Unit-3

**23.What is Decision Table**

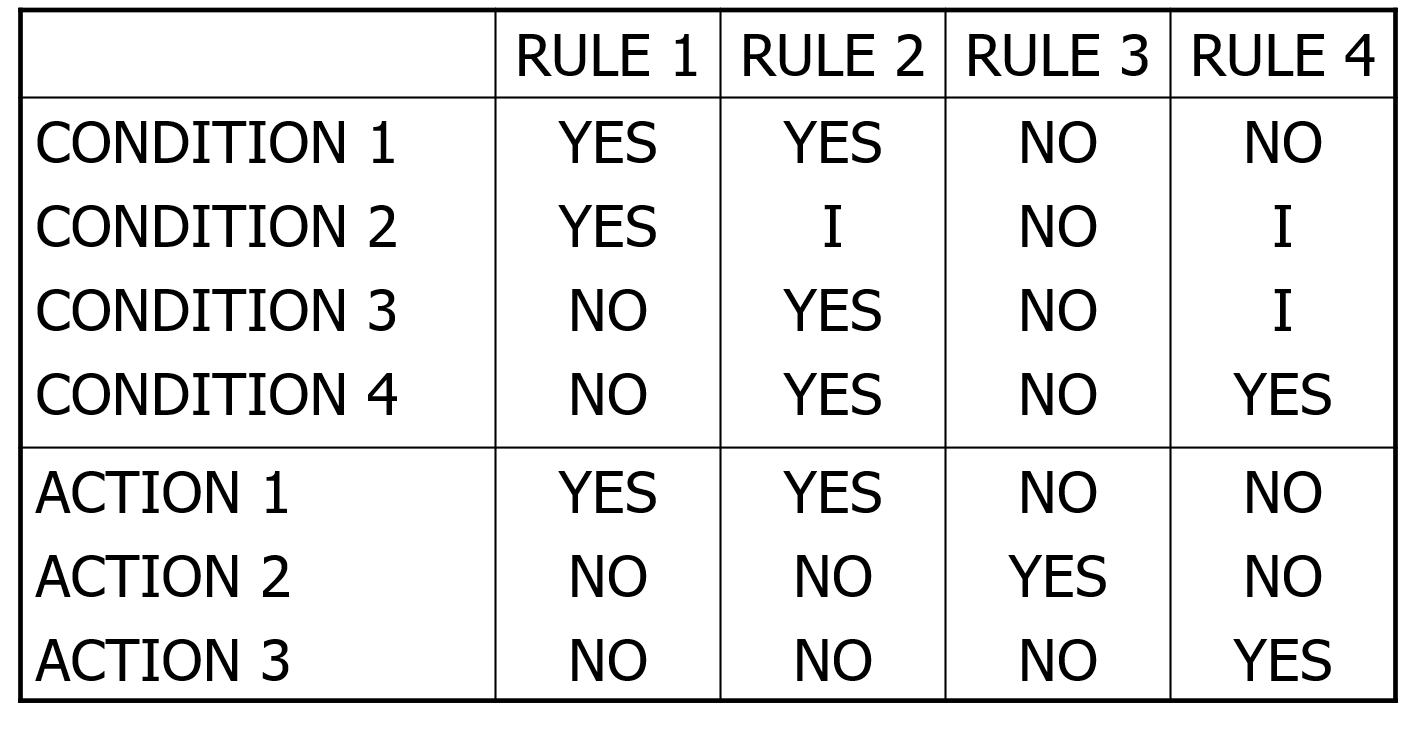
A **decision table** is a tabular method for representing and analyzing decision logic. It includes:

* **Condition Stub**: Lists the input conditions.
* **Condition Entries**: Shows true (YES), false (NO), or immaterial (I) for each condition under each rule.
* **Action Stub**: Lists the possible actions.
* **Action Entries**: Shows whether the action is taken (YES) or not (NO) for each rule.

**Uses:**

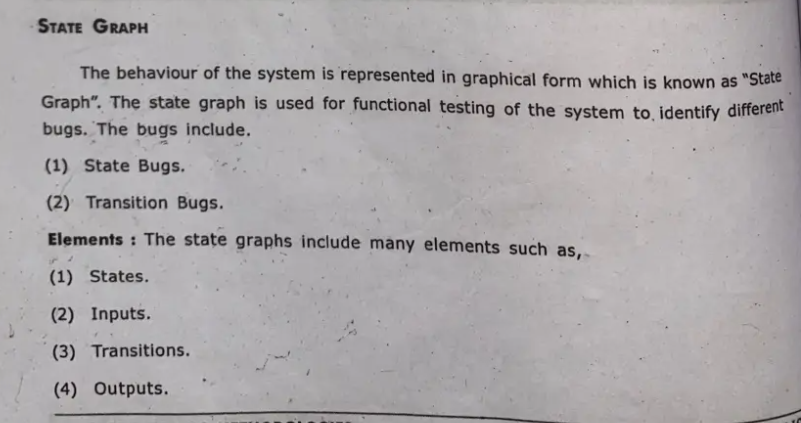
* Systematically represent complex decision logic.
* Serve as a basis for designing test cases.
* Help verify completeness and consistency.

