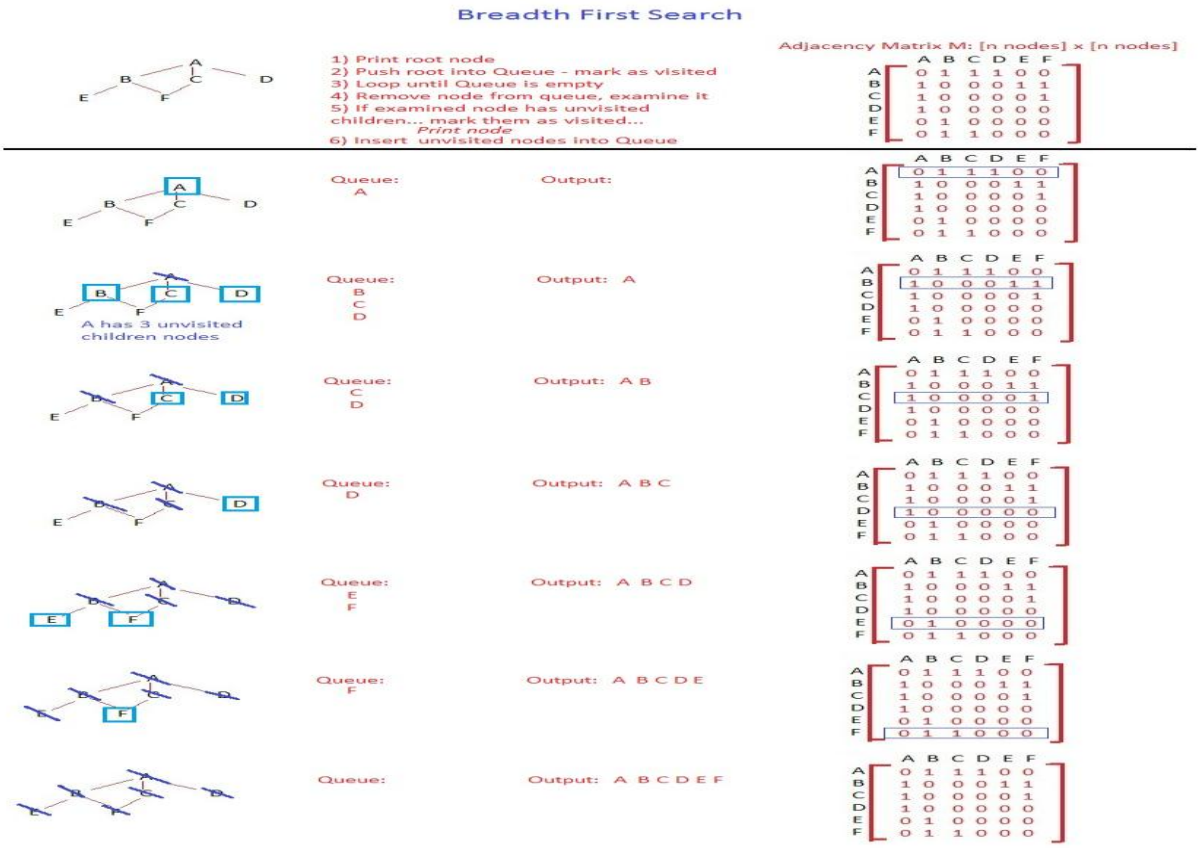


Queue: F

Output: A B C D E

Queue:

Output: A B C D E F



4. Algorithm:

1. Declare openlist
2. Add root node to our openlist
   while openlist not empty do following loops:
   a. retrieve then remove first node of our openlist
   b. check status of retrieved node
3. if it is the goal node then break loop and print solution
   if it is not goal node then:
   – expand retrieved node
   – ADD expanded node at THE BEGINNING of our openlist
   – continue loop
4. Openlist
   Using linked list data structure for openlist
5. Additional structure
   a. Hashmap to store currentState and its parent (without it we will not able to traceback)
   b. Hashmap to store currentState and its current level in tree (otherwise we can not limiting level and know how much step we already had)

5. Input For Assignment:
Path Adjancency matrix- File.xml

6. Expected Output: All the cities from start and finish connecting through common pipeline

7. Math Model:

Let S be the solution of the problem

S={s, e, i, o}

Where,

s=initial state
e=end state
i=input
o=output

s = {q, x, l| L!=NULL}
q=Queue/stack
x=starting vertex
l=List of vertices

i={v}

v=vertex of the tree

$v \in l$
$x \in l$

Enqueue(x)|Push(x)
$x \in v$

x=visited

o=Vertex reachable from v labelled as discovered and goal established
e= The system will be end in following conditions:
(i) If incorrect input is provided
(ii) Destination not mentioned correctly

8. Test Case(Testing):

Step	Test Step	Expected Result	Actual Result	Status
1	Enter valid data	Should accept the data	Accepted data	Pass
2	Enter valid goal state	Should accept the goal state	Successfully reached to goal state	Pass
3	Enter random goal state	Should accept goal state	Unsuccessful reaching goal	Fail
4	Random Initial State	Should accept state	Accepting as initial state	Pass

9. Time and Space Complexity:

Number of loops executing::BFS:5, DFS:3
Number recursive functions::3 calling each other and itself
Number of if cases::8loops*2=16 if cases executed 27 times

10. Conclusion:
Hence we have implemented the 2-uninformed search algorithms in order to install a pipeline between 5 cities.

11. Outcomes achieved (mark the outcomes achieved)

COURSE OUTCOME	ACHIEVED(√)
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness, embedded security and NLP.	√
To solve problems for multicore or distributed, concurrent/Parallel environments	

FAQs

1. What is blind search?
2. Differentiate between informed search and uninformed search techniques?
3. State and explain problem solving agent.
4. Explain drawback of Uniformed Search.

```
//lpg pipeline
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Map;
import java.util.Queue;

class Node {
  String city;
    LinkedList<Node> list;
}
public class SearchRomania {

    static Node[] node = new Node[20];
    static LinkedList<Node> path = new LinkedList<Node>();
    static LinkedList<Node> visited = new LinkedList<Node>();
    static Queue<Node> queue = new LinkedList<Node>();
    static boolean goalReached = false;
    static Map map = new HashMap();

    public static void initialize() {
        for (int i = 0; i < 5; i++) {
            node[i] = new Node();
            node[i].list = new LinkedList();
        }
        node[0].city = "oradea";
        node[1].city = "zerind";
        node[2].city = "arad";
        node[3].city = "timisoara";
        node[4].city = "lugoj";

//read from xml as integer
ReadXml rf=new ReadXml();
rf.readXML();
int city1=ReadXml.city1.charAt(1)-'0';

        node[0].list.add(node[ReadXml.city1.charAt(0)-'0']);
        node[0].list.add(node[ReadXml.city1.charAt(1)-'0']);
        node[1].list.add(node[ReadXml.city2.charAt(0)-'0']);
        node[1].list.add(node[ReadXml.city2.charAt(1)-'0']);
        node[2].list.add(node[ReadXml.city3.charAt(0)-'0']);
        node[2].list.add(node[ReadXml.city3.charAt(1)-'0']);
        node[2].list.add(node[ReadXml.city3.charAt(2)-'0']);
        node[3].list.add(node[ReadXml.city4.charAt(0)-'0']);
        node[3].list.add(node[ReadXml.city4.charAt(1)-'0']);
        node[4].list.add(node[ReadXml.city5.charAt(0)-'0']);
        node[4].list.add(node[ReadXml.city5.charAt(1)-'0']);

        System.out.println("Map for LPG pipeline installation::\n"+node[0].city+"-
>"+node[3].city+","+node[1].city+"\n"+node[1].city+"-
>"+node[2].city+","+node[0].city+"\n"+node[2].city+"-
>"+node[4].city+","+node[3].city+","+node[1].city+"\n"+node[3].city+"-
```

```
>"+node[2].city+","+node[4].city+"\n"+node[4].city+"->"+node[4].city+","+node[1].city);
    }
static void dfs(Node current, Node goal) {
  if (goalReached == false) {
          path.add(current);
          visited.add(current);

          if (current.city.equals(goal.city)) {
              goalReached = true;
             for (Node p : path) {
                System.out.println(p.city);
             }
          } else {
             for (Node next : current.list) {
                if (!visited.contains(next)) {
                   dfs(next, goal);
                   System.out.println(next.city);
             }
             }
             path.removeLast();
          }
       }
    }

    static void bfs(Node current, Node goal) {
       if (!visited.contains(current)) {
          visited.add(current);
          }
       if (goalReached == false) {
          for (Node child : current.list) {
             if (!visited.contains(child)) {
                visited.add(child);
                queue.add(child);
                map.put(child, current);

                if (child.city.equals(goal.city)) {
                   goalReached = true;
                   path.add(child);
                   path.add(current);

                   while (path.getLast() != visited.get(0)) {
                      current = (Node) map.get(current);
                      path.add(current);
                   }
                   while(!queue.isEmpty()){
                      System.out.println(queue.remove().city);
                    }
                   while (!path.isEmpty()) {
                      System.out.println(path.removeLast().city);
                   }
                   return;
                }
```

```
            }
          }
          bfs(queue.remove(), goal);
       }
    }

    public static void main(String[] args) {
       initialize();
       Node start=node[0];
       Node goal=node[1];
       String argr="oradea",argr1="arad";
       System.out.println("__________LPG Pipeline Installation between 5 cities_________");
System.out.println("Path starts from "+argr+" to "+argr1);

      for (int i = 0; i < 5; i++) {
          if (node[i].city.equals(argr)) {
             start = node[i];
          }
          if (node[i].city.equals(argr1)) {
             goal =node[i];
          }
       }
      System.out.println("Using DFS-----------");
         dfs(start, goal);

     System.out.println("Using BFS-----------");

         bfs(start, goal);
            System.out.println("Total number of nodes expanded: " + visited.size());
    }
}
```

//**lpg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<lpg>
 <city id="1">
  <oradea>41</oradea>
  <zerind>20</zerind>
<arad>431</arad>
  <timisoara>24</timisoara>
<lugoj>13</lugoj>
</city>
</lpg>
```

//**ReadXml.java**

```
import java.io.File;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
```

```
public class ReadXml {
          public static String city1,city2,city3,city4,city5;
 public void readXML() {
  try {

   File xmlFile = new File("E:/BE/Cl2 Pract/xmlFiles/lpg.xml");
   DocumentBuilderFactory documentFactory = DocumentBuilderFactory
     .newInstance();
   DocumentBuilder documentBuilder = documentFactory
     .newDocumentBuilder();
   Document doc = documentBuilder.parse(xmlFile);

   doc.getDocumentElement().normalize();
   NodeList nodeList = doc.getElementsByTagName("city");

   for (int temp = 0; temp < nodeList.getLength(); temp++) {
    Node node = nodeList.item(temp);
    if (node.getNodeType() == Node.ELEMENT_NODE) {

     Element student = (Element) node;

     /*System.out.println("city id : "
       + student.getAttribute("id"));

