Let S(n) = 1 + 2 + · · · + n be the sum of the first n natural numbers and let $C(n) = 1^3 + 2^3 + \cdots + n^3$ be the sum of the first n cubes. Prove the following equalities by induction on n: **2*3 = 6**

a. $S(n) = \frac{1}{2}n(n + 1)$

b. $C(n) = \frac{1}{4}n^2(n + 1)^2$

Example: Let $S(n) = 1+2+ \ldots\ n$ be the sum of n natural nmbr.

Prove, $S(n) = \frac{1}{2} n(n+1)$.

⇒ Basis: For the basis, we pick $n=0$. For this basis L.H.S = R.H.S = 0

Thus the equation is true when $n=0$.

Induction: Now, assume $n \geq 0$, we must prove that eq(1) implies the same formula with $n+1$.

By algebra, we can write, $S(n) = \frac{1}{2}(n+1)(n+2)$.

And, $S(n+1) = 1+2+ \cdots n + (n+1)$ [by definition]

$= S(n) + (n+1)$ [because $S(n) = 1+2+ \cdots + n$]

$= \frac{1}{2} n(n+1) + (n+1)$ [by induction hypothesis]

$= \frac{1}{2}(n+1)(n+2)$ [by algebra]

~~Example: $C(n) = 1^3+2^3+ \cdots + n^3$ be the~~

$\therefore$ we can say if $S(n)$ then $S(n+1)$ for $n \geq 0$ (Proved)

Example: $C(n) = 1^3+2^3+ \cdots + n^3$ be the sum of n cubes.

prove, $C(n) = \frac{1}{4} n^2 (n+1)^2$.

Basis: for the basis, we pick $n = 0$. For this basis L.H.S = R.H.S $= 0$. Thus the equation is true when $n = 0$.

Induction: Now, assume $n \geq 0$, then we must prove that eq(1) implies the same formula with $n+1$.

By algebra, we can write, $C(n) = 1^3+2^3+ \cdots + n^3 =$

$= \frac{1}{4}((n+1)^4 + 2(n+1)^3 + (n+1)^2)$.

And, $C(n+1) = 1^3+2^3+ \cdots + n^3 + (n+1)^3$ [by definition]

$= C(n) + (n+1)^3$ [$C(n) = 1^3+2^3+ \cdots + n^3$]

$= \frac{1}{4}(n^4 + 2n^4 + n^2) + (n+1)^3$ [induction hypothesis]

$= \frac{1}{4}((n+1)^4 + 2(n+1)^3 + (n+1)^2)$.

② What is finite automata and its application? — ②

⇒ Finite automata is the simplest machine to recognize patterns.

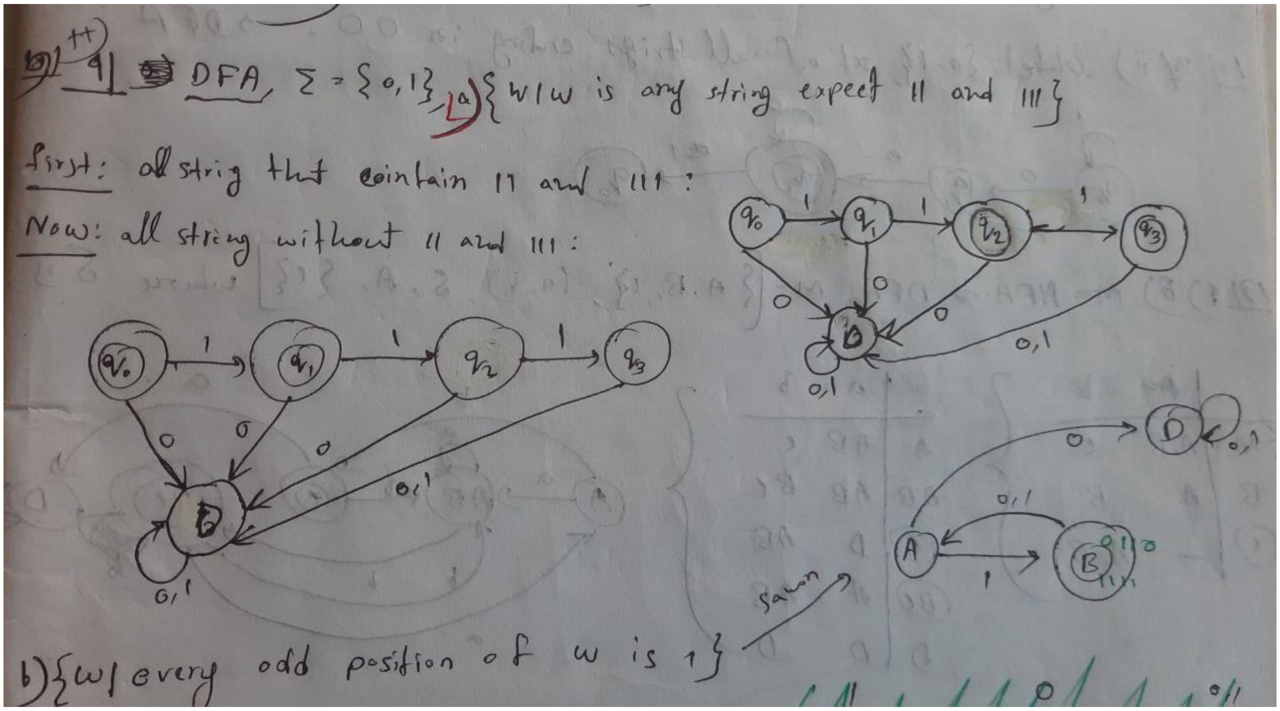
Finite Automata (FA):

The applications of Finite Automata are as follows —

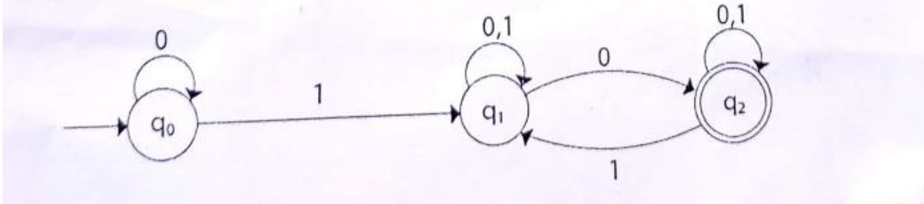
1. Design of the lexical analysis of a compiler.
2. Recognize the pattern by using regular expressions.
3. used in text editors.
4. Use of the mealy and moore machines for designing the combination and sequential circuits.
5. used for spell checkers.