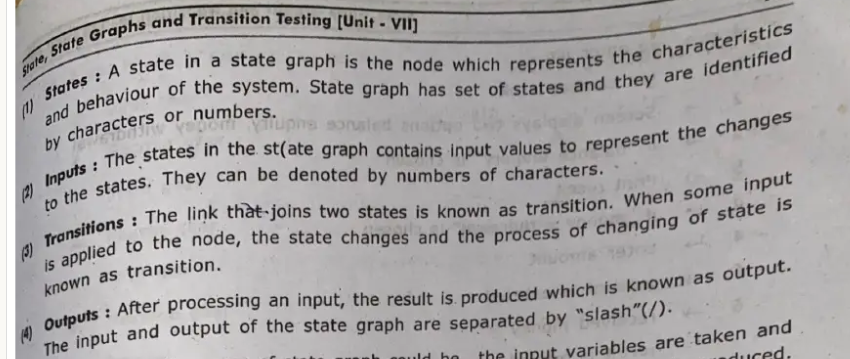
**Example**

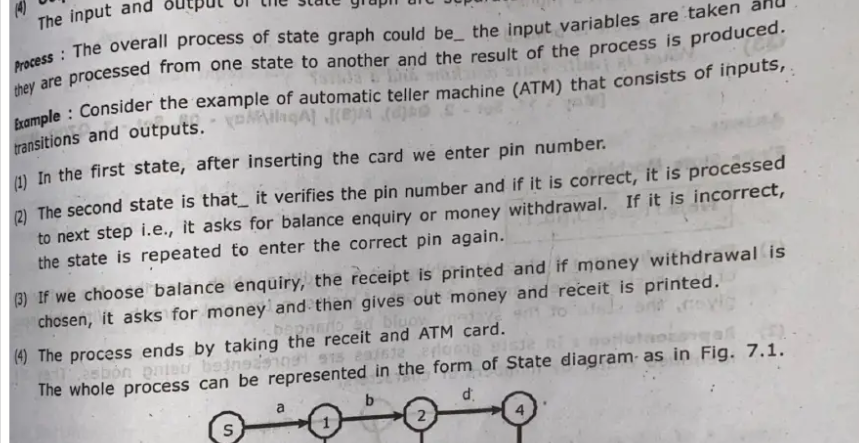


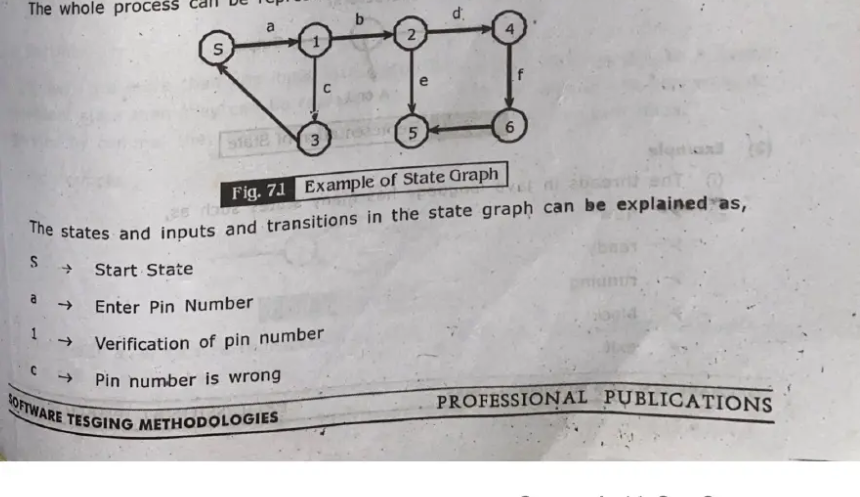
Unit-4 -🡪 8 marks

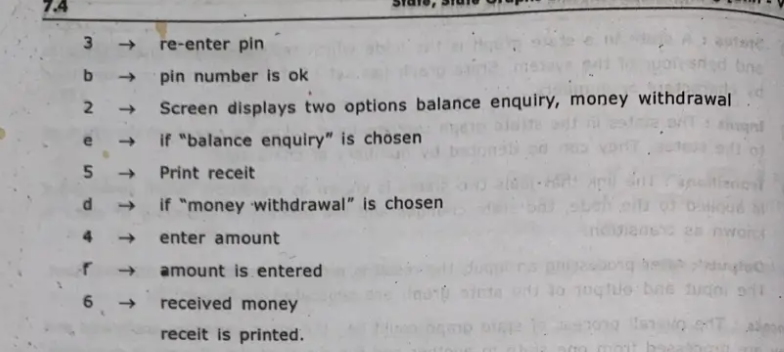
**31. Can you explain the process of determining number of states in a state graph?**











**32. How to identify equivalent states and how to merge them in representing state graph? Explain.**

**✅ What is a State Graph?**

A **state graph** (or state transition diagram) represents:

* **States** of the system (nodes)
* **Transitions** between states (edges) triggered by events or inputs.

**🎯 What are Equivalent States?**

Two states are said to be **equivalent** if:

* They respond to the **same inputs** in the **same way**, and
* Lead to **equivalent next states** under the same conditions.

👉 In other words, the behavior of the system is **identical** regardless of which state you are in.

**🔍 How to Identify Equivalent States?**

1. **Analyze Input-Output Behavior:**
   * Check how each state responds to every input.
   * Compare outputs for those inputs.
2. **Check Transition Paths:**
   * Compare the transitions (next states) from both states for the same input.
   * Ensure the next states are also equivalent.
3. **Use Partitioning Method (if needed):**
   * Initially group all states.
   * Iteratively split groups based on behavior differences until stable groups remain.

**🔄 How to Merge Equivalent States?**

Once two or more equivalent states are identified:

1. **Create a new single state** representing all equivalent ones.
2. **Redirect all incoming transitions** from the original states to the new merged state.
3. **Update outgoing transitions** so they originate from the merged state.
4. **Delete redundant original states**.

### Example: ATM Pin Verification System

#### 💡 System Description:

* A user has 3 attempts to enter the correct PIN.
* If the correct PIN is entered, go to **Access Granted** state.
* After 3 incorrect attempts, go to **Card Blocked** state.

### 🧩 States:

* **S0**: Start
* **S1**: 1st incorrect attempt
* **S2**: 2nd incorrect attempt
* **S3**: 3rd incorrect attempt → Blocked
* **S4**: Correct PIN → Access Granted

### 🧪 Transitions:

* From **S0**, wrong PIN → **S1**
* From **S1**, wrong PIN → **S2**
* From **S2**, wrong PIN → **S3** (Card Blocked)
* From **S0**, **S1**, or **S2**, correct PIN → **S4**

### ✅ Identifying Equivalent States:

Observe that:

* From **S0**, **S1**, and **S2**:
  + On correct PIN → **S4**
  + On incorrect PIN → next incorrect attempt state

The **output behavior is different**:

* From **S0**, an incorrect PIN leads to S1.
* From **S1**, an incorrect PIN leads to S2.
* From **S2**, it leads to **S3** (Blocked).

👉 These states are **not equivalent** because they lead to **different next states** on the same input.

### 🔁 But Now Consider:

Let’s say you have two states:

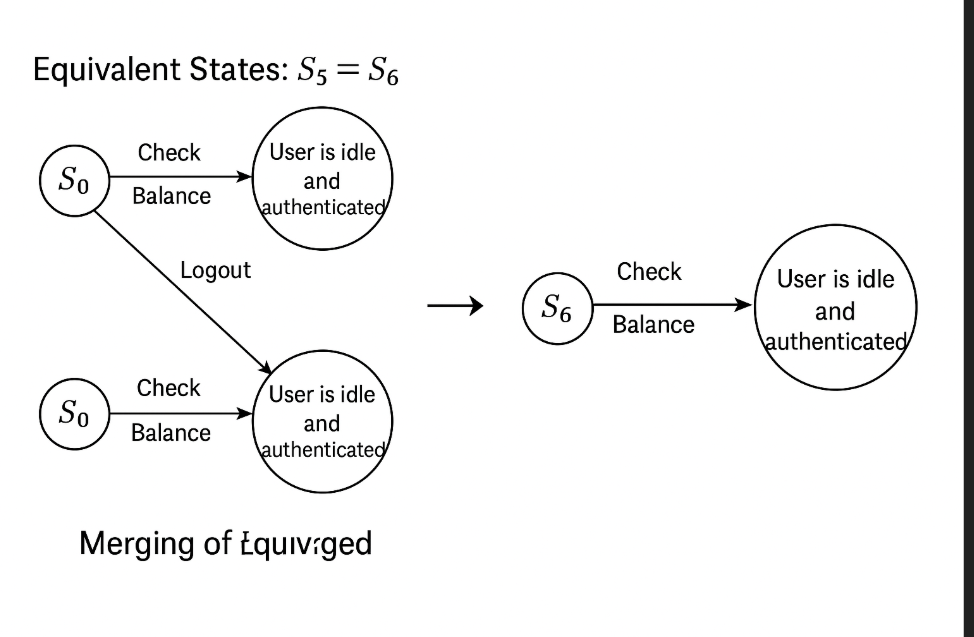
* **S5** and **S6**, both representing "User is idle and authenticated"
* From both, on action **"Check Balance"** → **Balance State**
* On **Logout** → **S0**