     System.out.println("oradea : "
       + student.getElementsByTagName("oradea").item(0)
         .getTextContent());*/
     city1=student.getElementsByTagName("oradea").item(0).getTextContent();
    /* System.out.println("zerind : "
       + student.getElementsByTagName("zerind").item(0)
         .getTextContent());*/
    city2=student.getElementsByTagName("zerind").item(0).getTextContent();
    /* System.out.println("arad : "
       + student.getElementsByTagName("arad").item(0).getTextContent());*/
    city3=student.getElementsByTagName("arad").item(0).getTextContent();
    /* System.out.println("timisoara : "
       + student.getElementsByTagName("timisoara").item(0)
         .getTextContent());*/
   city4=student.getElementsByTagName("timisoara").item(0).getTextContent();
     /*System.out.println("lugoj : "+ student.getElementsByTagName("lugoj").item(0)
                 .getTextContent());*/
   city5=student.getElementsByTagName("lugoj").item(0).getTextContent();

    }

   }
  } catch (Exception e) {
   e.printStackTrace();
  }
 }
}
```

Output:

Map for LPG pipeline installation::
oradea->timisoara,zerind
zerind->arad,oradea
arad->lugoj,timisoara,zerind
timisoara->arad,lugoj
lugoj->lugoj,zerind
_____LPG Pipeline Installation between 5 cities_____
Path starts from oradea to arad
Using DFS----------
oradea
lugoj
zerind
arad
arad
zerind
timisoara
lugoj
Using BFS----------
Total number of nodes expanded: 4

ASSIGNMENT NO-B6

1. Problem Statement: Case study on LIBSVM -A Library for Support Vector Machines. (using LDA for dimensionality reduction).

2. Objective:

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

3. Theory:

Support Vector Machine:

The support vector machines (SVM) were created by Vladimir Vapnik in the 1990s and can be used for solving classification and regression tasks. They are based on the principle of structural risk minimization, which enhances model robustness by ensuring that the model complexity is not too high as measured by the so called VC-dimension (Vapnik, 1998). For comparison, neural networks and other traditional black box techniques normally minimize the empirical risk which basically is the average quadratic error over a number of samples, the training set. Within the SVM framework, radial basis networks, single hidden layer sigmoidal neural networks as well as other kinds of models can be set up, depending on the chosen kernel. A nice property of the SVM is that it yields a unique optimal solution of the resulting optimization problem.

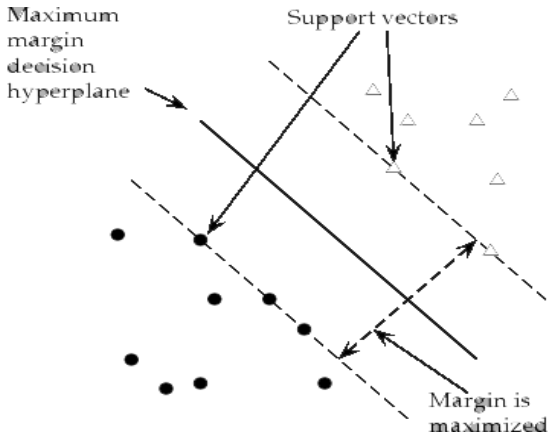


Fig.1. The support vectors are the 5 points right up against the margin of the classifier.

For two-class, separable training data sets, there are lots of possible linear separators. Intuitively, a decision boundary drawn in the middle of the void between data items of the two classes seems better than one which approaches very close to examples of one or both classes. While some learning methods such as the perceptron algorithm, find just any linear separator, others, like Naive Bayes, search for the best linear separator according to some criterion. The SVM in particular defines the criterion to be looking for a decision surface that is maximally far away from any data point. This distance from the decision surface to the closest data point determines the margin of the classifier. This method of construction necessarily means that the decision function for an SVM is fully specified by a (usually small) subset of the data which defines the position of the separator. These points are referred to as the *support vectors* (in a vector space, a point can be thought of as a vector between the origin and that point). Figure.1 shows the margin and support vectors for a sample problem. Other data points play no part in determining the decision surface that is chosen.

LIBSVM:
LIBSVM is an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It supports multi-class classification.

The primary goal is to help users from other fields to easily use SVM as a tool. **LIBSVM** provides a simple interface where users can easily link it with their own programs. Main features of **LIBSVM** include

- Different SVM formulations
- Efficient multi-class classification
- Cross validation for model selection
- Probability estimates
- Various kernels (including precomputed kernel matrix)
- Weighted SVM for unbalanced data
- Both C++ and Java sources
- GUI demonstrating SVM classification and regression
- Python, R, MATLAB, Perl, Ruby, Weka, Common LISP, CLISP, Haskell, OCaml, LabVIEW, and PHP interfaces. C# .NET code and CUDA extension is available. It's also included in some data mining environments: RapidMiner, PCP, and LIONsolver.
- Automatic model selection which can generate contour of cross validation accuracy.

The programs:
How to use the most important executables here. The filenames are a little bit different under Unix and Windows, apply common sense to see:
svmtrain : Use your data for training. Running SVM is often referred to as 'driving trains' by its non-native English speaking authors because of this program. svmtrain accepts some specifically format which will be explained below and then generate a 'Model' file. You may think of a 'Model' as a storage format for the internal data of SVM. This should appear very reasonable after some thought, since training with data is a time-consuming process, so we 'train' first and store the result enabling the 'predict' operation to go much faster.
svmpredict
Output the predicted class of the new input data according to a pre-trained model.
svmscale
Rescale data. The original data maybe too huge or small in range, thus we can rescale them to the proper range so that training and predicting will be faster.

File Format
libsvm "heart_scale": This is the input file format of SVM. You may also refer to the file "heart_scale" which is bundled in official libsvm source archive.
[label] [index1]:[value1] [index2]:[value2] ...
[label] [index1]:[value1] [index2]:[value2] ...
.
.
One record per line, as:
+1 1:0.708 2:1 3:1 4:-0.320 5:-0.105 6:-1
label
Sometimes referred to as 'class', the class (or set) of your classification. Usually we put integers here.
index
Ordered indexes. usually continuous integers.
value

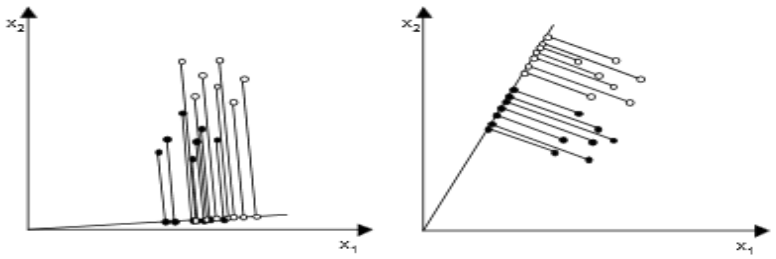
The data for training. Usually lots of real (floating point) numbers.

Linear Discriminant Analysis, two-classes (1)

- **The objective of LDA is to perform dimensionality reduction while preserving as much of the class discriminatory information as possible**
  - Assume we have a set of D-dimensional samples $\{x_1, x_2, \ldots, x_N\}$, N_1 of which belong to class ω_1, and N_2 to class ω_2. We seek to obtain a scalar y by projecting the samples x onto a line

$$y = w^T x$$

  - Of all the possible lines we would like to select the one that maximizes the separability of the scalars
    - This is illustrated for the two-dimensional case in the following figures



4. Conclusion:
Hence we have done case study of the LIBVSM tool for Support Vector Machines.

5. Outcomes achieved (mark the outcomes achieved)

COURSE OUTCOME	ACHIEVED(√)
Problem solving abilities for smart devices.	
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness, embedded security and NLP.	√
To solve problems for multicore or distributed, concurrent/Parallel environments	

FAQs
1. What is support vector machine?
2. What is classification?
3. What is decision trees? Explain with example.
4. What is inductive learning? What is Okham's Razor?

ASSIGNMENT NO-B7

PROBLEM STATEMENT :

Developing an book recommender (a book that the reader should read and is new) Expert system or (any other).

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

PROLOG :

Prolog is a general purpose logic programming language associated with artificial intelligence and computational linguistics. Prolog has its roots in first-order logic, a formal logic, and unlike many other programming languages, Prolog is declarative: the program logic is expressed in terms of relations, represented as facts and rules. A computation is initiated by running a *query* over these relations.

There are only three basic constructs in Prolog: facts, rules, and queries. A collection of facts and rules is called a knowledge base (or a database) and Prolog programming is all about writing knowledge bases. That is, Prolog programs simply are knowledge bases, collections of facts and rules which describe some collection of relationships that we find interesting. So how do we use a Prolog program? By posing queries. That is, by asking questions about the information stored in the knowledge base.

There are four kinds of term in Prolog: atoms, numbers, variables, and complex terms (or structures). Atoms and numbers are lumped together under the heading constants, and constants and variables together make up the simple terms of Prolog.

Atoms

An atom is either:

1. A string of characters made up of upper-case letters, lower-case letters, digits, and the underscore character, that begins with a lower-case letter. Here are some examples: butch , big_kahuna_burger , listens2Music and playsAirGuitar .