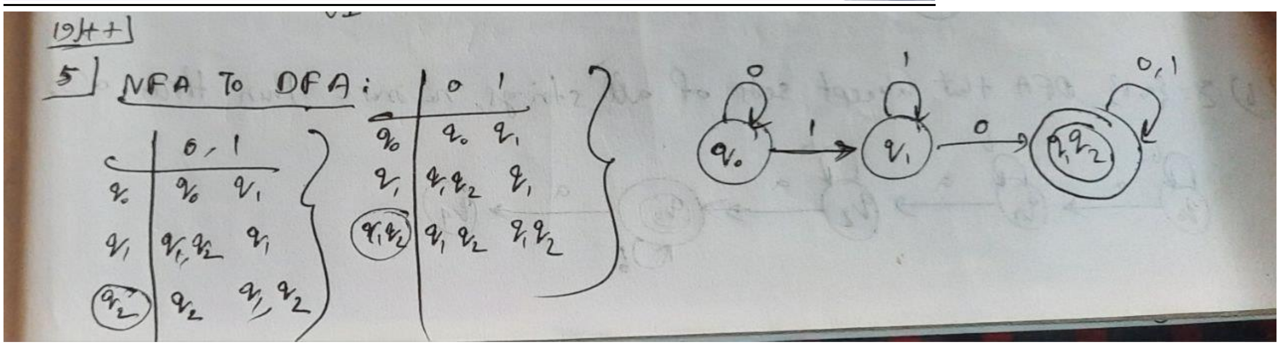
Q1⇒ What is Regular Language?

⇒ A regular language can be defined as a language recognised by a finite automata.



5. Convert the given NFA to DFA.





1.Why does the Finite Automata can't solve the counting problem but the PDA can?

Certainly! Let's consider the specific example of counting the number of occurrences of a symbol 'a' in a string of the form a^n b^n, where n is a non-negative integer. This language is not regular, and a Finite Automaton (FA) cannot recognize it, whereas a Pushdown Automaton (PDA) can.

1. **Finite Automaton (FA):**

- A Finite Automaton has a finite number of states and no memory (or a finite memory in the case of a Deterministic Finite Automaton).

- In the language a^n b^n, the FA needs to remember the count of 'a's to compare it with the count of 'b's, but it has only a finite number of states and no way to remember an unbounded count.

- FA cannot recognize non-regular languages that involve counting, as it lacks the ability to maintain a count beyond its finite set of states.

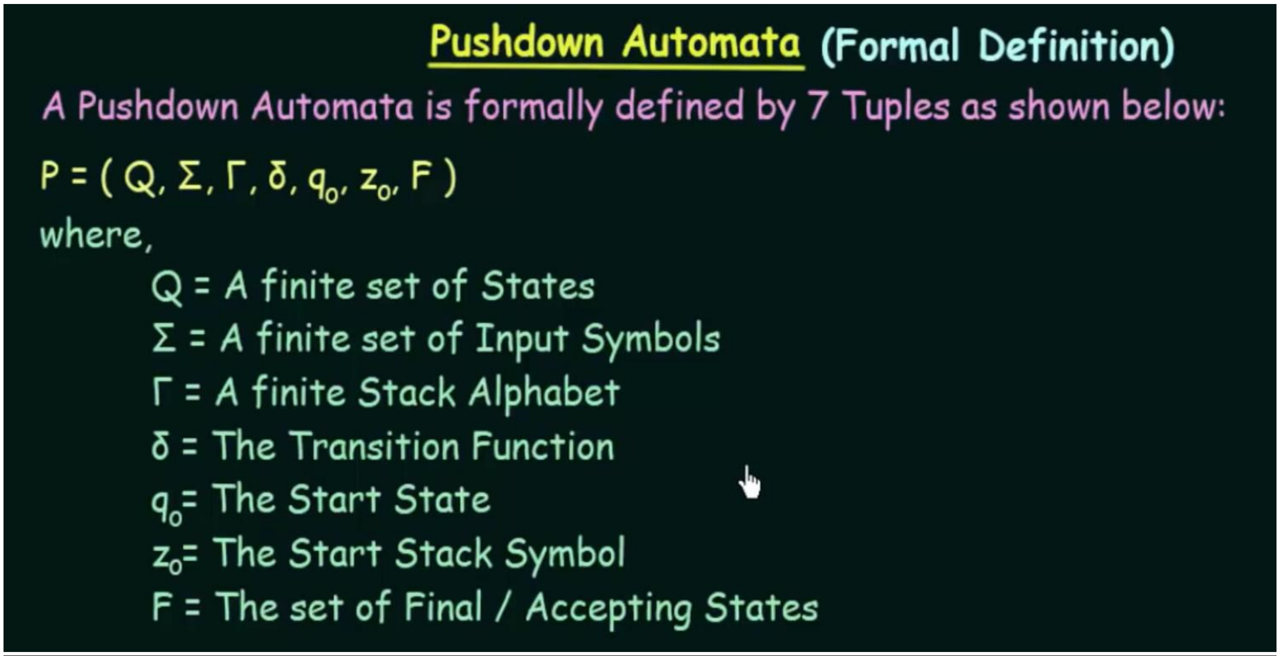
2. **Pushdown Automaton (PDA):**

- A Pushdown Automaton has a stack that can store an unbounded amount of information.

- The PDA can use its stack to keep track of the count of 'a's as it reads the input. It can push 'a' onto the stack for each 'a' it encounters and pop from the stack for each 'b'. At the end of the input, if the stack is empty, the PDA accepts; otherwise, it rejects.