If all inputs in **S5** and **S6** result in the same outputs and transitions, then:

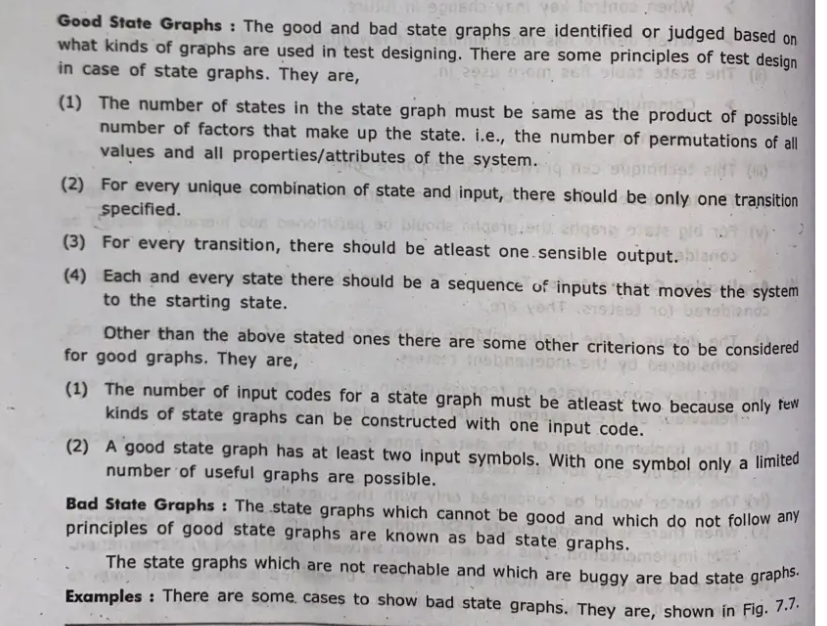
✅ **S5 ≡ S6** (Equivalent)

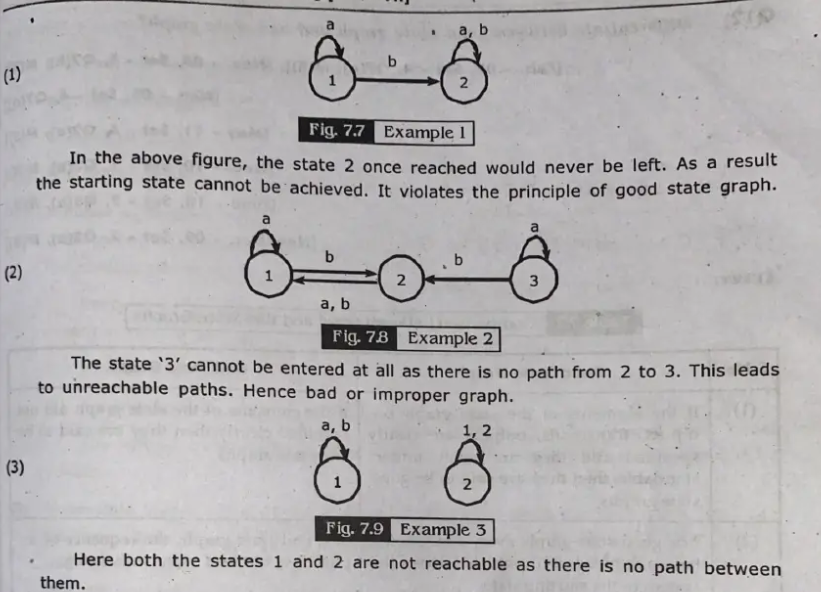
### 🔄 Merging:

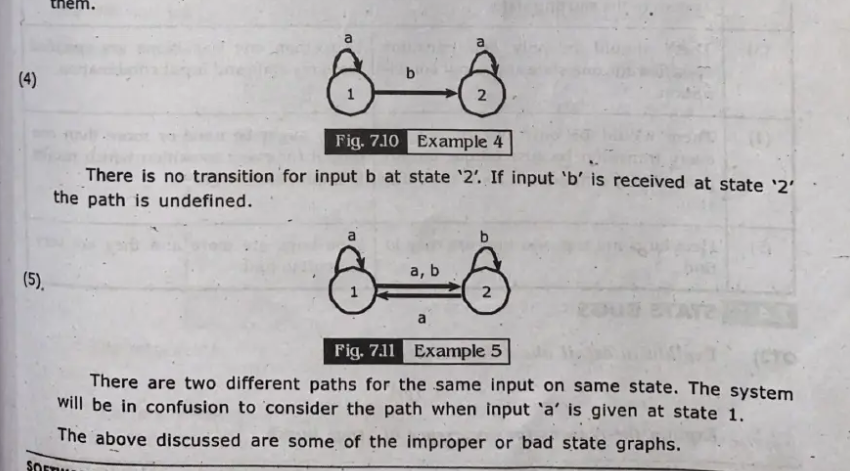
* Replace S5 and S6 with a new state **S5\_6**
* Redirect all transitions to/from S5 and S6 to/from **S5\_6**