2. An arbitrary sequence of characters enclosed in single quotes. For example ’ Vincent ’, ’ The Gimp ’, ’ Five_Dollar_Shake ’, ’ &^%&#@$ &* ’, and ’ ’. The sequence of characters between the single quotes is called the atom name. Note that we are allowed to use spaces in such atoms; in fact, a common reason for using single quotes is so we can do precisely that.
3. A string of special characters. Here are some examples: @= and ====> and ; and :- are all atoms. As we have seen, some of these atoms, such as ; and :- have a pre-

defined meaning.

Numbers

Real numbers aren't particularly important in typical Prolog applications. So although most Prolog implementations do support floating point numbers or floats (that is, representations of real numbers such as 1657.3087 or π)

Variables

A variable is a string of upper-case letters, lower-case letters, digits and underscore characters that starts either with an upper-case letter or with an underscore. For example, X , Y , Variable , _tag , X_526 , List , List24 , _head , Tail , _input and Output are all Prolog variables.

The variable _ (that is, a single underscore character) is rather special. It's called the anonymous variable .

Complex terms

Constants, numbers, and variables are the building blocks: now we need to know how to fit them together to make complex terms. Recall that complex terms are often called structures.

Complex terms are build out of a functor followed by a sequence of arguments. The arguments are put in ordinary parentheses, separated by commas, and placed after the functor. Note that the functor has to be directly followed by the parenthesis; you can't have a space between the functor and the parenthesis enclosing the arguments. The functor must be an atom. That is, variables cannot be used as functors. On the other hand, arguments can be any kind of term.

CONCEPT OF UNIFICATION :

PROLOG uses the concept of unification internally . Two terms unify if they are the same term or if they contain variables that can be uniformly instantiated with terms in such a way that the resulting terms are equal.

This means, for example, that the terms mia and mia unify, because they are the same atom. Similarly, the terms 42 and 42 unify, because they are the same number, the terms X and X unify, because they are the same variable, and the terms woman(mia) and woman(mia) unify, because they are the same complex term. The terms woman(mia) and woman(vincent) , however, do not unify, as they are not the same (and neither of them contains a variable that could be instantiated to make them the same).

Now, what about the terms mia and X ? They are not the same. However, the variable X can be instantiated to mia which makes them equal. So, by the second part of our working

definition, mia and X unify. Similarly, the terms woman(X) and woman(mia) unify, because they can be made equal by instantiating X to mia . How about loves(vincent,X) and loves(X,mia) ? No. It is impossible to find an instantiation of X that makes the two terms equal. Do you see why? Instantiating X to vincent would give us the terms loves(vincent,vincent) and loves(vincent,mia) , which are obviously not equal. However, instantiating X to mia, would yield the terms loves(vincent,mia) and loves(mia,mia) , which aren't equal either.

FIRST ORDER LOGIC :

FOL assumes that the world contains Objects , Relations and Functions .The basic syntactic elements of first-order logic are the symbols that stand for objects, relations, and functions. The symbols, therefore, come in three kinds: constant symbols, which stand for objects; predicate symbols, which stand for relations; and function symbols, which stand for functions.

Syntax of First-order Logic: Basic Elements

Symbols

Constants	$KingJohn,\ 2,\ Koblenz,\ C, \ldots$
Predicates	$Brother,\ >,\ =,\ \ldots$
Functions	$Sqrt,\ LeftLegOf,\ \ldots$
Variables	$x,\ y,\ a,\ b,\ \ldots$
Connectives	$\wedge \quad \vee \quad \neg \quad \Rightarrow \quad \Leftrightarrow$
Quantifiers	$\forall \quad \exists$

Terms :

A term is a logical expression that refers to an object. Constant symbols are therefore terms, but it is not always convenient to have a distinct symbol to name every object. For example, in English we might use the expression "King John's left leg" rather than giving a name to his leg. This is what function symbols are for: instead of using a constant symbol, we use **LeftLeg(John).** In the general case, a complex term is formed by a function symbol followed by a parenthesized list of terms as arguments to the function symbol.

Atomic sentences :

Now that we have both terms for referring to objects and predicate symbols for referring to relations, we can put them together to make atomic sentences that state facts. An atomic sentence is formed from a predicate symbol followed by a parenthesized list of terms:

Brother(Richard, John).

This states, under the intended interpretation given earlier, that Richard the Lionheart is the brother of King John .

Atomic sentences can have complex terms as arguments.

Married (Father(Richard), Mother(John))

Complex sentences :

We can use logical connectives to construct more complex sentences, just as in propositional calculus. The semantics of sentences formed with logical connectives is identical to that in the propositional case.

¬ Brother(LeftLeg(Richard), John)

Brother(Richard, John) Λ Brother(John, Richard)

Quantifiers :

Once we have a logic that allows objects, it is only natural to want to express properties of entire collections of objects, instead of enumerating the objects by name. Quantifiers let us do this. First-order logic contains two standard quantifiers, called universal and existential.

A x King (x) Λ Person (x)

з x Crown(x) → OnHead(x, John) .

ALGORITHM :

1. Define all the clauses and predicates to be used .
2. Create a customized Expert window .
3. Start a interactive session . Ask questions to the user .
4. Based on User's answer's make ask more appropriate questions .
5. Continue step 4 until we get a complete idea of user's choice .
6. Recommend appropriate book .

INPUT :

User's answers in the form of choices based on questions asked .

EXPECTED OUTPUT :

Appropriate recommendation of book .

MATHEMATICAL MODEL :

Let P be the solution perspective .

P ={ S , E , I , O, F }

S = { Initial state of the expert system consisting of a intercative session }

I = Input of the system → { I1 , I2 }
where I1 = { User's answers in the form of choices based on questions asked }
I2 = { Questions based on users choice repeatedly }

O = Output of the system → { O1 , O2 }
where O1 = { Appropriate response question }
O2 = { Appropriate recommended Book }

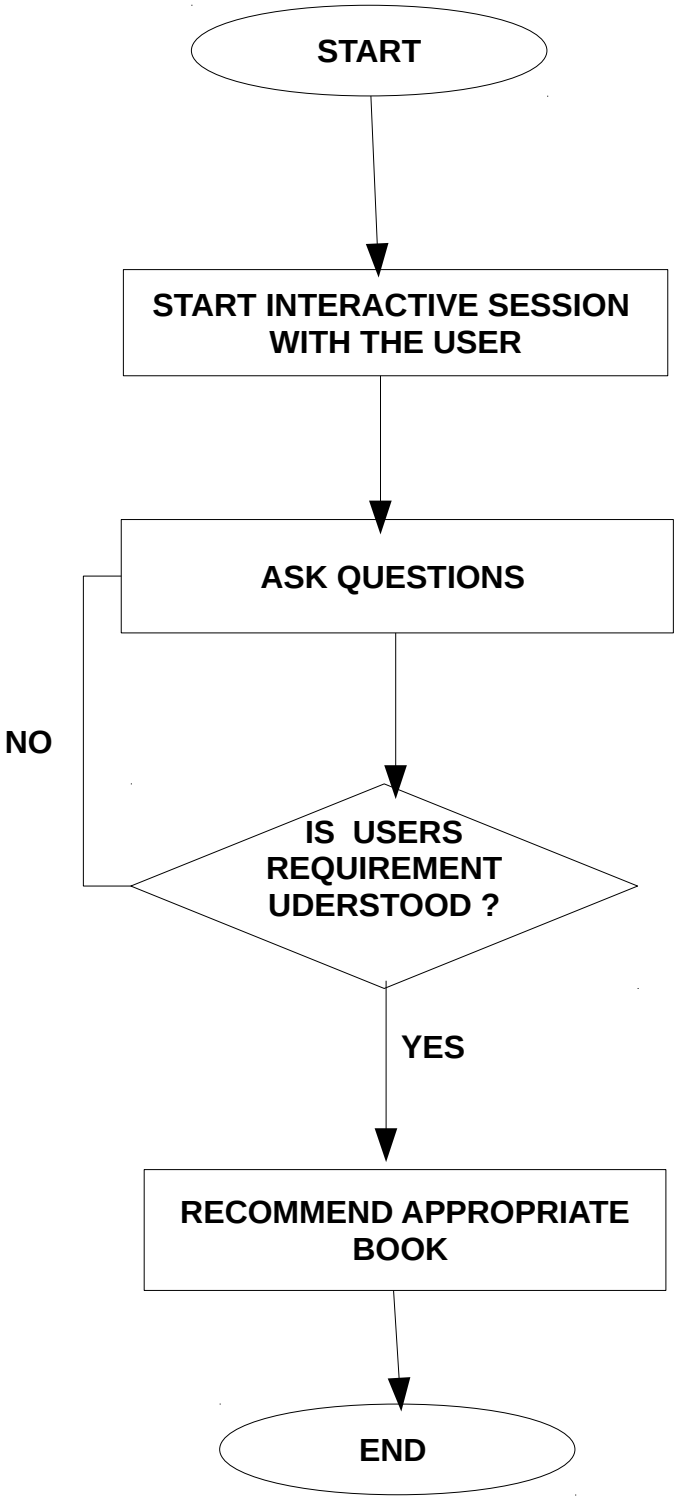
F = Functions used → { f1 , f2 , f3 }
where f1 = { readint for getting choice }
f2 = {write for writing to screen }
f3 = { predicate() for selecting appropriate question }

E = End state of the system which shows the proper recommendationn of book according to users need .

TEST CASES :

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1	CHILD	CHILDISH BOOKS	COMICS, FAIRYTALES	Correct
2	YOUTH	LATEST TREND BOOKS	NOVELS, AUTOBIOGRAPHY	Correct
3	OLD AGE	SPIRITUAL BOOKS	MYTHOLOGY, RELIGION HYMNS	Correct

FLOWCHART :



CONCLUSION :

Hence we have successfully implemented a smart expert Book Recommender system using PROLOG.

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	√
To solve problems for multicore or distributed,concurrent/Parallel environments	√

FAQs

1. What is expert system?
2. What is ontology?
3. What is decision network?
4. What is decision theorotic expert system?

ASSIGNMENT NO-B8

1. Problem Statement:
Develop a POP for scheduling your higher studies exam. Assume suitable data like college submission schedule, college exams, Constraint that a paper publication is must to appear before the exam, a family function at home and so on.

2. Objective:
To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

3. Theory
Partial-Order Planning:

The idea of a **partial-order planner** is to have a partial ordering between actions and only commit to an ordering between actions when forced. This is sometimes also called a **non-linear planner**, which is a misnomer because such planners often produce a linear plan.

For uniformity, treat start as an action that achieves the relations that are true in the initial state, and treat finish as an action whose precondition is the goal to be solved. The pseudoaction start is before every other action, and finish is after every other action. The use of these as actions means that the algorithm does not require special cases for the initial situation and for the goals. When the preconditions of finish hold, the goal is solved.

An action, other than start or finish, will be in a partial-order plan to achieve a precondition of an action in the plan. Each precondition of an action in the plan is either true in the initial state, and so achieved by start, or there will be an action in the plan that achieves it.