- The stack allows the PDA to maintain a count of 'a's and match them with 'b's, making it capable of recognizing non-regular languages like a^n b^n.

This informal explanation illustrates why a PDA is more powerful than a FA for languages involving counting. The stack provides a form of unbounded memory that allows the PDA to handle more complex patterns and structures in the input. This difference in computational power is a result of the theoretical distinctions between regular and context-free languages. While it's challenging to provide a formal proof in this format, the idea is supported by the rigorous theory of automata and formal languages.



<u>Write the Regular expression that matches the following types of patterns :</u>

"pencil#2","mambo#5", "grade#8"

/[a-z]+\#[0-9]/

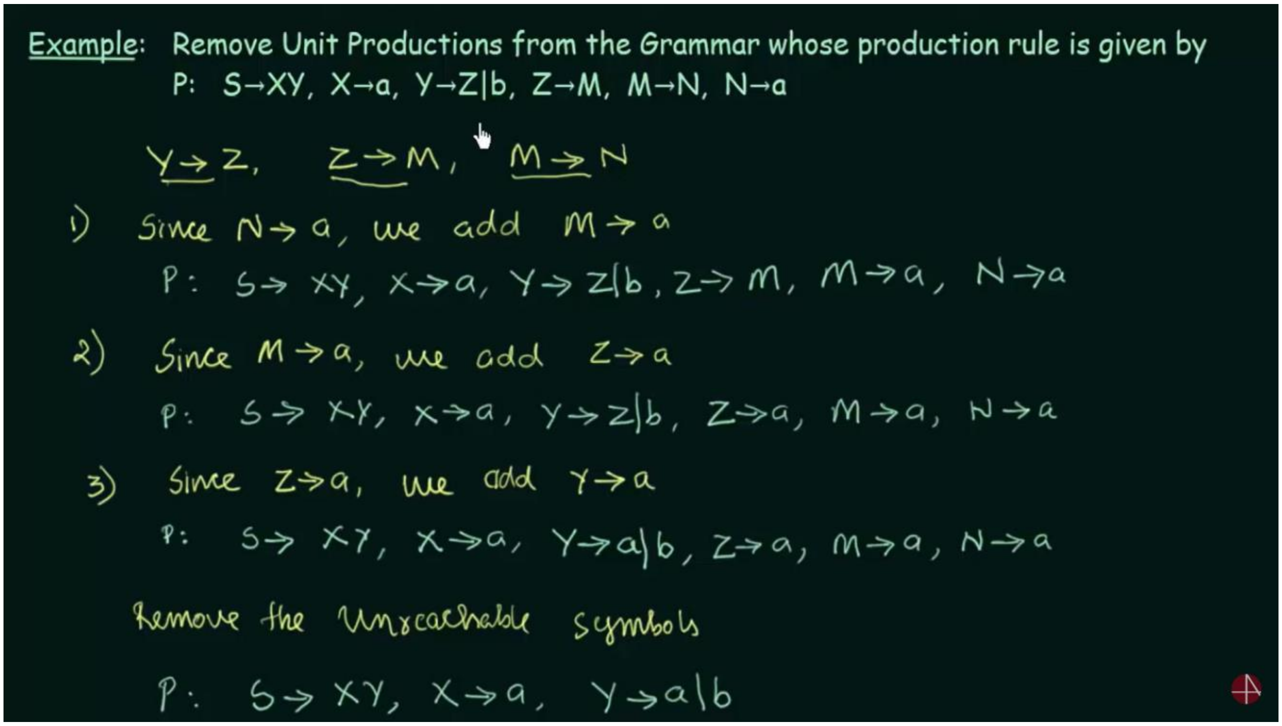
This regular expression consists of the following components:

[a-z]+: Matches one or more letters.

\#: Matches the literal character #.

[0-9]: Matches one digit.

This regular expression will match any string that consists of one or more letters, followed by a # symbol, followed by one digit. For example, it will match the strings "pencil#2", "mambo#5", and "grade#8". It will not match strings that do not follow this format, such as "123" or "pencil2".

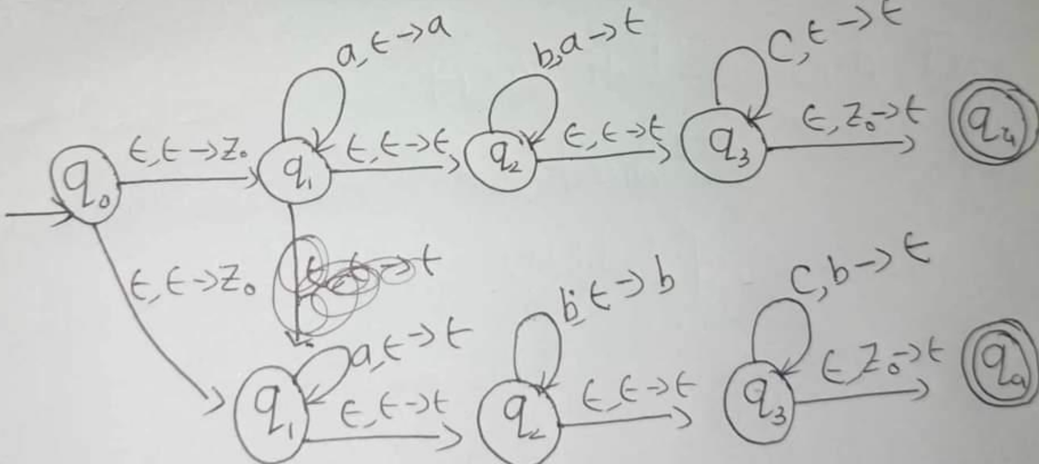


Draw the PushDown Automata for the language D= { aibjck | i, j, k≥0, and i=j or j =k}

Construct a PDA for the language

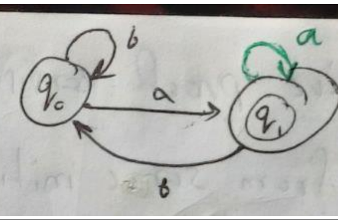
$D = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } j = k\}$

$i = j = n$	$j = k = m$	
$a^n b^n c^k$	$a^i b^m c^m$	
Read a - Push a	Read a	No op
b - Pop	b	Push b
c - No op	c	Pop



b) Draw the DFA for the string ends with 'a'.