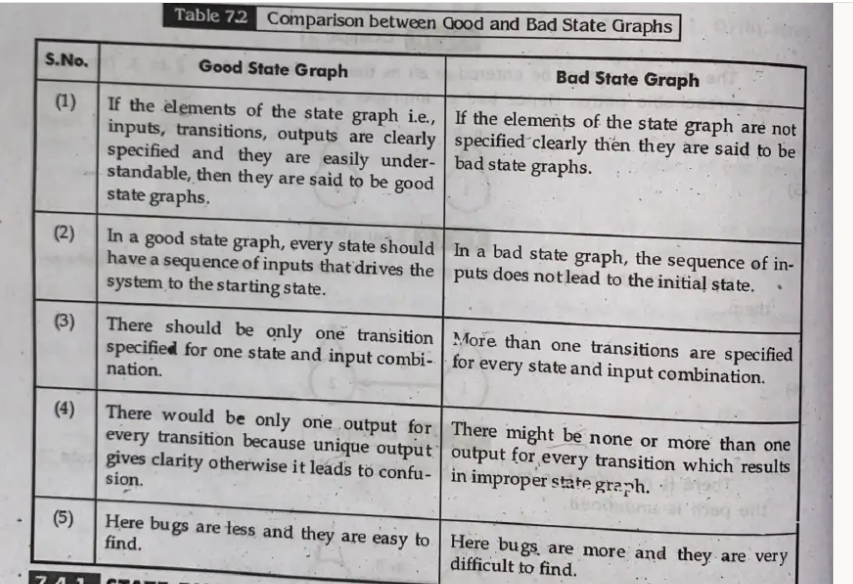
****

**33 Explain in detail about good and bad state graphs**

****

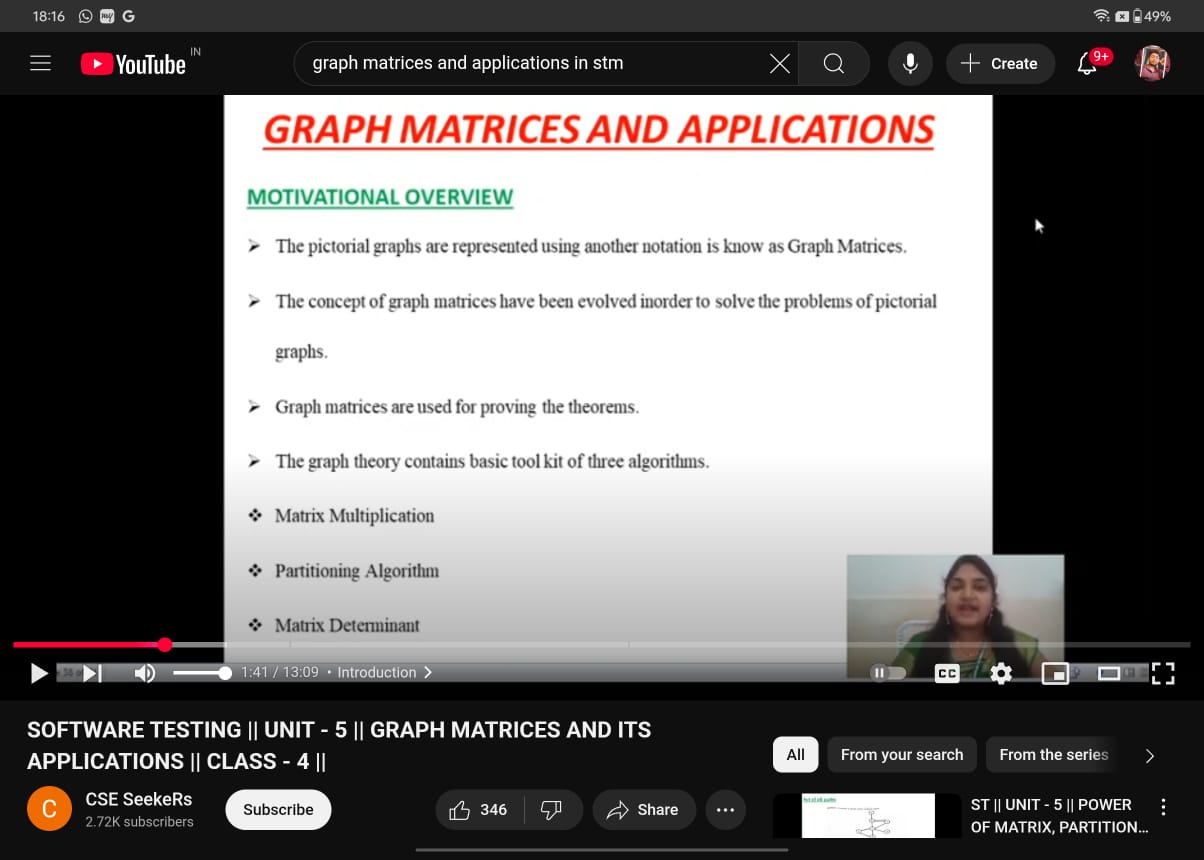
****

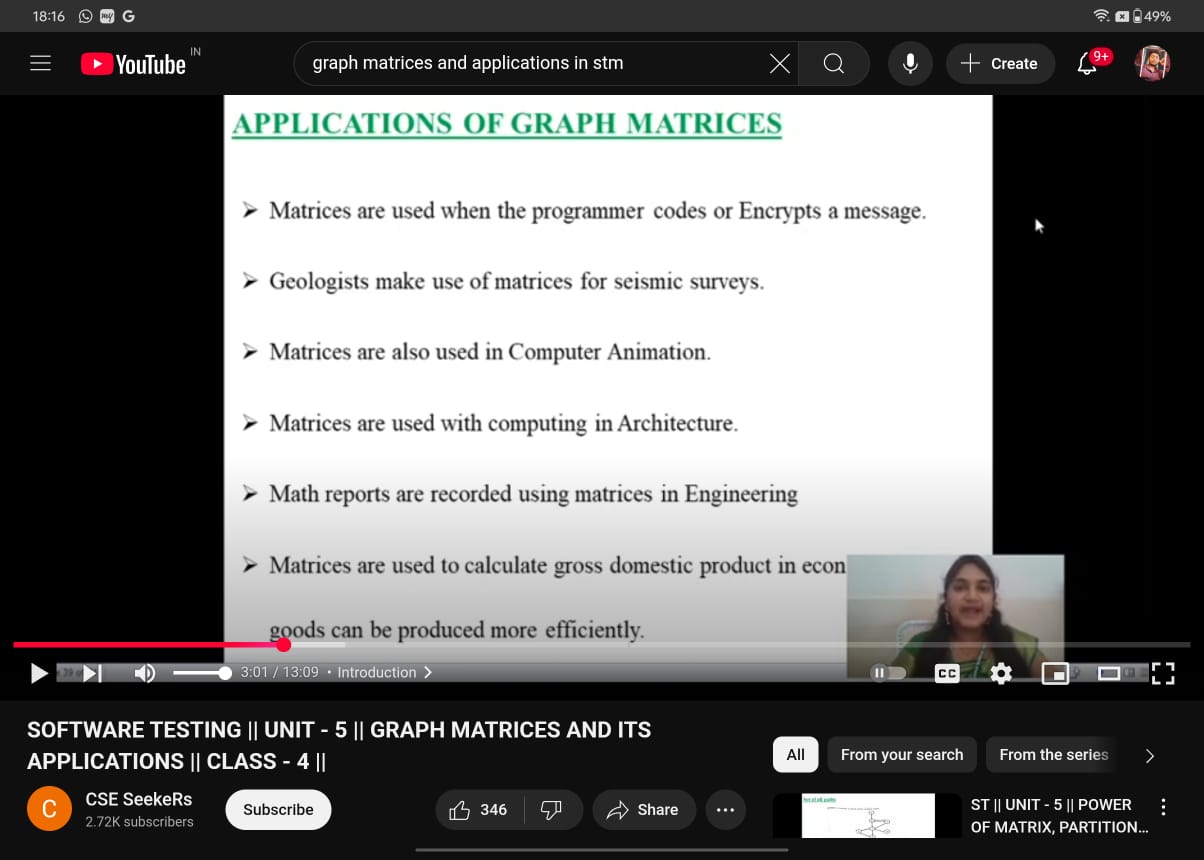
****

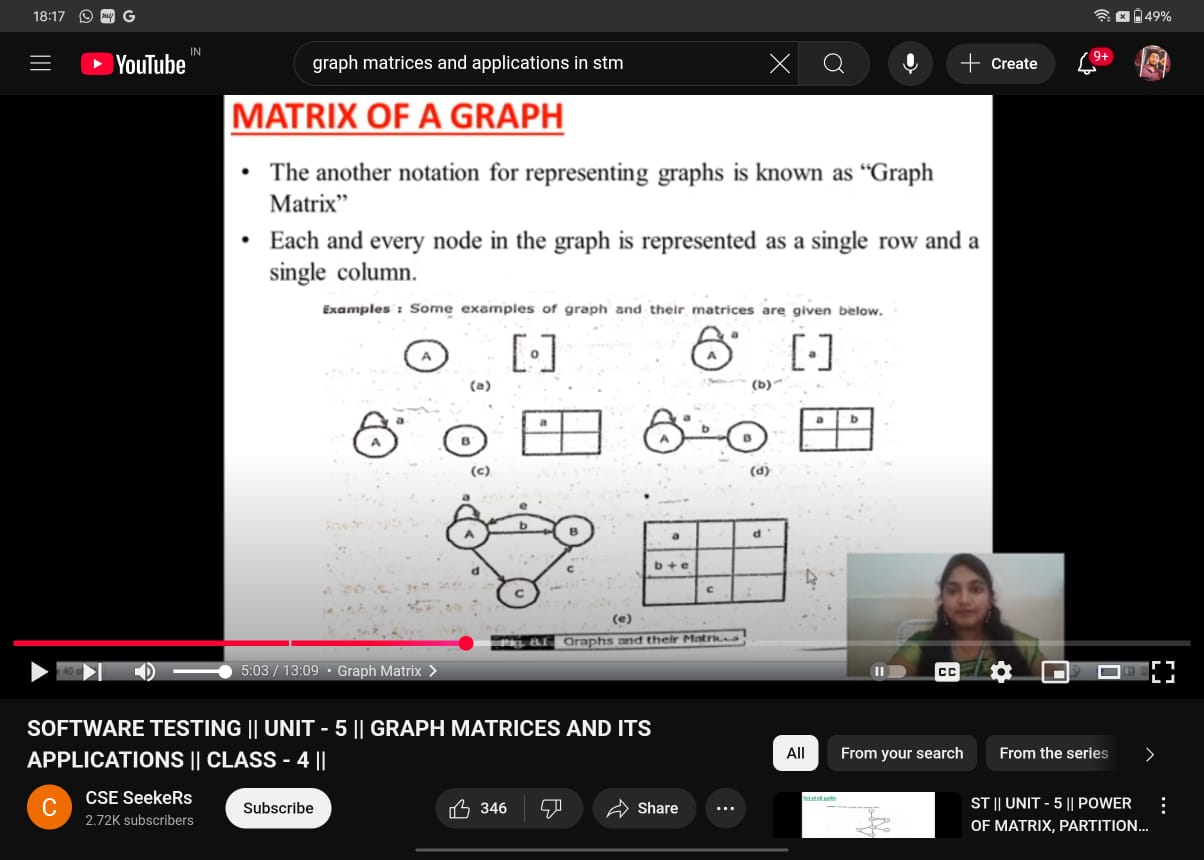
****

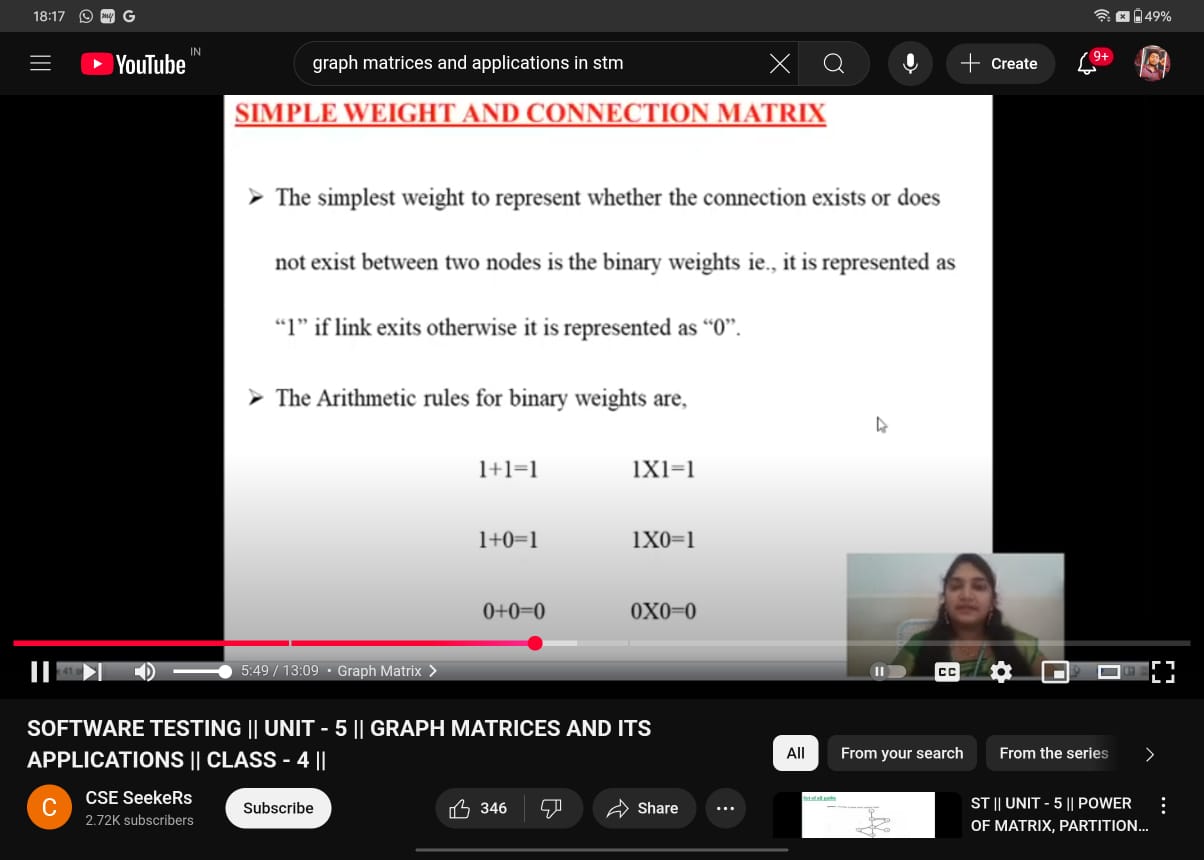
**Unit-5**

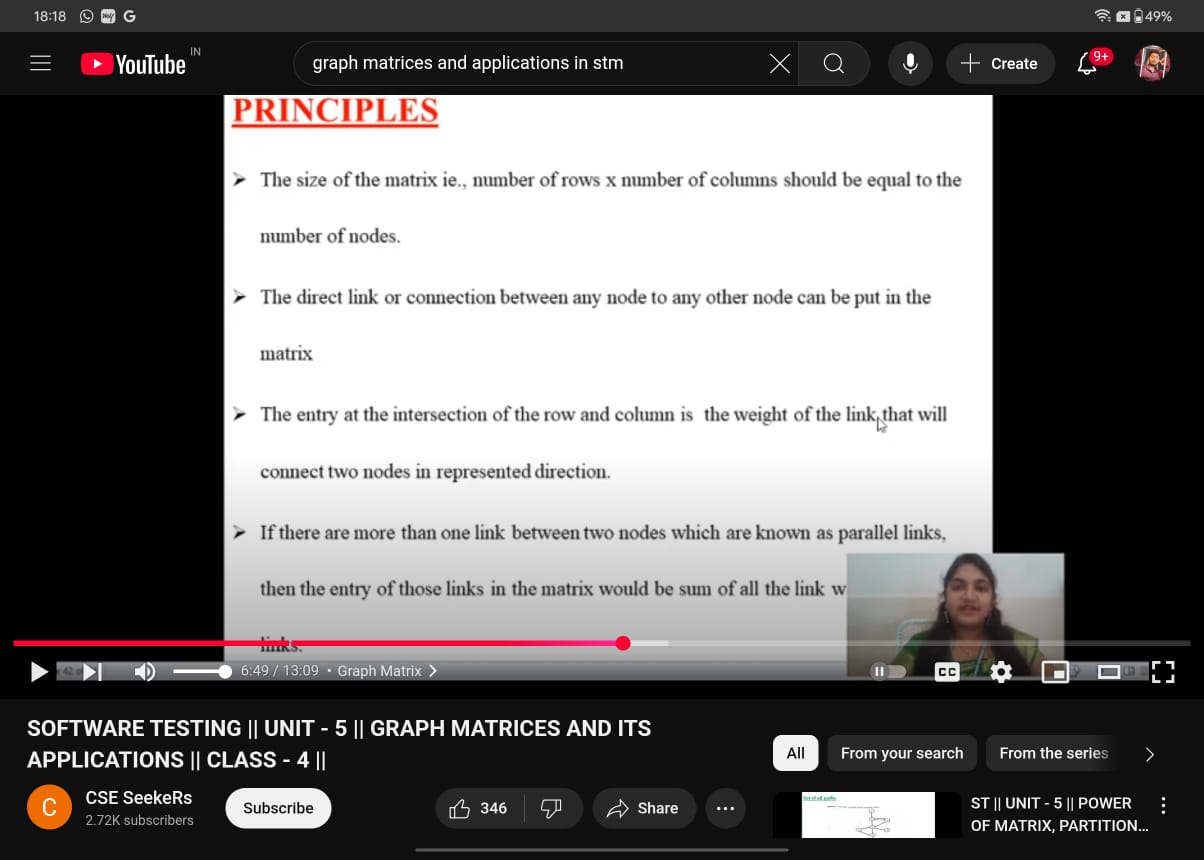
40. Explain Graph Matrices and applications

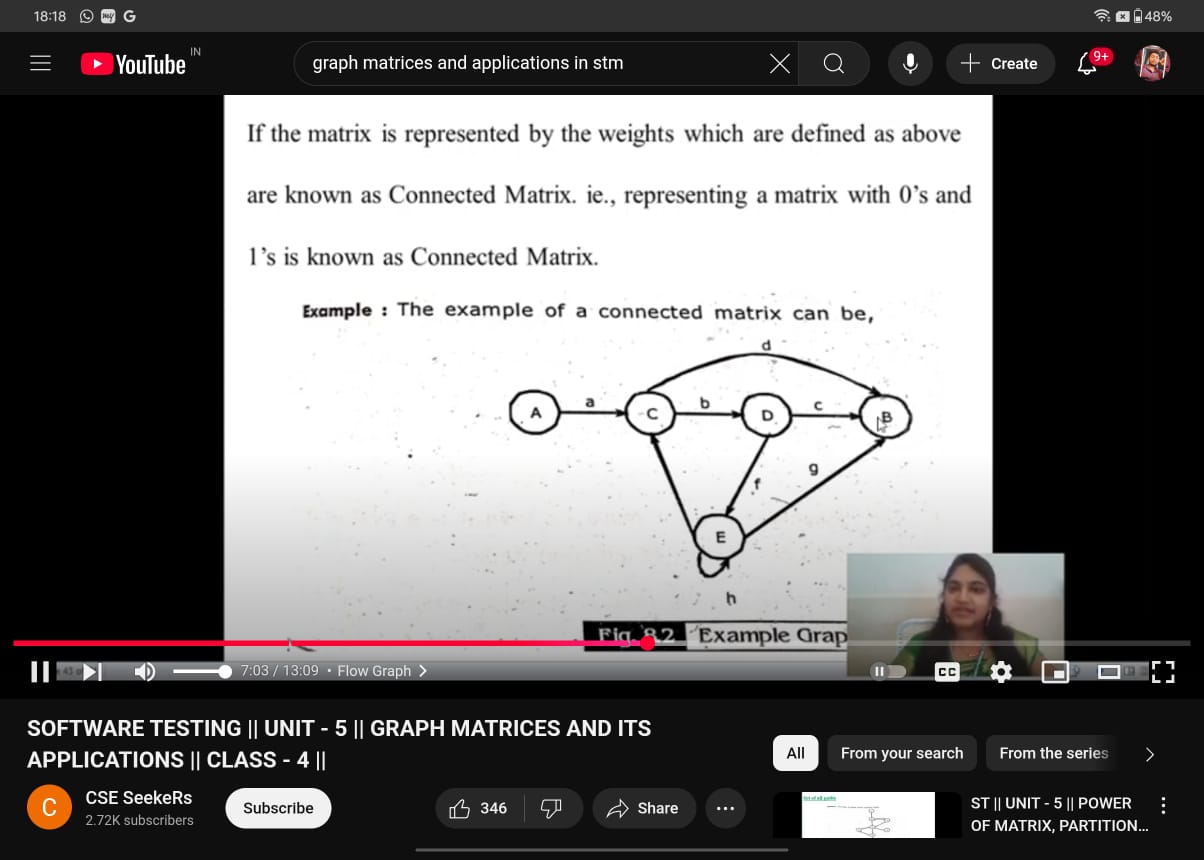
****

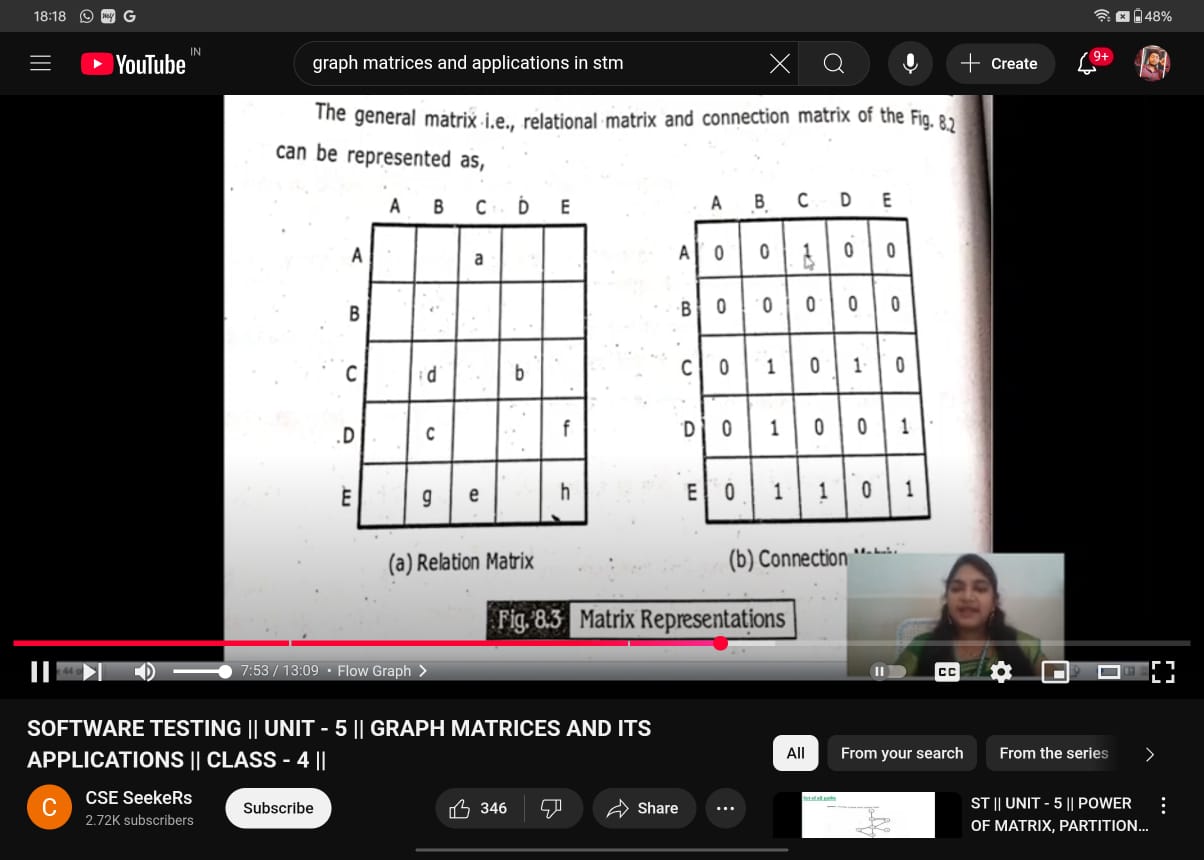
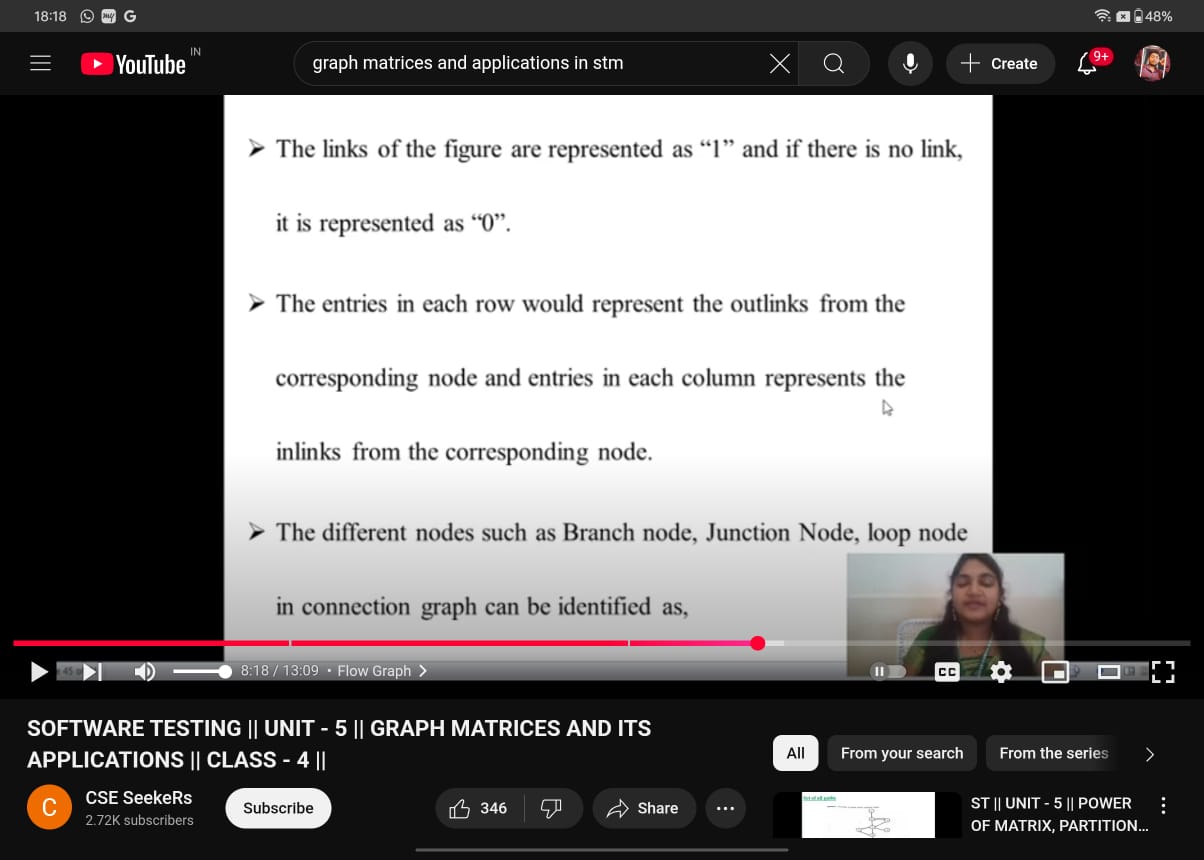
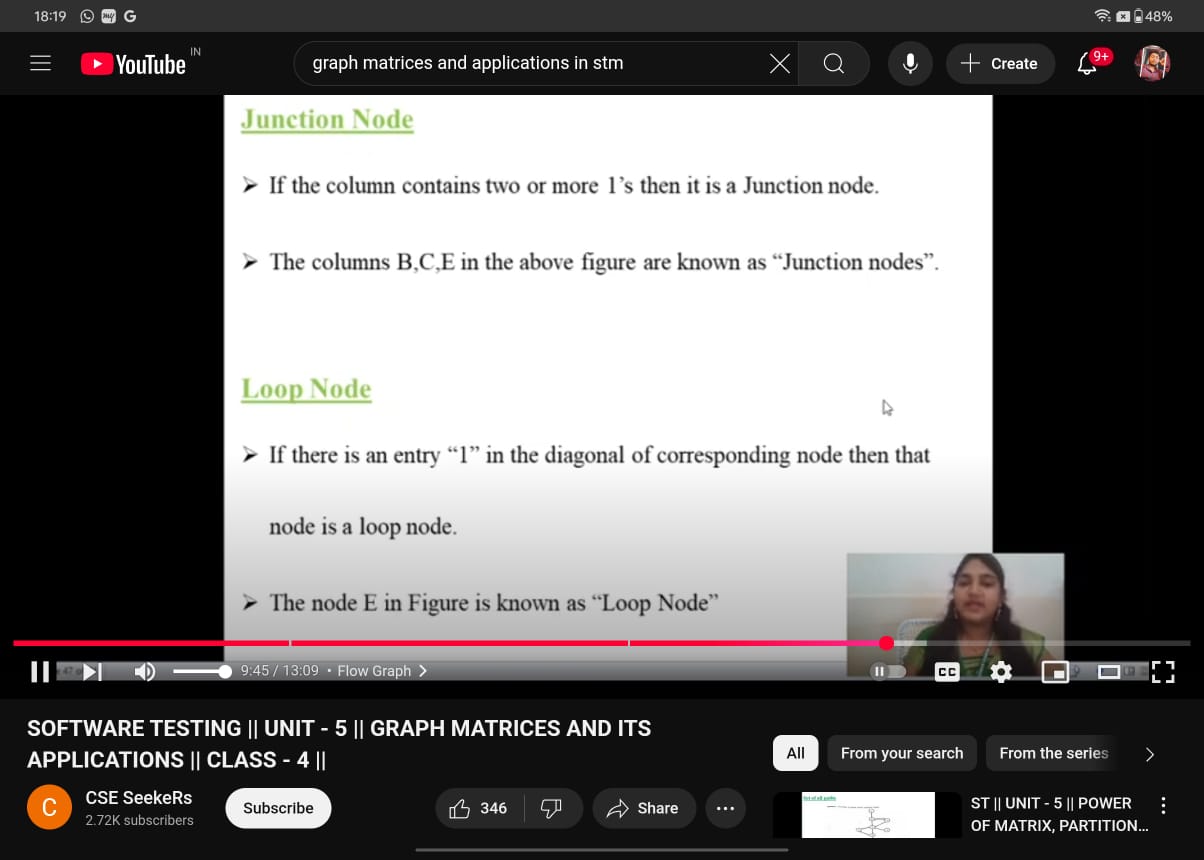
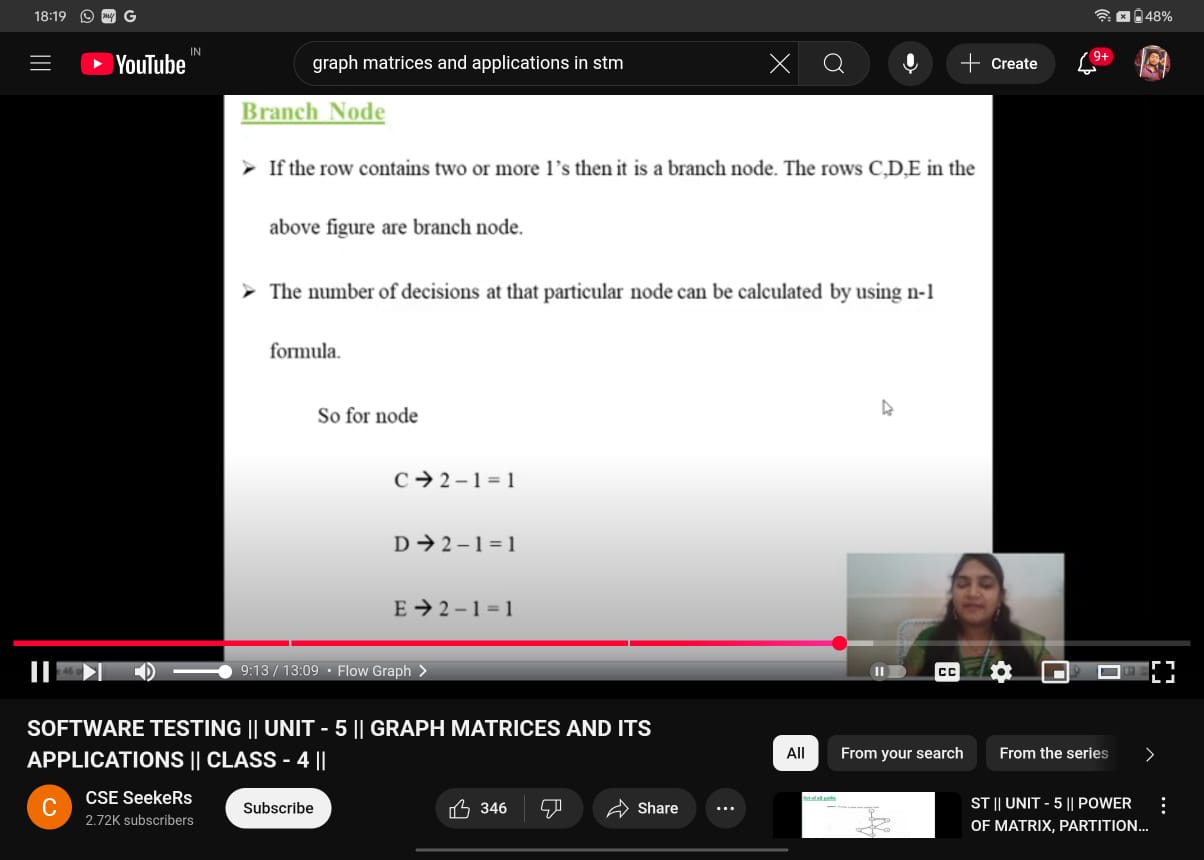
****

****

****

****

****

**** ****  

**42.** Write the usage and applications of Win runner tools in software testing

**✅ Usage and Applications of WinRunner in Software Testing**

**WinRunner** is an automated functional GUI testing tool developed by **Mercury Interactive** (now part of Micro Focus). It is widely used to perform **regression, functional, and data-driven testing** of software applications.

**🔧 Usage of WinRunner in Software Testing**

1. **Automated Functional Testing**
   * Automates GUI interactions like mouse clicks, keyboard inputs, form submissions, etc.
   * Helps in testing how the application behaves under various input conditions.
2. **Regression Testing**
   * Re-runs recorded test cases to ensure that recent code changes haven’t broken existing functionality.
3. **Data-Driven Testing**
   * Allows separation of test logic and test data.
   * Test scripts can run multiple times with different data sets (from Excel, databases, etc.).
4. **Test Script Language (TSL)**
   * Uses a scripting language called TSL (Test Script Language) for customizing test cases, inserting checkpoints, loops, and conditions.
5. **GUI Mapping and Testing**
   * Identifies and stores all GUI objects in an application in a **GUI map**.
   * Scripts use this map to interact with UI components during tests.
6. **Synchronization**
   * Ensures test scripts wait for the application to be ready (like loading a page or a button becoming enabled).
7. **Error Handling**
   * Supports recovery scenarios to handle unexpected errors or popups during test execution.

**📌 Applications of WinRunner**

1. **Banking and Financial Systems**
   * Automates repetitive test cases for transactional applications to ensure consistency and accuracy.
2. **ERP and CRM Testing**
   * Tests SAP, Oracle, and PeopleSoft applications for UI consistency and functional correctness.
3. **E-commerce Applications**
   * Validates workflows such as user login, cart operations, and payment processes.
4. **Desktop and Web-based Applications**
   * Used in testing both standalone applications and browser-based interfaces.
5. **Cross-version Compatibility Testing**
   * Ensures that older features continue working with newer releases (especially important in legacy systems).
6. **Product-based Companies**
   * Used to regularly test software builds to maintain stability and reduce manual testing overhead.

**🚫 Note:**

* **WinRunner is now obsolete** and has been officially replaced by **HP UFT (Unified Functional Testing)**, previously known as **QuickTest Professional (QTP)**.
* However, its concepts still form the foundation of modern GUI automation tools.