We must ensure that the actions achieve the conditions they were assigned to achieve. Each precondition P of an action act_1 in a plan will have an action act_0 associated with it such that act_0 achieves precondition P for act_1. The triple $\langle act_0,P,act_1\rangle$ is a **causal link**. The partial order specifies that action act_0 occurs before action act_1, which is written as $act_0 < act_1$. Any other action A that makes P false must either be before act_0 or after act_1.

Informally, a partial-order planner works as follows: Begin with the actions start and finish and the partial order start < finish. The planner maintains an agenda that is a set of $\langle P,A\rangle$ pairs, where A is an action in the plan and P is an atom that is a precondition of A that must be achieved. Initially the agenda contains pairs $\langle G,finish\rangle$, where G is an atom that must be true in the goal state.

At each stage in the planning process, a pair $\langle G,act_1\rangle$ is selected from the agenda, where P is a precondition for action act_1. Then an action, act_0, is chosen to achieve P. That action is either already in the plan - it could be the start action, for example - or it is a new action that is added to the plan. Action act_0 must happen before act_1 in the partial order. It adds a causal link that records that act_0 achieves P for action act_1. Any action in the plan that deletes P must happen either before act_0 or after act_1. If act_0 is a new action, its preconditions are added to the agenda, and the process continues until the agenda is empty.

This is a non-deterministic procedure. The "choose" and the "either ...or ..." form choices that must be searched over. There are two choices that require search:

- which action is selected to achieve G and
- whether an action that deletes G happens before act_0 or after act_1.

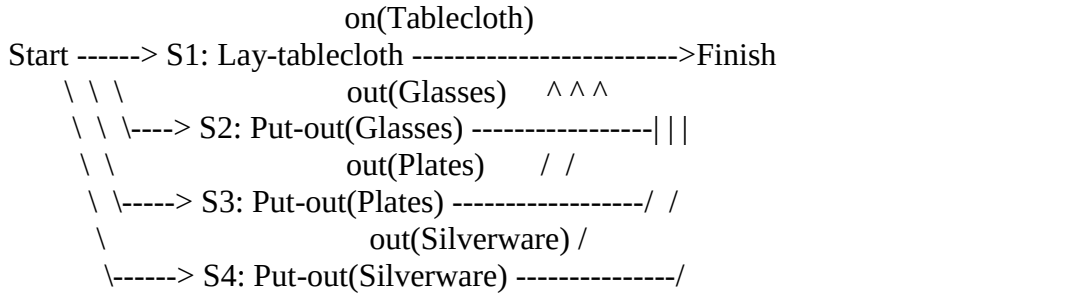
Example

- Goal: Set the table, i.e., on(Tablecloth) ^ out(Glasses) ^ out(Plates) ^ out(Silverware)
- Initial State: clear(Table)
- Operators:
    1. Lay-tablecloth

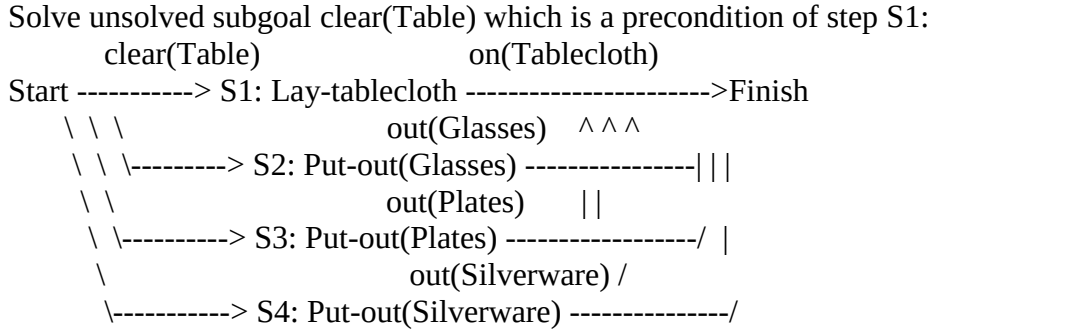
       P: clear(Table)
       E: on(Tablecloth), ~clear(Table)

    2. Put-out(x)
       P: none
       E: out(x), ~clear(Table)

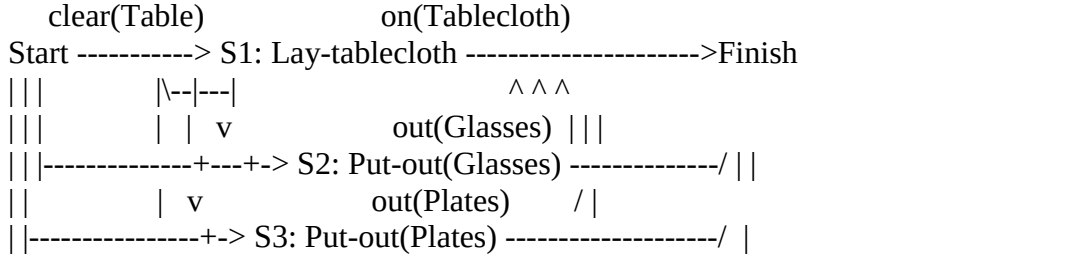
- Searching for a Solution in Plan Space
    1. Initial Plan
       Start ----------> Finish
    2. Solve 4 unsolved goals in Finish by adding 4 new steps with the minimal temporal constraints possible:



    3. Solve unsolved subgoal clear(Table) which is a precondition of step S1:



    4. Fix threats caused by steps S2, S3, and S4 on the link from Start to S1. That is, clear(Table) is a necessary precondition of S1 that is created by step Start. But S2 causes clear(Table) to be deleted (negated), so if S2 came before S1, clear(Table) wouldn't be true and step S1 couldn't be performed. Therefore, add a temporal constraint that forces S2 to come anytime after S1. That is, add constraint S1 < S2. Similarly, add S1 < S3, and S1 < S4, resulting in the new plan:



| v out(Silverware) /
|----------> S4: Put-out(Silverware) ----------/

5. No threats and no unsolved goals in this plan, so it is a **complete plan** (i.e., a solution to the planning problem). Any total ordering of the steps implied by this partial-order plan is a solution plan. Here, there are six possible plans, where the first step is S1, and the steps S2, S3, and S4 follow in any order. (Don't include the pseudo-steps Start and Finish.)

4. Algorithm:

1. on-deterministic procedure PartialOrderPlanner(Gs)
2. Inputs
   Gs: set of atomic propositions to achieve
3. Output
   linear plan to achieve Gs
4. Local: Agenda: set of ⟨P,A⟩ pairs where P is atom and A an action
   Actions: set of actions in the current plan
   Constraints: set of temporal constraints on actions
   CausalLinks: set of $\langle act_0,P,act_1\rangle$ triples
5. Agenda ←{⟨G,finish⟩:G ∈Gs}
   Actions ←{start,finish}
   Constraints ←{start<finish}
   CausalLinks ←{}
   repeat
6. select and remove $\langle G,act_1\rangle$ from Agenda
   Either
   choose act_0 ∈Actions such that act_0 achieves G
   or
   choose act_0 ∉Actions such that act_0 achieves G
   Actions ←Actions ∪{act_0}
   Constraints ←add_const(start<act_0,Constraints)
7. for each CL∈CausalLinks do
   Constraints ←protect(CL,act_0,Constraints)

8. Agenda ←Agenda ∪{⟨P,act_0⟩: P is a precondition of act_0 }

9. Constraints ←add_const(act_0<act_1,Constraints)
   CausalLinks ∪ {$\langle act_o,G,act_1\rangle$}
   for each A∈Actions do
   Constraints ←protect($\langle act_o,G,act_1\rangle$,A,Constraints)
10. until Agenda={}
    return total ordering of Actions consistent with ConstraintsInput for assignment:

User selection- O's placement

5. Expected Output:

Solution in Plan Space

6. Math Model:

Let S be the Solution of the Problem-
S={s, e, i, o}

Where,

s=initial state
e=end state
i=input
o=output

s=initialize the space plan
s= {start,finish,goal}

start={clear the space_object}
finish={Final state}

goal={ Set the table}

i={space_plan,start}

o={Partial Order plan}

e= The system will be end in following conditions:

(i) If incorrect input is provided
(ii) System failure

7. Test Case(Testing):

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1	Invalid input	Should not accept	Not accepting	Pass
2	Valid goal state	Should accept and return POP	Accepting goal state	Pass
3	Valid Start value	Should start searching for space plan	Searching for space plan	Pass

8. Conclusion:

Hence, we have implemented the Partial Order Planning for scheduling higher studies.

9. Outcomes achieved (mark the outcomes achieved)

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices. Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP. To solve problems for multicore or distributed,concurrent/Parallel environments	√ √

FAQs

1. What is differnce between classial planning and POP?
2. Explain steps to create POP?
3. Which are the components of POP?
4. What is HTN planning?

ASSIGNMENT NO-B9

PROBLEM STATEMENT :

Implement k-means for clustering data of children belonging to different age groups to perform some specific activities. Formulate the Feature vector for following parameters:
i. height
ii. weight
iii. age
iv. IQ
Formulate the data for 40 children to form 3 clusters.

OBJECTIVE:

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

Cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. The set of clusters resulting from a cluster analysis can be referred to as a clustering. In this context, different clustering methods may generate different clusterings on the same data set. The partitioning is not performed by humans, but by the clustering algorithm. Hence, clustering is useful in that it can lead to the discovery of previously unknown groups within the data.

k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

Suppose a data set, D, contains n objects in Euclidean space. Partitioning methods distribute the objects in D into k clusters, C 1 , . . . , C k , that is, C i $\subset$ D and C i $\cap$ C j = $\emptyset$ for (1 $\leq$ i, j $\leq$ k). An objective function is used to assess the partitioning quality so that objects within a cluster are similar to one another but dissimilar to objects in other clusters. This is, the objective function aims for high intracluster similarity and low intercluster similarity. A centroid-based partitioning technique uses the centroid of a cluster, C i , to represent that cluster. Conceptually, the centroid of a cluster is its center point. The centroid can be defined in various ways such as by the mean or medoid of the objects (or points) assigned to the cluster. The difference between an object p $\in$ C i and c i , the representative of the cluster, is measured by dist(p, c i), where dist(x, y) is the Euclidean distance between two points x and y. The quality of cluster C i can be measured by the within cluster variation, which is the sum of squared error between all objects in C i and the centroid c i , defined as

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(\boldsymbol{p}, \boldsymbol{c_i})^2,$$

The k-means algorithm defines the centroid of a cluster as the mean value of the points within the cluster. It proceeds as follows. First,it randomly selects k of the objects in D, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the Euclidean distance between the object and the cluster mean. The k-means algorithm then iteratively improves the within-cluster variation.
For each cluster, it computes the new mean using the objects assigned to the cluster in the previous iteration. All the objects are then reassigned using the updated means as the new cluster centers. The iterations continue until the assignment is stable, that is, the clusters formed in the current round are the same as those formed in the previous round.

ALGORITHM:
The k-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

(1) arbitrarily choose k objects from D as the initial cluster centers;

(2) repeat

(3) (re)assign each object to the cluster to which the object is the most similar,

based on the mean value of the objects in the cluster;

(4) update the cluster means, that is, calculate the mean value of the objects for

each cluster;

(5) until no change;

PSEUDO CODE:

```
def kmeans(dataSet, k):
    # Initialize centroids randomly
    numFeatures = dataSet.getNumFeatures()
    centroids = getRandomCentroids(numFeatures, k)

    # Initialize book keeping vars.
    iterations = 0
    oldCentroids = None

    # Run the main k-means algorithm
    while not shouldStop(oldCentroids, centroids, iterations):
        # Save old centroids for convergence test. Book keeping.
        oldCentroids = centroids
        iterations += 1

        # Assign labels to each datapoint based on centroids
        labels = getLabels(dataSet, centroids)

        # Assign centroids based on datapoint labels
        centroids = getCentroids(dataSet, labels, k)

    return centroids
```

INPUT : k: the number of clusters,

D: a data set containing n objects.

OUTPUT : A set of k clusters.

MATHEMATICAL MODEL :

Let S be the solution perspective .

S={s, e, i, o, f, DD, NDD, success, failure}

s = { Initial state of the 8-tile puzzel consisting of all tiles at particular places }

I = Input of the system → { I1, I2 }
where I1 = { no of clusters k}
I2 = { data set of 'n' objects }

o = Output of the system → { O1 }
where O1 = { k clusters }

f = Functions used → { f1}
where f1 = { k-means algorithm }

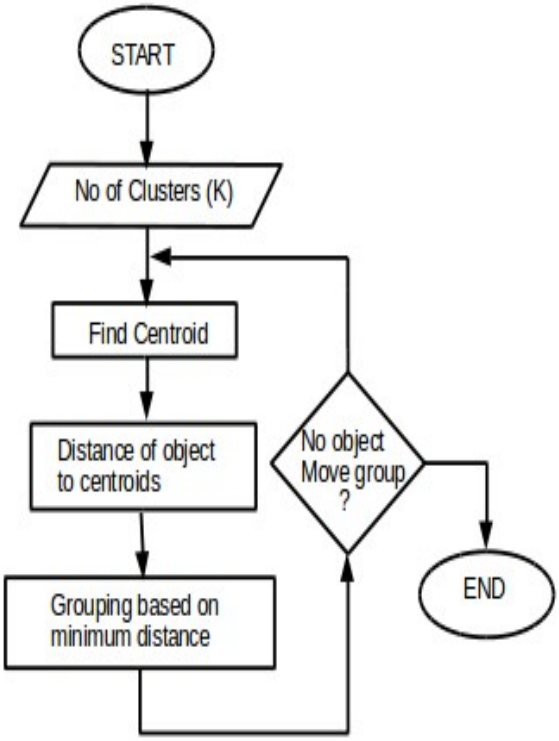
DD - Deterministic data → { }

NDD - Non deterministic data → { Cluster objects, size of the cluster}

Success - Desired outcome generated → { k clusters }

Failure - Desired outcome not generated or forced exit due to system error.

FLOWCHART :



TEST CASE :

rohan@rohan-HP-ProBook-4410s:~/Dropbox/7_SEMESTER$ python k-means-new.py

Initial Centroid 1 : [172, 45, 20, 105]

Initial Centroid 2 : [180, 65, 23, 160]

Initial Centroid 3 : [189, 69, 23, 143]

CLUSTER 1 : [[172, 160, 162, 128, 102, 103, 105, 120, 125, 134, 127, 129, 123, 124, 134, 200, 168, 165, 167, 160, 157, 159, 155, 139, 147, 178], [45, 73, 52, 98, 76, 74, 72, 79, 80, 82, 67, 65, 40, 45, 100, 89, 95, 73, 75, 79, 55, 56, 57, 58, 54, 53], [20, 12, 19, 19, 12, 11, 10, 10, 9, 8, 23, 34, 35, 56, 28, 29, 20, 27, 31, 30, 33, 42, 43, 47, 49, 65], [105, 123, 112, 108, 120, 110, 111, 112, 115, 114, 117, 119, 141, 132, 90, 89, 88, 100, 102, 109, 108, 123, 124, 126, 122, 121]]

CLUSTER 2 : [[175, 180, 145], [48, 65, 95], [21, 23, 17], [156, 160, 145]]

CLUSTER 3 : [[189, 169, 197, 172, 174, 172, 176, 190, 173, 165, 179], [69, 89, 67, 63, 71, 87, 88, 49, 50, 102, 60], [23, 21, 18, 35, 40, 45, 32, 78, 80, 25, 62], [143, 119, 147, 123, 124, 127, 140, 133, 134, 137, 130]]

--

New Centroid 1 : [143, 68, 27, 113]

New Centroid 2 : [166, 69, 20, 153]

New Centroid 3 : [177, 72, 41, 132]

CLUSTER 1 : [[172, 160, 162, 128, 102, 103, 105, 120, 125, 134, 127, 129, 123, 124, 134, 168, 165, 167, 160, 157, 155, 139, 147], [45, 73, 52, 98, 76, 74, 72, 79, 80, 82, 67, 65, 40, 45, 100, 95, 73, 75, 79, 55, 57, 58, 54], [20, 12, 19, 19, 12, 11, 10, 10, 9, 8, 23, 34, 35, 56, 28, 20, 27, 31, 30, 33, 43, 47, 49], [105, 123, 112, 108, 120, 110, 111, 112, 115, 114, 117, 119, 141, 132, 90, 88, 100, 102, 109, 108, 124, 126, 122]]

CLUSTER 2 : [[175, 180, 145, 197], [48, 65, 95, 67], [21, 23, 17, 18], [156, 160, 145, 147]]

CLUSTER 3 : [[189, 169, 172, 174, 172, 176, 190, 173, 165, 200, 159, 178, 179], [69, 89, 63, 71, 87, 88, 49, 50, 102, 89, 56, 53, 60], [23, 21, 35, 40, 45, 32, 78, 80, 25, 29, 42, 65, 62], [143, 119, 123, 124, 127, 140, 133, 134, 137, 89, 123, 121, 130]]

--

New Centroid 1 : [139, 69, 25, 113]

New Centroid 2 : [174, 68, 19, 152]

New Centroid 3 : [176, 71, 44, 126]

CLUSTER 1 : [[160, 162, 128, 102, 103, 105, 120, 125, 134, 127, 129, 123, 124, 134, 168, 165, 160, 157, 139, 147], [73, 52, 98, 76, 74, 72, 79, 80, 82, 67, 65, 40, 45, 100, 95, 73, 79, 55, 58, 54], [12, 19, 19, 12, 11, 10, 10, 9, 8, 23, 34, 35, 56, 28, 20, 27, 30, 33, 47, 49], [123, 112, 108, 120, 110, 111, 112, 115, 114, 117, 119, 141, 132, 90, 88, 100, 109, 108, 126, 122]]

CLUSTER 2 : [[175, 180, 189, 145, 197, 165], [48, 65, 69, 95, 67, 102], [21, 23, 23, 17, 18, 25], [156, 160, 143, 145, 147, 137]]

CLUSTER 3 : [[172, 169, 172, 174, 172, 176, 190, 173, 200, 167, 159, 155, 178, 179], [45, 89, 63, 71, 87, 88, 49, 50, 89, 75, 56, 57, 53, 60], [20, 21, 35, 40, 45, 32, 78, 80, 29, 31, 42, 43, 65, 62], [105, 119, 123, 124, 127, 140, 133, 134, 89, 102, 123, 124, 121, 130]]

-

New Centroid 1 : [135, 70, 24, 113]

New Centroid 2 : [175, 74, 21, 148]

New Centroid 3 : [174, 66, 44, 121]

CLUSTER 1 : [[160, 128, 102, 103, 105, 120, 125, 134, 127, 129, 123, 124, 134, 168, 139], [73, 98, 76, 74, 72, 79, 80, 82, 67, 65, 40, 45, 100, 95, 58], [12, 19, 12, 11, 10, 10, 9, 8, 23, 34, 35, 56, 28, 20, 47], [123, 108, 120, 110, 111, 112, 115, 114, 117, 119, 141, 132, 90, 88, 126]]

CLUSTER 2 : [[175, 180, 189, 145, 197, 176, 165], [48, 65, 69, 95, 67, 88, 102], [21, 23, 23, 17, 18, 32, 25], [156, 160, 143, 145, 147, 140, 137]]

CLUSTER 3 : [[172, 162, 169, 172, 174, 172, 190, 173, 200, 165, 167, 160, 157, 159, 155, 147, 178, 179], [45, 52, 89, 63, 71, 87, 49, 50, 89, 73, 75, 79, 55, 56, 57, 54, 53, 60], [20, 19, 21, 35, 40, 45, 78, 80, 29, 27, 31, 30, 33, 42, 43, 49, 65, 62], [105, 112, 119, 123, 124, 127, 133, 134, 89, 100, 102, 109, 108, 123, 124, 122, 121, 130]]

New Centroid 1 : [128, 73, 22, 115]

New Centroid 2 : [175, 76, 22, 146]

New Centroid 3 : [169, 64, 41, 116]

CLUSTER 1 : [[128, 102, 103, 105, 120, 125, 134, 127, 129, 123, 124, 134], [98, 76, 74, 72, 79, 80, 82, 67, 65, 40, 45, 100], [19, 12, 11, 10, 10, 9, 8, 23, 34, 35, 56, 28], [108, 120, 110, 111, 112, 115, 114, 117, 119, 141, 132, 90]]

CLUSTER 2 : [[175, 180, 189, 160, 169, 145, 197, 176, 165], [48, 65, 69, 73, 89, 95, 67, 88, 102], [21, 23, 23, 12, 21, 17, 18, 32, 25], [156, 160, 143, 123, 119, 145, 147, 140, 137]]

CLUSTER 3 : [[172, 162, 172, 174, 172, 190, 173, 200, 168, 165, 167, 160, 157, 159, 155, 139, 147, 178, 179], [45, 52, 63, 71, 87, 49, 50, 89, 95, 73, 75, 79, 55, 56, 57, 58, 54, 53, 60], [20, 19, 35, 40, 45, 78, 80, 29, 20, 27, 31, 30, 33, 42, 43, 47, 49, 65, 62], [105, 112, 123, 124, 127, 133, 134, 89,

88, 100, 102, 109, 108, 123, 124, 126, 122, 121, 130]]

New Centroid 1 : [121, 73, 21, 115]

New Centroid 2 : [172, 77, 21, 141]

New Centroid 3 : [167, 64, 41, 115]

CLUSTER 1 : [[128, 102, 103, 105, 120, 125, 134, 127, 129, 123, 124, 134], [98, 76, 74, 72, 79, 80, 82, 67, 65, 40, 45, 100], [19, 12, 11, 10, 10, 9, 8, 23, 34, 35, 56, 28], [108, 120, 110, 111, 112, 115, 114, 117, 119, 141, 132, 90]]

CLUSTER 2 : [[175, 180, 189, 160, 169, 145, 197, 176, 165], [48, 65, 69, 73, 89, 95, 67, 88, 102], [21, 23, 23, 12, 21, 17, 18, 32, 25], [156, 160, 143, 123, 119, 145, 147, 140, 137]]

CLUSTER 3 : [[172, 162, 172, 174, 172, 190, 173, 200, 168, 165, 167, 160, 157, 159, 155, 139, 147, 178, 179], [45, 52, 63, 71, 87, 49, 50, 89, 95, 73, 75, 79, 55, 56, 57, 58, 54, 53, 60], [20, 19, 35, 40, 45, 78, 80, 29, 20, 27, 31, 30, 33, 42, 43, 47, 49, 65, 62], [105, 112, 123, 124, 127, 133, 134, 89, 88, 100, 102, 109, 108, 123, 124, 126, 122, 121, 130]]

TIME AND SPACE COMPLEXITY :

Time Complexity of the algorithm:

To analyze the complexity of the algorithm in terms of time required, we need to identify the operations required in each step and in each iteration of the algorithm. Here, for k-means algorithm, the operations for each iteration are:

- Calculation of distances: To calculate the distance from a point to the centroid, we can use the squared Euclidean proximity function. For this, we need two subtractions, one summation, two multiplications and one square - root operations, i.e., 6-operations.
