1. Short note on derivation tree with suitable examples.

Ans

A **derivation tree** or **parse tree** represents the structure of a string according to a grammar. Each internal node is labelled with a non-terminal, and each leaf node is a terminal.  
Example:  
Grammar:  
S → aSb | ε  
String: aabb  
Parse Tree:

S

/ | \

a S b

/ | \

a S b

|

ε

1. Explain rightmost derivation and left most derivation with an suitable examples.

Ans

* **Leftmost derivation**: Always expand the leftmost non-terminal first.
* **Rightmost derivation**: Always expand the rightmost non-terminal first.

Example for string aabb:  
Grammar: S → aSb | ε

Leftmost derivation:  
S ⇒ aSb ⇒ aaSbb ⇒ aabb

Rightmost derivation:  
S ⇒ aSb ⇒ aaSbb ⇒ aabb

1. Define ambiguous grammar.

Ans

A **grammar is ambiguous** if a string can have more than one leftmost or rightmost derivation or parse tree.  
Example:  
Grammar:  
S → SbS|a  
String: abababa

1. Let G be a CFG that generates the set of palindromes given by S→ a S a | b S b | a | b Find the PDA that accepts L(G).

Ans

* CFG for palindromes:
  + 
  + 
  + 
  + 

1. **Step 1** Initial state :
   * On input or , push it onto the stack and remain in .
     + 
     + 
2. **Step 2** Transition to state (guessing the middle):
   * On input or or , transition to without changing the stack.
     + 
     + 
     + 
3. **Step 3** State (matching phase):
   * If the input symbol matches the top of the stack, pop the stack and remain in .
     + 
     + 
4. **Step 4** Accepting state :
   * If the input is empty and the stack is empty, transition to the accepting state .
     + 

Solution

The PDA accepts the language of palindromes by pushing the first half of the input onto the stack, then comparing the second half with the stack content, and accepting if the stack is empty at the end.

1. Define Parse Tree with suitable example.

Ans

A **parse tree** shows how a start symbol derives a string based on the grammar rules.

Example:  
Grammar: S → aSb | ε  
String: ab  
Parse Tree:

S

/|\

a S b

|

ε

1. Briefly Explain PDA with example.
2. Briefly Explain DPDA with example.
3. List applications of CFG.

Ans

* Design of programming languages
* Syntax analysis in compilers
* Natural Language Processing (NLP)
* Generation of artificial languages

1. Different between leftmost and Right derivation with example.

Ans

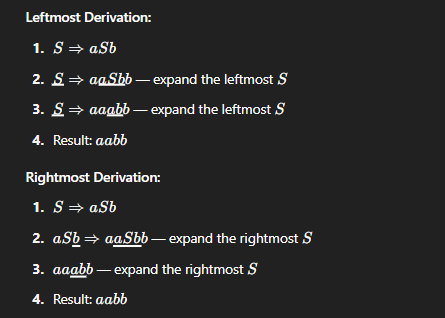
| **Aspect** | **Leftmost Derivation** | **Rightmost Derivation** |
| --- | --- | --- |
| **Definition** | Always expand the **leftmost** non-terminal first | Always expand the **rightmost** non-terminal first |
| **Order of expansion** | Left to right | Right to left |
| **Use** | Helpful in constructing **parse trees**, especially **top-down parsers** | Used in **bottom-up parsers** and in constructing **reverse parse trees** |

**Example:**

Grammar GGG:

* S→aSb ∣ ab

Let's derive the string aabb.



1. Write closure properties of CFL.

Ans

CFLs are closed under:

* **Union**: If L₁ and L₂ are CFLs, then L₁ ∪ L₂ is also a CFL.
* **Concatenation**: If L₁ and L₂ are CFLs, then L₁L₂ is a CFL.
* **Kleene Star**: If L is a CFL, then L\* is also a CFL.
* **Substitution**: CFLs are closed under substitution where each symbol is replaced by a CFL.

1. Write Decision properties of CFL.

Ans

Decision problems that are **decidable** for CFLs include:

* **Emptiness**: Checking if L(G) = ∅ for a CFG G.
* **Finiteness**: Checking if L(G) is finite.
* **Membership**: Given a string w and CFG G, checking if w ∈ L(G).
* **Equivalence with Regular Languages**: Given a CFG G and a Regular Language R, check if L(G) = R.