19/7/61 ~~Dra~~ DFA, ends with 'a':



Why do we convert CFG to CNF?

d) Converting a CFG to CNF is a technique that is used to simplify the structure of CFG. This is achieved by reducing the number of rules in the grammar and making the grammar more restrictive which is easier to parse.

e) Push down automata is a finite automata with extra memory called stack which helps push down automata to recognise content free languages.

f) for $\Sigma = \{a, b\}$, regular expression for odd string -

$$(a+b)\left((a+b)(a+b)\right)^*$$

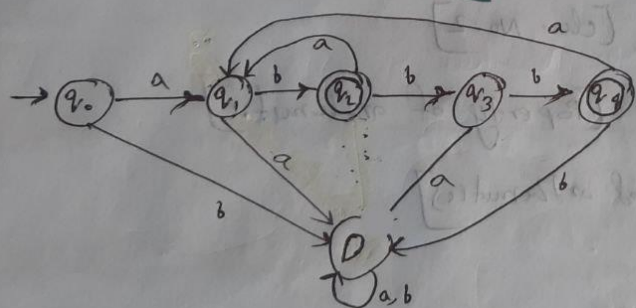
g) A grammar is ambiguous if there exists more than one way to derive or ~~parg~~ parse the same input string.

19/2/ a) DFA, $\Sigma = \{a, b\}$, $\overbrace{\text{exactly 1 or 3 b's}}^{\text{each a followed by}}$:

[redacted]

ab, ~~aab~~

abbb, ~~aabbb~~



To prove that the language $L = \{a^{n^2} \mid n \geq 1\}$ is not regular, we can use the Pumping Lemma for regular languages. The Pumping Lemma states that for any regular language, there exists a constant p (the pumping length) such that any string s in the language with length at least p can be split into three parts xyz, satisfying certain conditions. One of the conditions is that for all $i \geq 0$, the string xy^iz must also be in the language.

Let's assume, for the sake of contradiction, that L is regular. Then, according to the Pumping Lemma, there exists a pumping length p such that any string s in L with length at least p can be split into xyz such that the conditions hold.

Consider the string $s = a^{p^2}$. This string is in L because p^2 is a square and $p \geq 1$. According to the Pumping Lemma, s can be split into xyz with $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L$ for all $i \geq 0$.

Now, let's consider xy^2z. Since $|y| > 0$, the string xy^2z will have more a's than a^{p^2}, and it won't be a square. Therefore, xy^2z cannot be in L because it doesn't satisfy the condition that the number of a's must be a square.

This contradicts the Pumping Lemma, and therefore, our assumption that L is regular must be false. Thus, we conclude that L is ↓ a regular language.

/[a-z]+[0-9][0-9]@student\.sust\.edu/

d⇒ Write the RE that will recognize only the student mail of SUST.

Pumping Lemma দিয়ে বুঝতে পারবো যে একটি language regular language or not regular language.

$a^n b^n ; n \geq 0$

Not a regular L.

FSM ✗

Pumping lemma for regular language:

If A is a Regular language then A has a pumping length "p" such that any string "s" where $|s| \geq p$ may be divided into 3 parts, $s = xyz$, such that the following condition is true.

1) $xy^iz \in A$ when $i \geq 0$ (Y টা variable হবে as power i থাকবে)
2) $|y| > 0$ (Condition গুলো ঠিক হবে যেখানে)
3) $\underbrace{|xy| \leq p}_{x}$ (total length $\rightarrow |xy|$) $\Rightarrow x \leq p$ and $y \leq p$

Lets assume, $a^n b^n ; n > 0$ is a regular language.

$\Rightarrow$ lets a that $p = 7$ (যেকোনো number নিলেই হবে)

$\rightarrow s =$ aaaaaaa bbbbbbb

এখন ভাগে ৩টা ভাগে ভাগ করতে হবে, 3 টা Condition follow করতে হবে,

$\underbrace{aaa}_{x}\underbrace{aaaa}_{y}\underbrace{bbbbbbb}_{z}$

*) 1st case / Case 1 : Y is in "a" part :

$i=2$ নিলে, $\underbrace{aaa}_{x}\underbrace{aaaaaaaa}_{y^2}\underbrace{bbbbbbb}_{z}$

তাহলে $a = 11$, $b = 7$; $a \neq b$

→ y is in b part:

*) Case 2: $\underbrace{aaaaaaa}_{z}\underbrace{bbbb}_{y}\underbrace{bbb}_{x}$

এখানেও আসলে মিলবে না।

So, $a \neq b$.

(যেকোনো একটা case না মিলালেই আর হবে না সব case check করার দরকার নেই।)

→ y is in the a and b part:

*) Case 3: $\underbrace{aaaaa}_{x}\underbrace{aab}_{y}\underbrace{bbbbb}_{z}$

$i=2$, aaaaaaaaabbbbbbbbb

$a = 9 \neq b = 8$

As condition মানে না, So $a^n b^n$ is not regular language.

Now lets consider the 3rd condition, $|XY| <= p$

For the first case, xy=7, p=7 so it satisfies

For the 2nd case, xy=11, p=7 so it doesn't satisfy

For the 3rd case, xy=8, p=7 so it doesn't satisfy

Inductive Proof: Proof by induction is a way of ~~proof~~ Proving that a certain statement is true for every positive integer n.

Suppose, we are given a statement S(n), about an integer n, to prove. One common approach is:

1| The basis: where we show S(i) for a particular integer i. Usually i = 0 or i = 1, but not mandatory.

2| The inductive step: where we assume $n \geq i$, where i is the basis integer, and we show that "if S(n) then S(n+1)".

Proof: For all $n \geq 0$; $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$.

⇒ Basis: For the basis, we pick $n = 0$.

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{2n^3+3n^2+n}{6} \quad \text{—(1)}$$

For this basis L.H.S = R.H.S = 0

Thus the equation is true when $n=0$.