- Comparisons between distances
- Calculation of centroids

So, the total number of operations for an iteration

= distance calculation operations + comparison operations + centroids calculation operations

= 6*[k*m*n] operations + [(k-1)*m*n] operations + [k*((m-1) + 1)*n] operations

Assume that the algorithm converges after I iterations, then total number of operations for the algorithm
= 6*[I*k*m*n] operations + [I*(k-1)*m*n] operations + [I*k*((m-1) + 1)*n] operations
= [6Ikmn + I(k-1)mn + Ikmn] operations
≈ O (I*k*m*n)

Here, we've applied the basic assumption of each operation requiring the same amount of time. Since, normally, k << m & n << m, the analysis shows that the k-means is linear in m, the number of points, and is scalable and efficient in processing large data sets.

Space Complexity of the Algorithm:

The space requirement for k-means are modest because only the data points and centroids are needed to be stored. Specifically, the storage required is O((m+k)n), where m is the number of objects (points) & n is the number of attributes considering n-dimensional objects.

CONCLUSION

Hence we have successfully implemented the k-means algorithm to perform clustering of children based on the data given.

COURSE OUTCOMES :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	
To solve problems for multicore or distributed,concurrent/Parallel environments	√

FAQs

1. What is clustering?
2. What is differnce between k-means and k-nearest neighbour algortihm?
3. What is euclidean distance and cosine similarity?
4. What are drawbacks of k-means algorithm?

ASSIGNMENT NO-EL-II-B1

PROBLEM STATEMENT :

In a rolling display program of news display on a smart TV or Computer Display the input strings are supplied by another computer connected through wireless networks.Develop necessary app using Scala/Python/ Java/ C++.

OBJECTIVE :

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

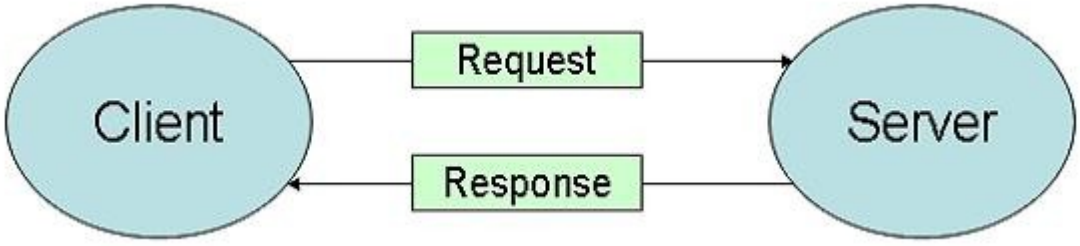
THEORY :

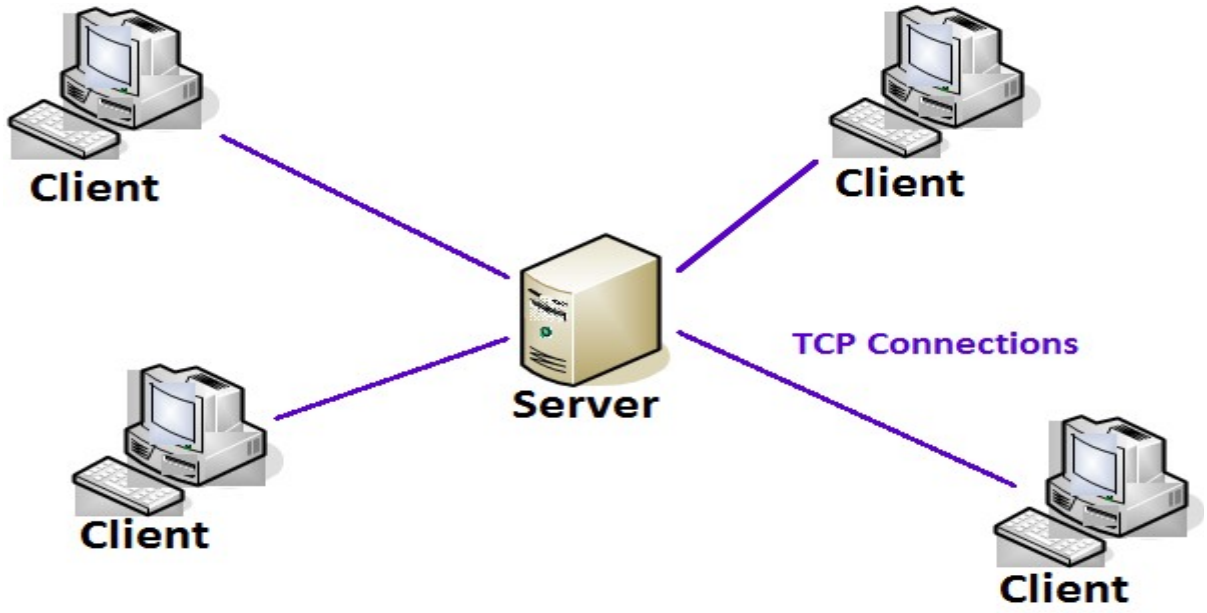
Client – Server Architecture :

Client/server architecture is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to a central server over a network or Internet connection. Client/server architecture may also be referred to as a networking computing model because all the requests and services are delivered over a network.

Client/server architecture is a producer-consumer computing architecture where the server acts as the producer and the client as a consumer. The server houses and provides high-end, computing-intensive services to the client on demand. These services can include applications access, storage, file sharing, printer access and/or direct access to the server's raw computing power.

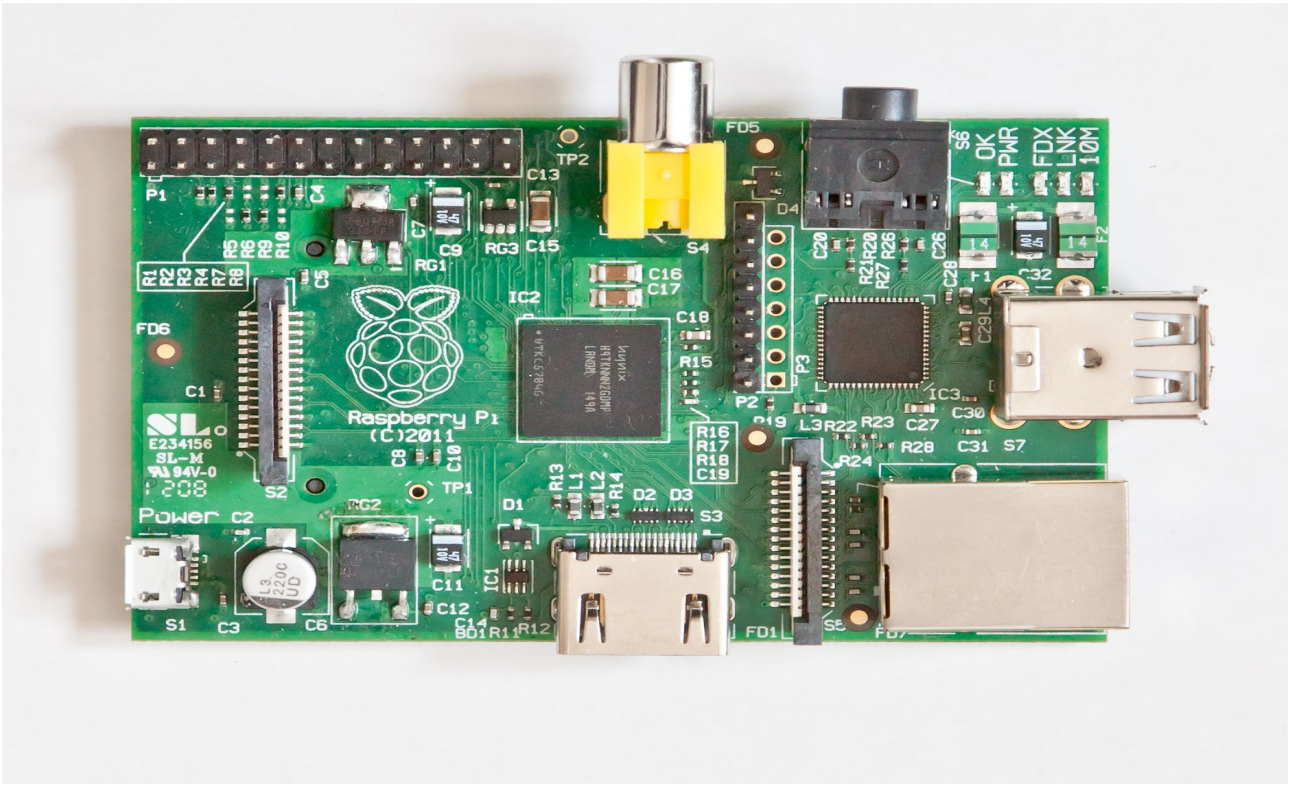
This architecture works when the client computer sends a resource or process request to the server over the network connection, which is then processed and delivered to the client. A server computer can manage several clients simultaneously, whereas one client can be connected to several servers at a time, each providing a different set of services. In its simplest form, the Internet is also based on client/server architecture where the Web server serves many simultaneous users with Web page and or website data.





RASPBERRY PI :

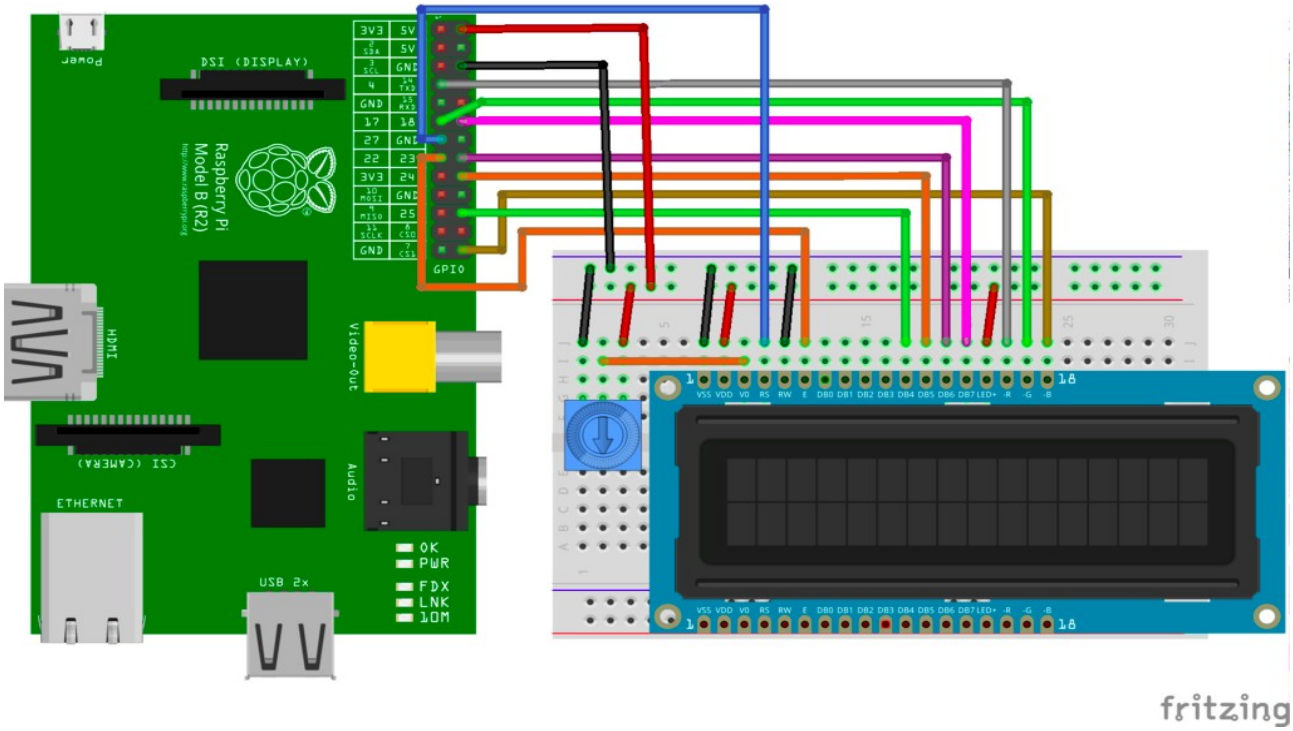
The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games**.** What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras.



Raspberry PI Hardware General Specifications :

Feature	Model A	Model B	Model A+	Model B+
BRCM2835 SoC	Yes	Yes	Yes	Yes
Standard SoC Speed	700Mhz	700Mhz	700Mhz	700Mhz
RAM	256MB	512MB*	256MB	512MB
Storage	Full SD	Full SD	Micro SD	Micro SD
Ethernet 10/100	No	Yes	No	Yes
HDMI output port	Yes	Yes	Yes	Yes
Composite video output	Yes	Yes	On 3.5mm jack	On 3.5mm jack
Number of USB2.0 ports	1	2	1	4
Expansion header	26	26	40	40

INTERFACING DIAGRAM :



ALGORITHM :

1. Create a Server on Raspberry Pi .
2. Create a Client on another PC . The PC and Raspberry Pi are connected to each other wirelessly
3. Enter an Input string from the PC which sends it to the server .
4. The server receives the string and gives it to the LCD .
5. The input string is displayed in a rolling fashion on the LCD .

INPUT :

Enter input string from a PC .

EXPECTED OUTPUT :

Rolling display of the input string on the LCD .

MATHEMATICAL MODEL :

Let P be the solution perspective .

P ={ S , E , I , O, F }

S = { Initial state of the system consisting of the server listening for connections }

I = Input of the system → { I1 }

where I1 = { Input String }

O = Output of the system → { O1 }
where O1 = { Rolling display on the LCD }

F = Functions used → { f1 , f2 , f3 ,f4 , f5 }
where f1 = { listen() : Listen for incoming connections }
f2 = { connect() : Connect to the server }
f3 = { bind() : Bind a host to a particular port }
f4 = { send() : Send a mesage }
f5 = { receive() : Receive a mesage }

E = End state of the system showing rolling dislay of the input string on LCD on a wirelessly connected Raspberry Pi.

CONCLUSION :

Hence we have successfully implemented a rolling display program by sending the string from a wirelessly connected PC to another display device .

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	√
To solve problems for multicore or distributed,concurrent/Parallel environments	

ASSIGNMENT NO-C1

1. Problem Statement: Study and implementation of research paper in Multidisciplinary NLP using open source tool

2. Objective:

To develop problem solving abilities for smart devices.
To develop problem solving abilities of pervasiveness, embedded security and NLP.
To study algorithmic examples in distributed, concurrent and parallel environments

3. Theory:

Natural Language Processing:

NLP is strongly concerned with the interactions between computer and human languages.
GOAL:
To get computers to perform useful tasks involving human language.

Tasks:

1. Enabling human-machine communication,
2. Improving human-human communication and
3. Useful processing of text or speech

OpenNLP:

The Apache OpenNLP library is a machine learning based toolkit for processing of natural language text. It includes a sentence detector, a tokenizer, a name finder, a parts-of-speech (POS) tagger, a chunker, and a parser. It has very good APIs that can be easily integrated with a Java program. However, the documentation contains unupdated information.

Download Jar Files and Set Up Environment

Before starting the examples, you need to download the jar files required and add to your project build path. The jar files required are loaced at "apache-opennlp-1.5.3-bin.zip", the download page looks like the following:

File	Signatures
apache-opennlp-1.5.3-bin.tar.gz	md5 sha1 asc
apache-opennlp-1.5.3-bin.zip	md5 sha1 asc
apache-opennlp-1.5.3-src.tar.gz	md5 sha1 asc
apache-opennlp-1.5.3-src.zip	md5 sha1 asc

Unzip the .zip file and copy the 4 jar files in the "lib" directory to your project. In addition, you will need to download some model files later based on what you want to do.

1. Sentence Detector

Sentence detector is for detecting sentence boundaries. Given the following paragraph:
Hi. How are you? This is Mike.
sentence detector returns an array of strings. In this case, the array has two elements as below.
Hi. How are you?
This is Mike.

Example Code:

```
public static void SentenceDetect() throws InvalidFormatException,
```

```
                IOException {
        String paragraph = "Hi. How are you? This is Mike.";

        // always start with a model, a model is learned from training data
        InputStream is = new FileInputStream("en-sent.bin");
        SentenceModel model = new SentenceModel(is);
        SentenceDetectorME sdetector = new SentenceDetectorME(model);

        String sentences[] = sdetector.sentDetect(paragraph);

        System.out.println(sentences[0]);
        System.out.println(sentences[1]);
        is.close();
}
```

2. Tokenizer

Tokens are usually words which are separated by space, but there are exceptions. For example, "isn't" gets split into "is" and "n't, since it is a a brief format of "is not". Our sentence is separated into the following tokens:

```
Hi
.
How
are
you
?
This
is
Mike
.
```

Example Code:

```
public static void Tokenize() throws InvalidFormatException, IOException {
        InputStream is = new FileInputStream("en-token.bin");

        TokenizerModel model = new TokenizerModel(is);

        Tokenizer tokenizer = new TokenizerME(model);

        String tokens[] = tokenizer.tokenize("Hi. How are you? This is Mike.");

        for (String a : tokens)
                System.out.println(a);

        is.close();
}
```

3. Name Finder

By its name, name finder just finds names in the context. Check out the following example to see what name finder can do. It accepts an array of strings, and find the names inside.

Example Code:

```
public static void findName() throws IOException {
        InputStream is = new FileInputStream("en-ner-person.bin");

        TokenNameFinderModel model = new TokenNameFinderModel(is);
        is.close();

        NameFinderME nameFinder = new NameFinderME(model);

        String []sentence = new String[]{
                    "Mike",
                    "Smith",
                    "is",
                    "a",
                    "good",
                    "person"
                    };

                Span nameSpans[] = nameFinder.find(sentence);

                for(Span s: nameSpans)
                        System.out.println(s.toString());
}
```

4. POS Tagger

Hi._NNP How_WRB are_VBP you?_JJ This_DT is_VBZ Mike._NNP

Example Code:

```
public static void POSTag() throws IOException {
        POSModel model = new POSModelLoader()
                .load(new File("en-pos-maxent.bin"));
        PerformanceMonitor perfMon = new PerformanceMonitor(System.err, "sent");
        POSTaggerME tagger = new POSTaggerME(model);

        String input = "Hi. How are you? This is Mike.";
        ObjectStream<String> lineStream = new PlainTextByLineStream(
                        new StringReader(input));

        perfMon.start();
        String line;
        while ((line = lineStream.read()) != null) {

                String whitespaceTokenizerLine[] = WhitespaceTokenizer.INSTANCE
                                .tokenize(line);
                String[] tags = tagger.tag(whitespaceTokenizerLine);

                POSSample sample = new POSSample(whitespaceTokenizerLine, tags);
                System.out.println(sample.toString());
```

```
            perfMon.incrementCounter();
        }
        perfMon.stopAndPrintFinalResult();
}
```

5. Chunker

Chunker may not be a concern for some users, but it is worth to mention it here. What chunker does is to partition a sentence to a set of chunks by using the tokens generated by tokenizer.
I don't have a good example to show why chunker is very useful, but you can replace the sentence with your own sentences in the following example code, and try to generate something you like.

Example Code:

```
public static void chunk() throws IOException {
        POSModel model = new POSModelLoader()
                        .load(new File("en-pos-maxent.bin"));
        PerformanceMonitor perfMon = new PerformanceMonitor(System.err, "sent");
        POSTaggerME tagger = new POSTaggerME(model);

        String input = "Hi. How are you? This is Mike.";
        ObjectStream<String> lineStream = new PlainTextByLineStream(
                        new StringReader(input));

        perfMon.start();
        String line;
        String whitespaceTokenizerLine[] = null;

        String[] tags = null;
        while ((line = lineStream.read()) != null) {
                whitespaceTokenizerLine = WhitespaceTokenizer.INSTANCE
                                .tokenize(line);
                tags = tagger.tag(whitespaceTokenizerLine);

                POSSample sample = new POSSample(whitespaceTokenizerLine, tags);
                System.out.println(sample.toString());
                        perfMon.incrementCounter();
        }
        perfMon.stopAndPrintFinalResult();

        // chunker
        InputStream is = new FileInputStream("en-chunker.bin");
        ChunkerModel cModel = new ChunkerModel(is);

        ChunkerME chunkerME = new ChunkerME(cModel);
        String result[] = chunkerME.chunk(whitespaceTokenizerLine, tags);

        for (String s : result)
                System.out.println(s);
```

```
        Span[] span = chunkerME.chunkAsSpans(whitespaceTokenizerLine, tags);
        for (Span s : span)
                System.out.println(s.toString());
}
```

6. Parser

Given this sentence: "Programcreek is a very huge and useful website.", parser can return the following:

```
(TOP
        (S
                (NP
                        (NN Programcreek)
                )
                (VP
                        (VBZ is)
                        (NP
                                (DT a)
                                (ADJP
                                        (RB very)
                                        (JJ huge)
                                        (CC and)
                                        (JJ useful)
                                )
                        )
                )
                (. website.)
        )
)
```

Example Code:

```
public static void Parse() throws InvalidFormatException, IOException {

        InputStream is = new FileInputStream("en-parser-chunking.bin");

        ParserModel model = new ParserModel(is);

        Parser parser = ParserFactory.create(model);

        String sentence = "Programcreek is a very huge and useful website.";
        Parse topParses[] = ParserTool.parseLine(sentence, parser, 1);

        for (Parse p : topParses)
                p.show();

        is.close();

        }
```

4. Conclusion:

Hence we have studied and learnt implementation of an Open Source tool for natural language

processing.

5. Outcomes achieved (mark the outcomes achieved)

COURSE OUTCOME	ACHIEVED(√)
Problem solving abilities for smart devices.	
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness, embedded security and NLP.	
To solve problems for multicore or distributed, concurrent/Parallel environments	√

ASSIGNMENT NO-C2

Introduction to Smart Watch App Development with Tizen

Becoming a smart watch app developer is a great idea, but what if you are an HTML developer? Will you be able to realize your dream without switching to a completely different platform? Do you have to abandon all of your HTML skills and start all over again? Don't worry. Tizen comes to your rescue.

Tizen is an operating system of the Linux family, targeting a range of devices from smartphones to smart watches and a lot more. While Tizen is a project within the Linux Foundation, it is guided by the Tizen Association, whose members include Samsung, Intel, and other well known companies in the technology industry.

In this tutorial, I will show you how to install and configure the Tizen SDK for Wearable and develop a smart watch application with the IDE. Let's get started.

Installing & Configuring the SDK

Step 1: Tizen SDK or Tizen SDK for Wearable?

Currently, two types of SDKs are available, Tizen SDK and Tizen SDK for Wearable. Since this tutorial is about developing a standalone smart watch app, what you need is Tizen SDK for Wearable.

You can download it from the Tizen Developers website. You need to download an appropriate **install manager** that matches your operating system and version. If you prefer an offline installation to an online one, you need to download an **SDK image** too. If your operating system is Windows 8 or Windows 8.1, you can download the installation files categorized under Windows 7. They will work on Windows 8 and 8.1 just fine.

Step 2: Requirements

Refer to Tizen's detailed instructions to read about the hardware and software requirements your computer should meet.

You can install the SDK even if your computer doesn't meet these hardware requirements. However, if you do, the smart watch emulator will be slow, resulting in poor app testing. Visit the documentation for more details. It explains how to enable **Virtualization Technology** (VT) in your BIOS and **Data Execution Prevention** on Windows.

Step 3: Installing the SDK

1. Run the **install manager** you downloaded earlier. This is an .exe file with a filename like tizen-wearable-sdk-2.2.159_windows64.exe, depending on your operating system and version.
2. Click **Advanced** to go to the next screen.
3. In that screen, check the **SDK image** radio button and navigate to the zip file containing the appropriate SDK Image. Note that I'm assuming that you prefer an offline installation and you have already downloaded the necessary SDK image to your development machine.
4. Select the **SDK image** zip file and click **Open** in the dialog box.
5. An **Extracting SDK Image** message will appear. Click **OK** after finishing the extraction.
6. Click **Next** and the **License Agreement** window will appear.
7. Agree to the license and click **Next.**
8. The **configuration window** will appear next. I recommend leaving all the check boxes checked and clicking **Next**.
9. Finally, when the **install manager** asks for a location for the installation, specify your choice by selecting a path and clicking **Install.**
10. If you have already configured your BIOS correctly, **Intel Hardware Accelerated Execution Manager** (Intel HAXM) will also be automatically installed during this process. If not, quit the installation process and configure the BIOS so that it can allow Intel HAXM to be installed.
11. Don't forget to enable **Data Execution Prevention** if your operating system is Windows. Resume the installation.

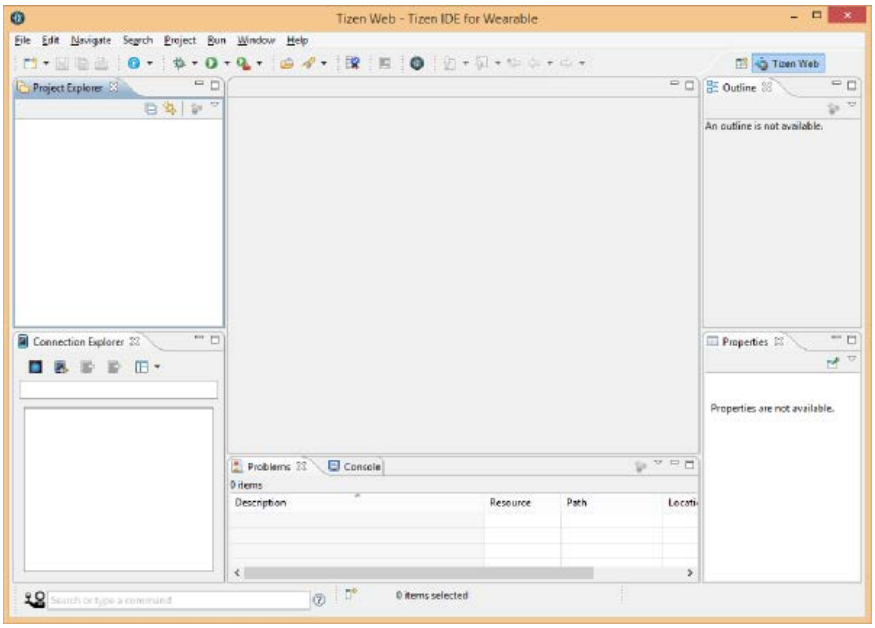
If you wish, you can install **Intel HAXM** separately. Once the installation is finished, restart your computer.

Step 4: Configuring the IDE

1. Browse to the folder in which you have installed the SDK and navigate to the **ide** subfolder. Run the executable file named **IDE**.

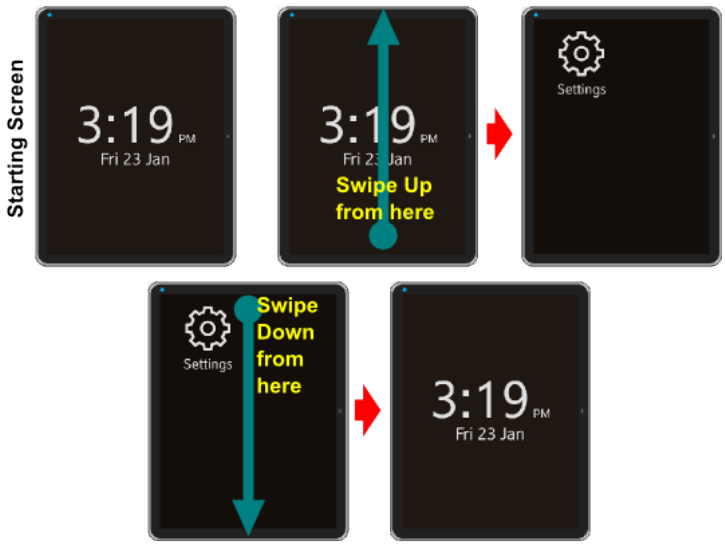
2. After a few minutes, a window will appear, asking for a location for the **workspace** to save the apps you develop. Specify a path of your choice for the location and click **OK**. After the configuration, the IDE should appear.

Step 5: Features of the IDE



On the left pane of the IDE, there are two windows, **Project Explorer** and **Connection Explorer**. The **Project Explorer** shows the projects created by the user. The **Connection Explorer** lists the connected devices that are currently available, emulator instances or remote devices.

Step 6: Creating an Emulator Instance



1. In the **Connection Explorer,** click on the **Emulator Manager** icon, the leftmost blue button.
2. Click **Yes** in the **User Account Control** window that appears. This will bring up the **Emulator Manager** window.
3. Click **Add New** and give the emulator instance a name.
4. When you click **Confirm,** the new emulator instance will be created. Click the blue button with an arrow in the emulator icon to launch the emulator.

It will take some time to launch the emulator. You should see a window with a starting screen similar to the below screenshot when it is up and running. The emulator instance should appear as an entry in the **Connection Explorer**.

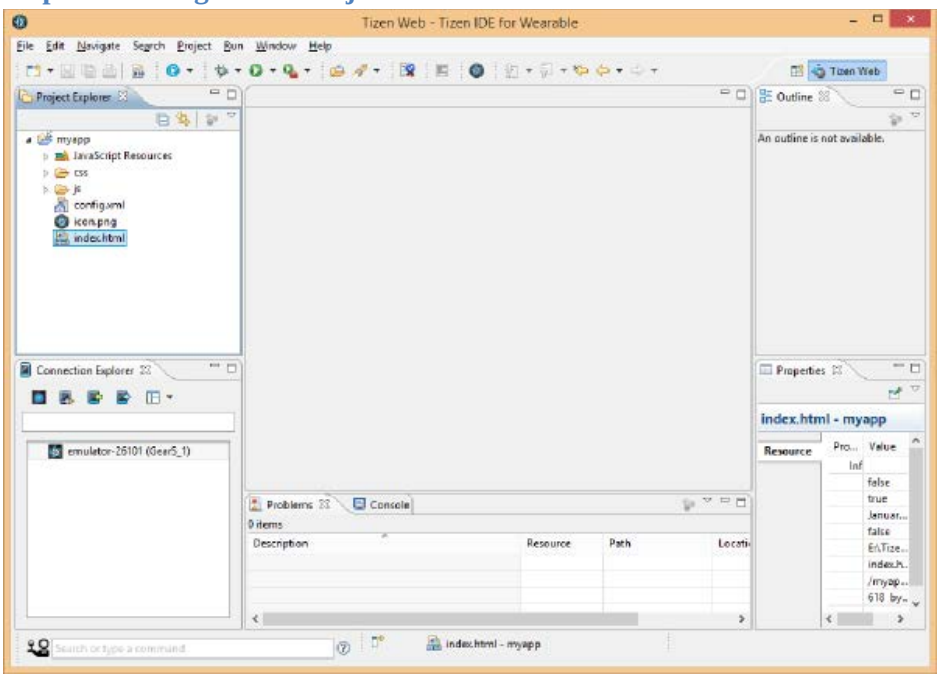
Swipe up from the bottom middle point of the starting screen to go to the screen that displays the installed apps on the device or emulator. Since you haven't installed any apps yet, only the **Settings** icon is displayed.

You can go back to the previous screen or quit an app by swiping down from the top middle of the screen.

Developing a Simple Comic App

In this example, we're going to create a simple app to display a comic strip. Let's look at each step in turn.

Step 1: Creating a New Project



Let's create a new project in the IDE.

1. Go to **File > New > Tizen Wearable Web Project**.

2. In the window that appears, select **Basic > Basic application** and set the **Project name** to **myapp**.
3. Tick the **Use default location** check box or browse to a different location of your choice, and click **Finish**.
4. Your new project, **myapp**, should appear in the **Project Explorer**.
5. Click the small arrow on the left of **myapp** to expand the project structure.
6. You should see an **index.html** file, a **css** subfolder, a **js** subfolder, and a few other files and folders.

HTML, CSS, and JavaScript form the basis for programming on the Tizen platform. If you're an HTML wizard, then you don't have to learn a new programming language to write applications for the Tizen platform. You can use your existing HTML, CSS, and JavaScript skills.

Step 2: Adding Files, Assets, and Resources

We first need to add two subfolders to the **myapp** project, **comic** and **images**. To do this, right-click the **myapp** project folder in the IDE and select **New > Folder**. The subfolders should appear in the expanded **myapp** folder in the IDE.

Download the source files for this project from GitHub and navigate to the **images** subfolder, which contains a number of png files. Copy the png files to the **images** subfolder you created a moment ago.

You can paste files to the **images** subfolder in the **Project Explorer** window by right-clicking the subfolder and selecting **Paste** from the popup menu.

Next, create nine HTML files with the following file names in the **comic** subfolder by right-clicking the **comic** subfolder and selecting **New > File**. Make sure to include the **.html** extension for the files.

- **page1.html**
- **page2.html**
- **...**
- **page9.html**

You should now have nine HTML files in the **comic** subfolder.

Step 3: Adding Code

Let's now edit the code in **index.html**. This file is the entry point of your application. Double-click **index.html** to open the file in the IDE. Change the contents of the <title> tag to <title>2nd Race</title>. Next, change the contents of the <body> tag with the following:

```
<body>
  <div>
    <div><img src="images/coverpage.png" alt="Cover Page" /></div>
              <div><a href="#" class="btn" >&lt;&lt;</a><a href="comic/page1.html"
              class="btn" >&gt;&gt;</a></div>
```

```
  </div>
</body>
```

All we did, is add an image to the page and two buttons to navigate to the other pages since our comic will have ten pages. Once you have made these changes, save the file by selecting **File > Save** from the menu.

Next, double-click **style.css** in the **css** subfolder and change its contents as shown below.

```
* {
    font-family: Verdana, Lucida Sans, Arial, Helvetica, sans-serif;
}

body {
    margin: 0px auto;
    background-color:#0a3003;
}

img {
  margin: 0;
  padding: 0;
  border: 0;
  width: 100%;
  height: auto;
  display: block;
  float: left;
}

.btn {
  display: inline-block;
  padding: 15px 4% 15px 4%;
  margin-left: 1%;
  margin-right: 1%;
  margin-top: 5px;
  margin-bottom: 5px;
  font-size: 30px;
  text-align: center;
  vertical-align: middle;
  border: 0;
  color: #ffffff;
  background-color: #4b4237;
  width: 40%;
  height: 80px;
  text-decoration: none;
}
```

We've added some styling for body, images, and the navigation menus. Once you have made the changes, save the file.

Similarly, add code to all the other HTML files you have created. The **style.css** file in the **css** subfolder must be externally linked to all of these HTML files. If you're note sure about this step, then download the source files from GitHub and examine the source for clarification.

Step 4: Testing the App

To test your app, select the **myapp** project folder and, from the menu, choose **Project > Build Project** to build the project. Make sure the emulator instance is up and running.

Right-click the **myapp** folder and select **Run As > Tizen Wearable Web Application** to run the project in the emulator. Use the arrow buttons in the user interface to navigate to the next or previous page. Swipe down from the top of the screen to quit the app.

Conclusion

In this tutorial, we built a simple comic app with the Tizen SDK for Wearable and ran it on the smart watch emulator that comes bundled with the IDE. The real fun begins when you are ready to install and run it on a physical device.