1. Define CNF with suitable.

Ans

A **Context-Free Grammar (CFG)** is in **Chomsky Normal Form (CNF)** if every production is of the form:

* A → BC (where A, B, and C are non-terminals)
* A → a (where a is a terminal)
* A → ε (only if allowed for the start symbol)

**Example:** Given grammar:

* S → ASA | aB
* A → B | S
* B → b | ε

After converting to CNF:

* S → AB | BC
* A → BA | SC
* B → b
* C → a

1. Define GNF with suitable.

Ans

A **Context-Free Grammar** is in **Greibach Normal Form (GNF)** if every production is of the form:

* A → aα (where a is a terminal and α is a string of non-terminals)

**Example:** Grammar in GNF:

* S → aA | bB
* A → aS | a
* B → bS | b

Here, every production starts with a **terminal**, followed by non-terminals.

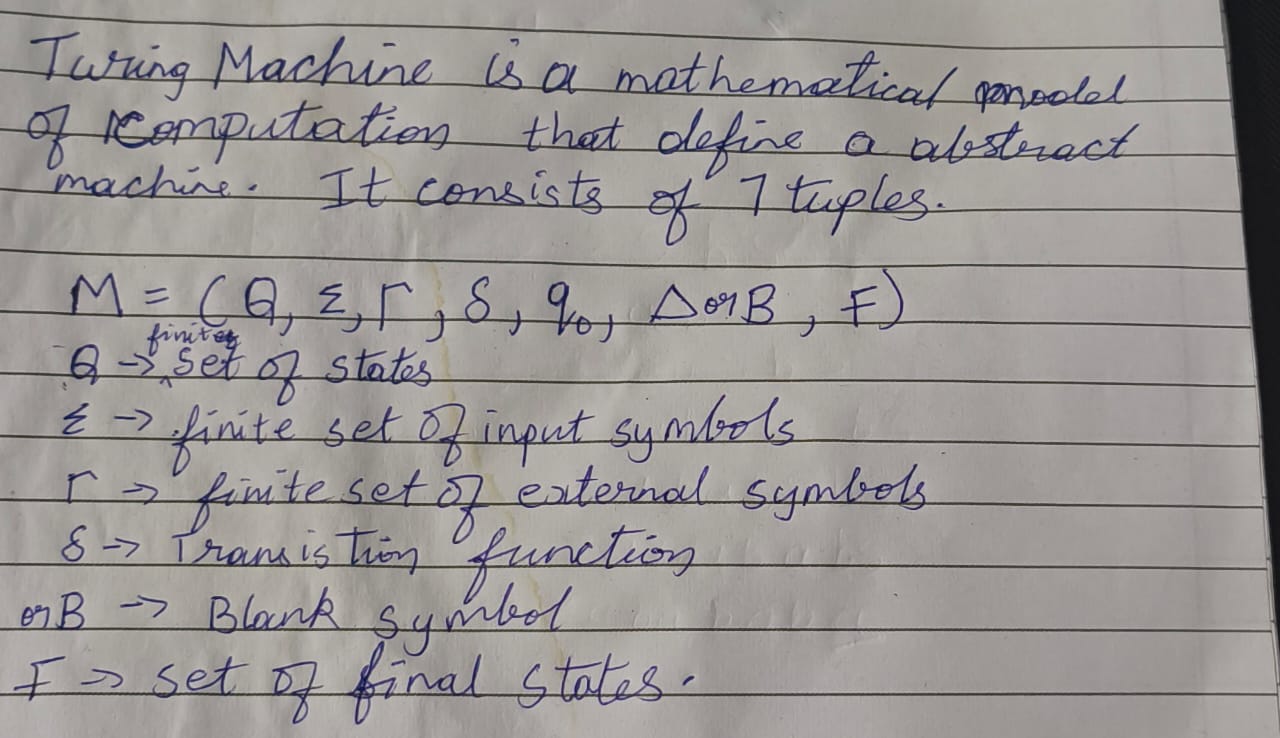
1. Different between NP Hard and NP Complete with example.

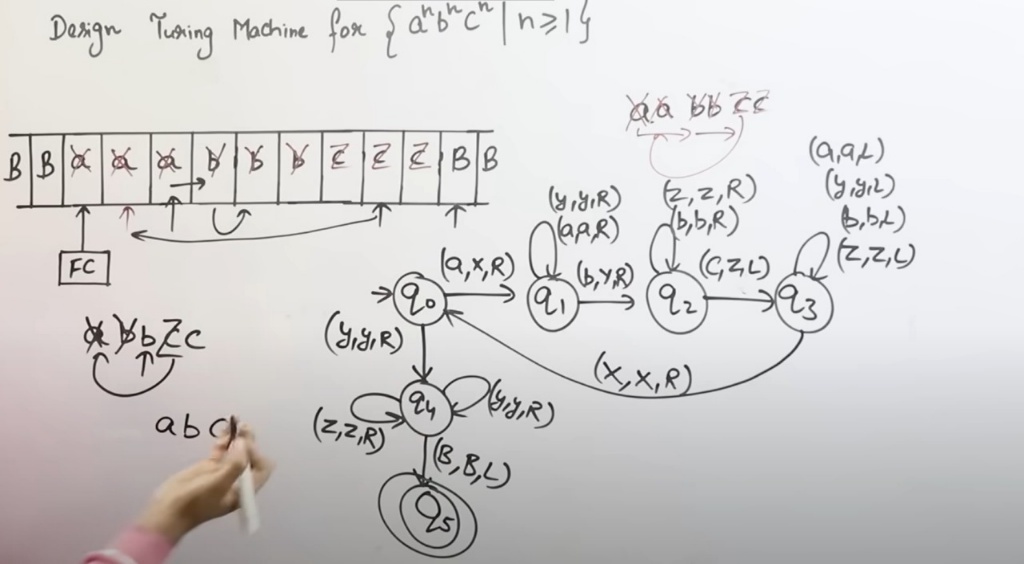
Ans

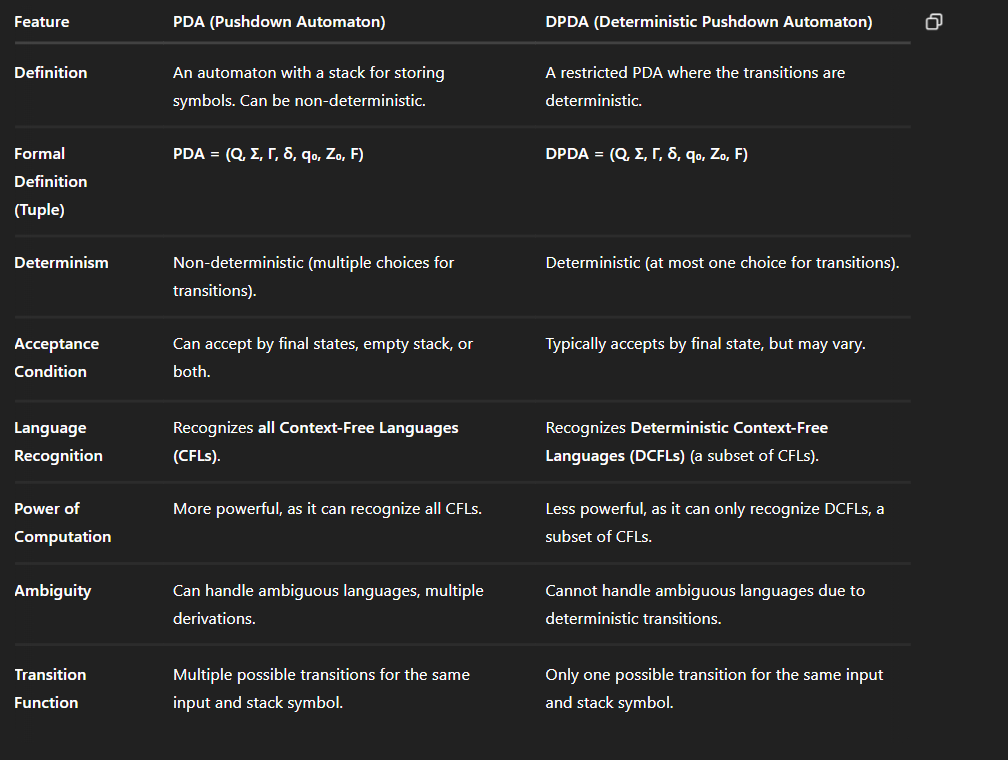
| **Aspect** | **NP-Hard** | **NP-Complete** |
| --- | --- | --- |
| **Definition** | Problems as hard as NP problems; may not be in NP. | Problems in NP that are also NP-Hard. |
| **Membership** | Not necessarily in NP. | Must be in NP. |
| **Examples** | Halting Problem (undecidable). | SAT (Boolean Satisfiability Problem), 3-SAT. |

1. Define Turning Machine with suitable example.

Ans



**Example:**  


8,9 ans

**PDA Example** (Non-Deterministic): The PDA for **L = {a^n b^n c^n | n ≥ 1}** uses **non-determinism** to decide when to switch from 'a' to 'b' and from 'b' to 'c'.

**DPDA Example** (Deterministic): The DPDA for **L = {a^n b^n | n ≥ 1}** deterministically pushes and pops symbols based on the sequence of 'a's and 'b's, with no ambiguity in transitions.