Induction: Now, assume $n \geq 0$, we must ~~proof that~~ prove that eq (1) implies the same formula with $n+1$.

$$S(n+1) = \sum_{i=1}^{n+1} i^2 = \frac{(n+1)(n+2)(2n+3)}{6} = \frac{2n^3+9n^2+13n+6}{6} \quad \text{—(2)}$$

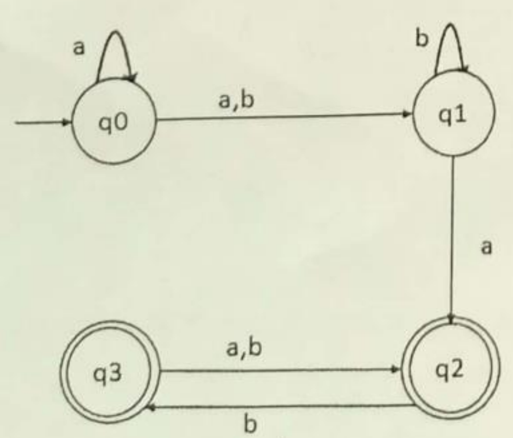
~~Now So, Now, S(n+1)~~ Now, $\sum_{n=1}^{n} i^2 + (n+1)^2 = \frac{2n^3+9n^2+13n+6}{6}$

And, $\frac{2n^3+3n^2+n}{6} + (n+1)^2 = \frac{2n^3+9n^2+13n+6}{6}$

Thus we can say, if $S(n)$ then $S(n+1)$ for $n \geq 0$ (Proved)

3. Answer any TWO

a)

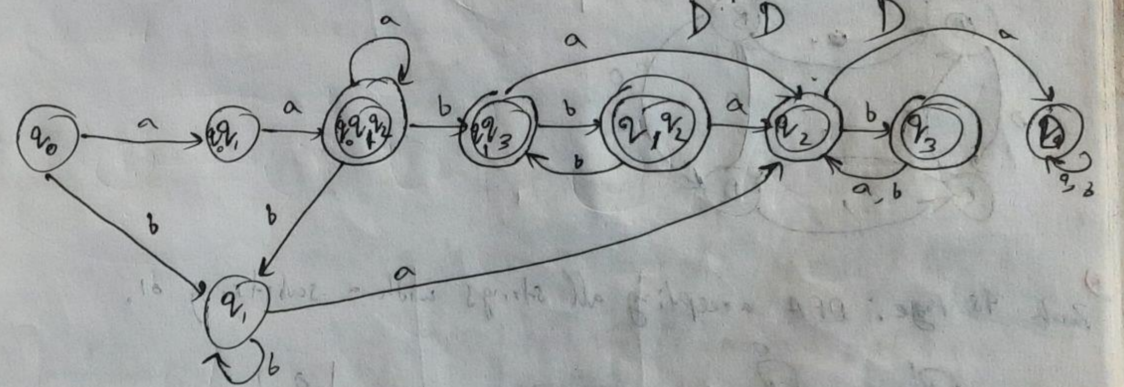


Convert the given NFA to DFA.

19|3| NFA to DFA :

	a, b	
q_0	q_0, q_1	q_1
q_1	q_2	q_1
q_2	ϕ	q_3
q_3	q_2	q_2

	a	b
q_0	$q_0 q_1$	q_1
$q_0 q_1$	$q_0 q_1 q_2$	q_1
$q_0 q_1 q_2$	$q_0 q_1 q_2$	$q_1 q_3$
$q_1 q_3$	q_2	$q_1 q_2$
$q_1 q_2$	q_2	$q_1 q_3$
q_2	D	q_3
q_3	q_2	q_2
q_1	q_2	q_1
D	D	D



Q10 ⇒ What is Turing Machine?

⇒ A turing machine consists of a tape of infinite length on which read and writes operation can be performed.

Q11 ⇒ What is a transition function?

⇒ The transition function defines the movement of an automaton from one state to another by treating the current state and current input symbol as an ordered pair.

Q8⇒ What is an escape sequence?

⇒ An escape sequence is regarded as a single character and is therefore valid as a character constant.

Q9⇒ What is proof by induction?

⇒ proof by induction is a way of proving that a certain statement is true for every positive integer n.

Proof by counterexamples: It shows that a given statement can't possibly be correct by showing an instance that contradicts a universal statement.

Theorem 1.13: All primes are odd.

→ The integer 2 is a prime, but is even.

Write the RE for an even length string only.

(a+b)(a+b)*

Q7⇒ What is FSM?

⇒ A finite state machine (FSM) has a set of states and two functions called the next-state and output function.

FORMAL DEFINITION OF A CONTEXT-FREE GRAMMAR

A context-free grammar is a 4-tuple (V, Σ, R, S), where

1. V is a finite set called the variables,

2. Σ is a finite set, disjoint from V , called the terminals,

3. R is a finite set of rules, with each rule being a variable and a

string of variables and terminals, and

4. S ∈ V is the start variable

FORMAL DEFINITION OF A REGULAR EXPRESSION

Say that R is a regular expression if R is

1. a for some a in the alphabet Σ,

2. ε,

3. ∅,

4. (R1 ∪ R2), where R1 and R2 are regular expressions,

5. (R1 ◦ R2), where R1 and R2 are regular expressions, or

6. (R1∗), where R1 is a regular expression.

In items 1 and 2, the regular expressions a and ε represent the

languages {a} and {ε}, respectively. In item 3, the regular expression ∅ represents the empty language. In items 4, 5, and 6, the

expressions represent the languages obtained by taking the union

or concatenation of the languages R1 and R2, or the star of the

language R1, respectively.

Don’t confuse the regular expressions ε and ∅. The expression ε represents

the language containing a single string—namely, the empty string—whereas ∅

represents the language that doesn’t contain any string

~~e)~~ Write regular expressions for the following languages over the alphabet $\Sigma = \{a, b\}$:

- All strings that do not end with aa.
- All strings that contain an even number of b's.

1st) /(a*b*)*(ab|ba|bb)/

2nd) /(a*ba*ba*)*/

- `(a*ba*ba*)*`: Match any sequence of characters where the number of "b"s is odd (including zero).
    - `a*`: Match any number of "a"s (including none).
    - `b`: Match a single "b".
    - `a*`: Match any number of "a"s (including none).
- This expression allows for any combination of "a" and "b" as long as the count of "b"s is odd.

5 ⇒ "Every DFA in a NFA but not vice versa" - explain that statement. —— (2.5)

⇒ The statement "Every DFA is in a NFA but not vice versa" means that every deterministic finite Automaton (DFA) can be converted into an equivalent NFA, but not every NFA can be converted into an equivalent DFA.

Every DFA is NFA:

NFA $(Q, \Sigma, \delta, q_0, F)$ $\quad \delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$ or $P(Q)$

DFA $(Q, \Sigma, \delta, q_0, F)$ $\quad \delta : Q \times \Sigma \longrightarrow Q$

There are all tuple are the same except the δ transition function, we can say that every DFA is NFA. Because DFA satisfies $\delta : Q \times \Sigma \rightarrow Q$, all moves satisfy the condition.

So, we can say if satisfy $\delta : Q \times \Sigma \rightarrow Q$ then it also satisfy $\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$ or $P(Q)$ because Q also belongs to $P(Q)$.

We also take $\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$ so, here Σ is also belongs to $(\Sigma \cup \{\lambda\})$. So, every DFA is NFA but every NFA is not DFA. Then DFA ⇒ NFA

$(NFA)^{\sim} \Rightarrow (DFA)^{\sim}$

To show that the language $L = \{a^p \mid p \text{ is a prime number}\}$ is not regular, we can use the Pumping Lemma for regular languages. The Pumping Lemma states that for any regular language, there exists a pumping length p such that any string s in the language with length at least p can be split into three parts xyz, satisfying certain conditions. One of the conditions is that for all $i \geq 0$, the string xy^iz must also be in the language.

Let's assume, for the sake of contradiction, that L is regular. Then, according to the Pumping Lemma, there exists a pumping length p such that any string s in L with length at least p can be split into xyz such that the conditions hold.

Consider the string $s = a^p$, where p is a prime number. According to the Pumping Lemma, s can be split into xyz with $|y| > 0$, $|xy| \leq p$, and $xy^iz \in L$ for all $i \geq 0$.

Now, let's consider xy^2z. Since $|y| > 0$, the string xy^2z will have more a's than a^p, and it won't be a prime number of a's. Therefore, xy^2z cannot be in L because it doesn't satisfy the condition that the number of a's must be a prime number.

This contradicts the Pumping Lemma, and therefore, our assumption that L is regular must be false. Thus, we conclude that L is not a regular language.

Example: Remove Null Productions from the following Grammar

$S \rightarrow ABAC, \quad A \rightarrow aA \mid \epsilon, \quad B \rightarrow bB \mid \epsilon, \quad C \rightarrow c$

$A \rightarrow \epsilon \quad , \quad B \rightarrow \epsilon$

1) To eliminate $A \rightarrow \epsilon$

$S \rightarrow \underline{A}B\underline{A}C$

$S \rightarrow ABC \mid BAC \mid BC$

$A \rightarrow aA$

$A \rightarrow a$

New production: $S \rightarrow ABAC \mid ABC \mid BAC \mid BC$

$A \rightarrow aA \mid a \quad , \quad B \rightarrow bB \mid \epsilon \quad , \quad C \rightarrow c$

2) To eliminate $B \rightarrow \epsilon$

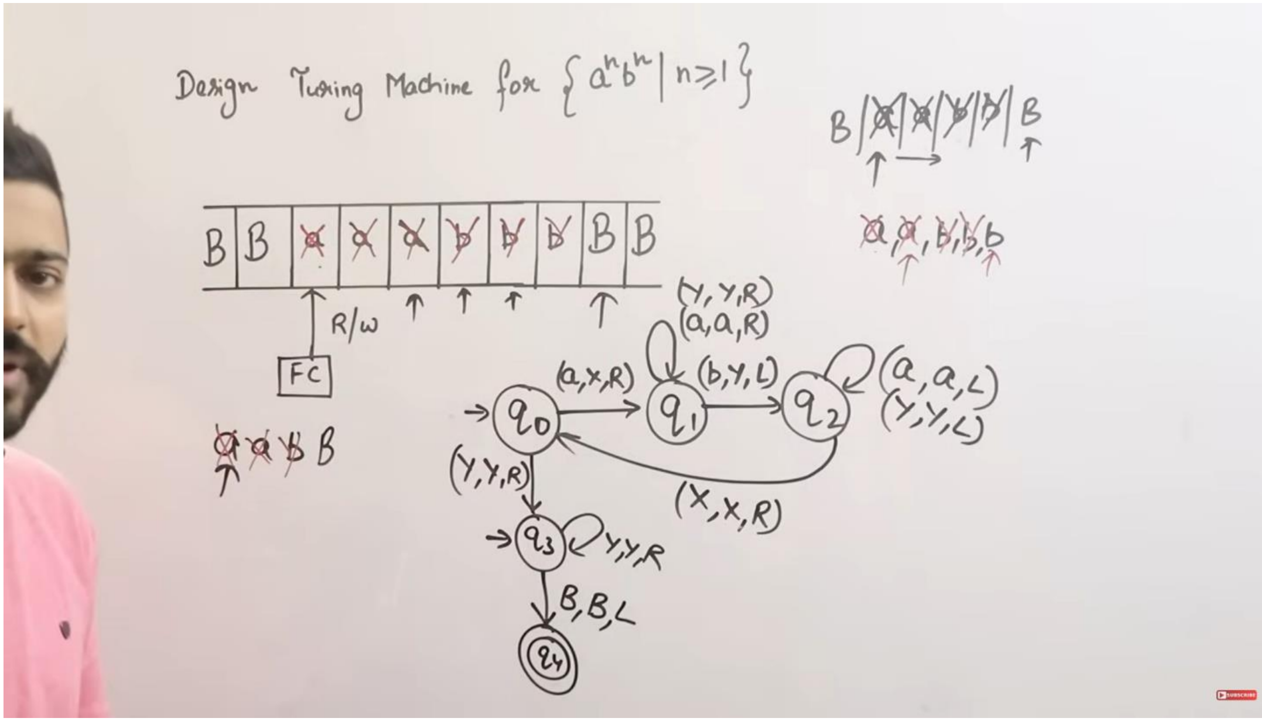
$S \rightarrow AAC \mid AC \mid C \quad , \quad B \rightarrow b$

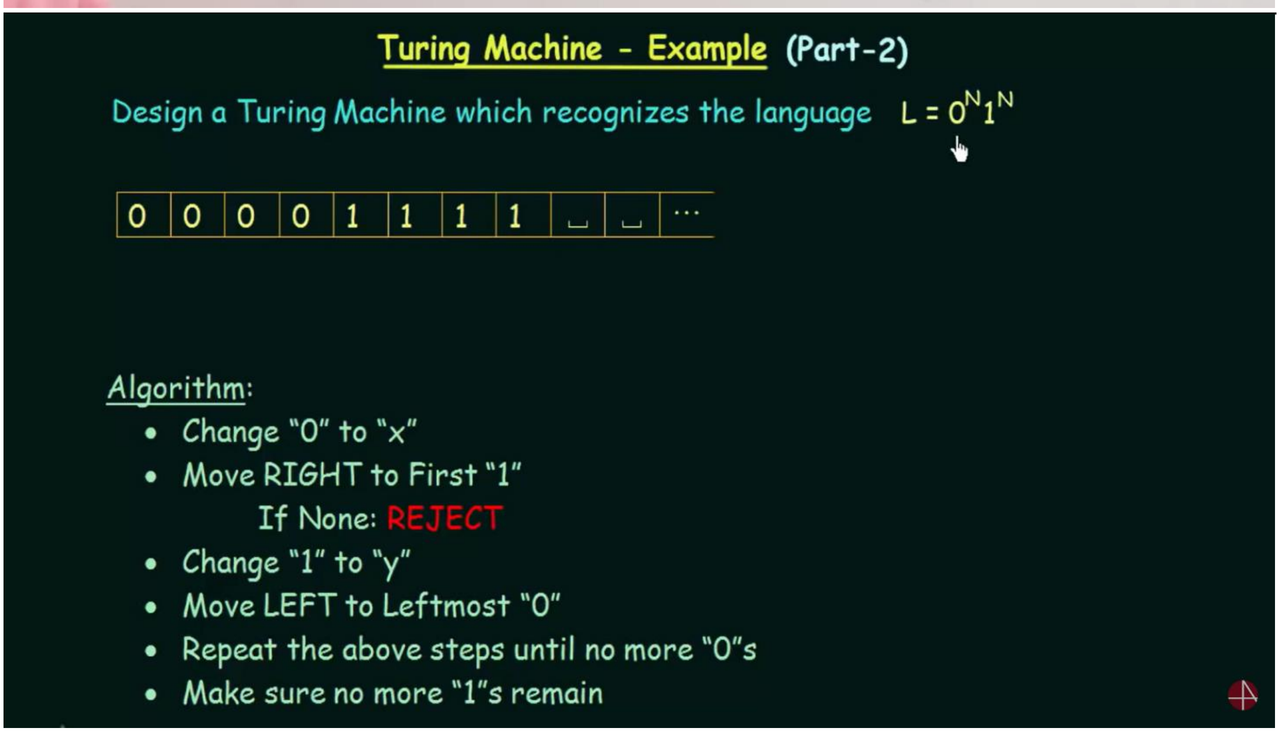
New production: $S \rightarrow ABAC \mid ABC \mid BAC \mid BC \mid AAC \mid AC \mid C$

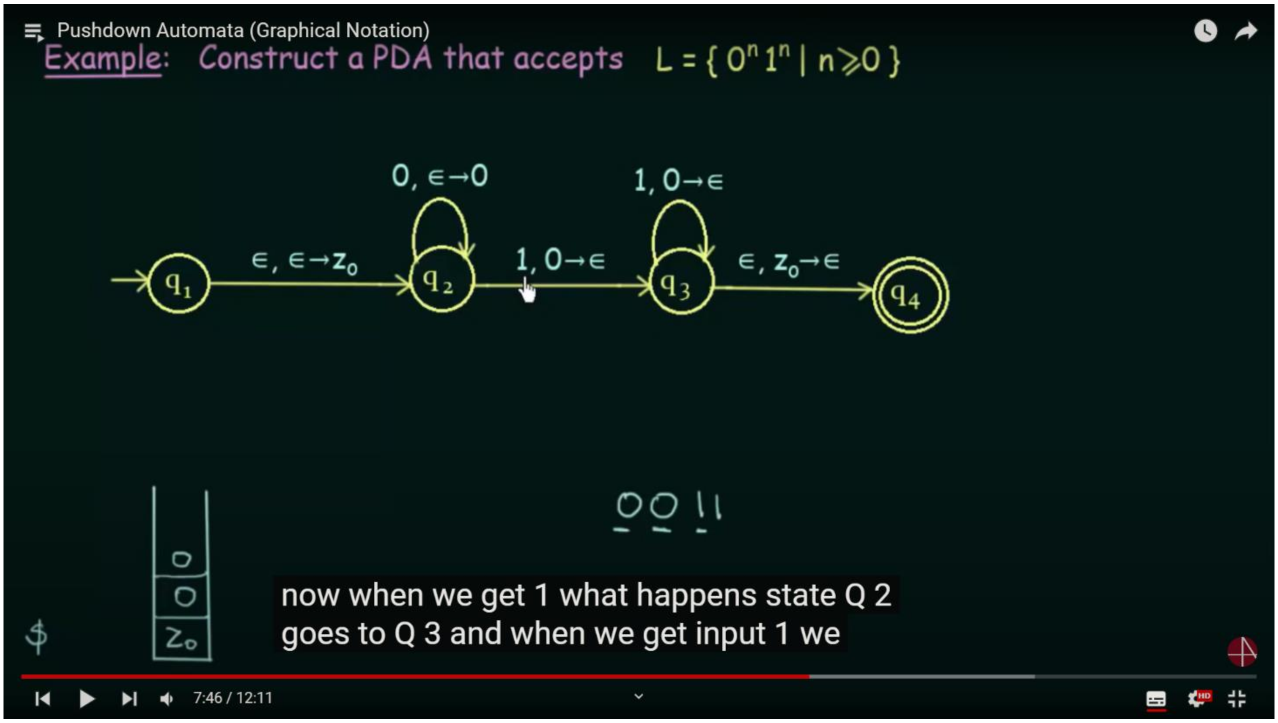
$A \rightarrow aA \mid a$

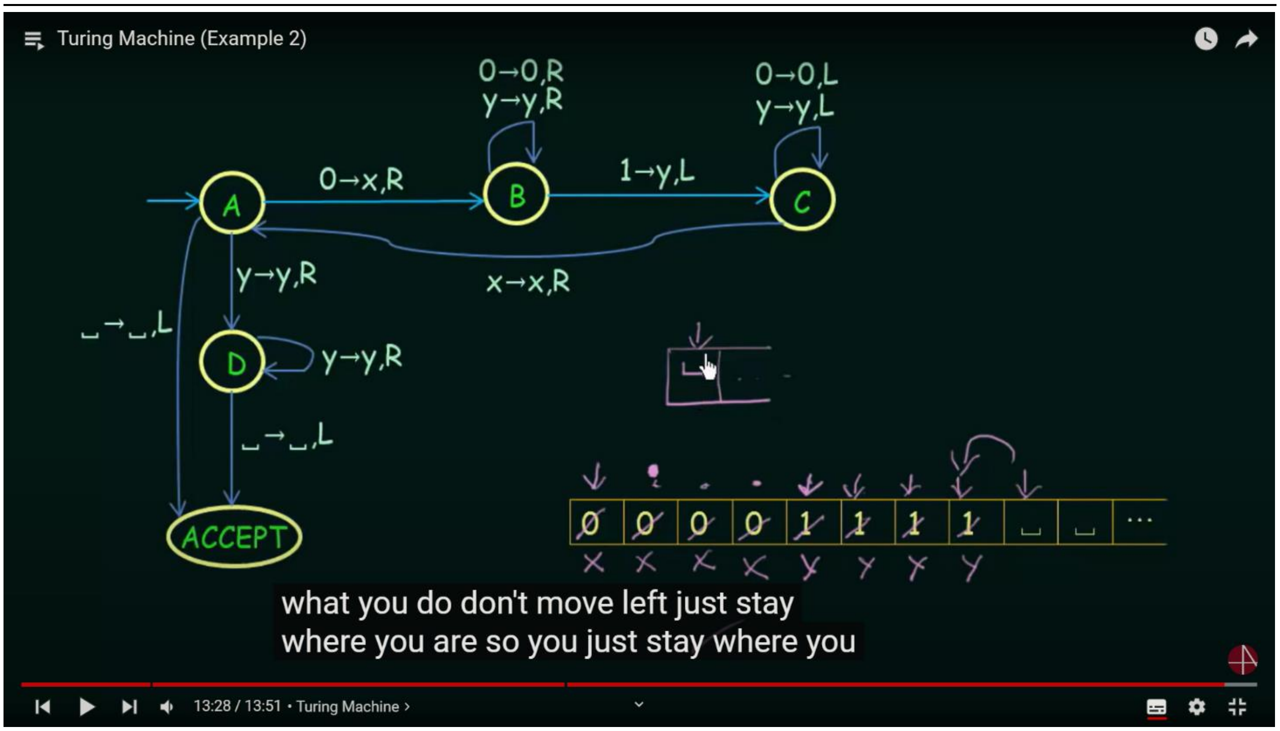
$B \rightarrow bB \mid b$

$C \rightarrow c$


Design Turing Machine for { a^n b^n | n≥1 }
B B a a a b b b B B
R/w
FC
(Y,Y,R)
(a,a,R)
(a,x,R)
(b,Y,L)
(a,a,L)
(Y,Y,L)
q0
q1
q2
(Y,Y,R)
(X,X,R)
q3
Y,Y,R
B,B,L
q4

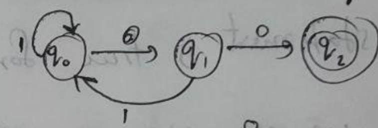

Turing Machine - Example (Part-2)
Design a Turing Machine which recognizes the language L = 0^N 1^N
0 0 0 0 1 1 1 1 ␣ ␣ ...
Algorithm:
• Change "0" to "x"
• Move RIGHT to First "1"
If None: REJECT
• Change "1" to "y"
• Move LEFT to Leftmost "0"
• Repeat the above steps until no more "0"s
• Make sure no more "1"s remain


Pushdown Automata (Graphical Notation)
Example: Construct a PDA that accepts L = { 0ⁿ1ⁿ | n≥0 }
0, ∈→0
1, 0→∈
∈, ∈→Z₀
1, 0→∈
∈, Z₀→∈
q1 q2 q3 q4
0 0 1 1
0
0
Z₀
$
now when we get 1 what happens state Q 2
goes to Q 3 and when we get input 1 we
7:46 / 12:11


Turing Machine (Example 2)
0→0,R
y→y,R
0→0,L
y→y,L
0→x,R
1→y,L
A B C
y→y,R
x→x,R
D
y→y,R
␣→␣,L
␣→␣,L
ACCEPT
0 0 0 0 1 1 1 1 ␣ ␣ ...
X X X X Y Y Y Y
what you do don't move left just stay
where you are so you just stay where you
13:28 / 13:51 • Turing Machine ›

19/6/e) ~~Q~~ Give state diagram of NFAs, Alphabet is $\{0,1\}$. [1.13 nfa s.]

a) $\{w \mid w$ ends with $00\}$ with three state:



b) $0^*1^*0^+$ with three states: