CS303 TANISHQ MALU 1901CS63

Q1

(a) $L = \{ a^n b^j : n = j^2 \}$

By pumping Lemma of content free Language; if a language is CFL then;

Let $s \in L$ and $s = uvxyz$ such that

$|vy| > 0$ and $|vxy| \leq P$ where P is pumping length

then $uv^i xy^i z \in L$ for $i \geq 0$.

(a) Now for given language let p be the pumping length and let the string be " $a^{P^2} b^P$ ".

Let $uvxyz = a^{P^2} b^P$

Let $|v| = |x| = |z| = 0$

$\therefore uy = a^{P^2} b^P$

Let $u = a^{P^2}$ and $y = b^P$

$\therefore$ we know if L is a CFL then $uv^i xy^i z \in L$ $i \geq 0$

Let $i = 0$

$\therefore uv^i xy^i z = uy^0 = a^{P^2}$

Clearly $a^{P^2} \notin L$

Thus $a^n b^j$ is not a CFL by contradiction to pumping lemma.

(b) $L = \{ w \in \{a,b,c\}^* : n_a(w) = n_b(w) \leq n_c(w)$

L contains various types of strings. One of the type is $a^n b^n c^m$ where $m \geq n$. If we prove that $a^n b^n c^m$ is not CFL then L is also not CFL.

Using pumping lemma for CFL. Let $L' = a^n b^n c^m$

Let p be the pumping length. Let $n = p$ and

$m = p+1$

$\therefore \quad S = a^p b^p c^{p+1} = uvxyz$

let $u = a^p$; $|v| = |x| = 0$

$y = b^p$; $z = c^{p+1}$

$\therefore \quad uv^i xy^i z \in L'$

Let $i = 3$

$uv^3xy^3z = uy^3z$

$= a^p b^{p^3} c^{p+1}$

Clearly $a^p b^{p^3} c^{p+1} \notin L'$

Thus $a^n b^n c^m$ is not CFL by contradiction

Thus $L = \{ w \in \{a,b,c\}^* \ n_a(w) = n_b(w) \leq n_c(w) \}$ is also not CFL c $L = \{ w \in \{a,b,c\}^* : n_a(w) / n_b(w) = n_c(w) \}$

Let the string be $s = a^{nm} b^m c^n$

$s \in L$.

Using pumping lemma of CFL

$uvxyz = s = a^{nm} b^m c^n$

Let $n = m = p$ where p is the pumping length

$\therefore uvxyz = a^{p^2} b^p c^p$

Let $|v| = |x| = 0$

$u = a^{p^2}$

$y = b^p$

$z = c^p$

$\therefore uv^i x y^i z \in L$ for $i \geq 0$

Let $i = 2$

$\therefore uv^i x y^i z = a^{p^2} (b^p)^i c^p$

$= a^{p^2} b^{2p} c^p$

$\therefore \dfrac{n_a(w)}{n_b(w)} = \dfrac{p^2}{2p} = \dfrac{p}{2} \neq (n_c(w) = p)$

$\therefore a^{p^2} b^{2p} c^p \notin L$

Thus L is not a CFL by contradiction

d $L = \{a^n b^j : n \geq (j-1)^3\}$

Using pumping lemma of CFL. Let P be the Pumping length of L.

Let $j = P$

$n = P^3 \geq (P-1)^3$

$\therefore s = a^{P^3} b^P$

let $s = uvxyz = a^{P^3} b^P$

Let $u = a^{P^3}$

$|v| = 0$

$|x| = 0$

$y = b^P$

$|z| = 0$

$\therefore uv^i xy^i z \in L$ if L is a CFL

for $i \geq 0$

let $i = 3$

$\therefore s' = uv^3 xy^3 z$

$= a^{P^3} (b^P)^3$

$= a^{P^3} b^{3P}$

now $n = P^3$

$j = 3P$

if $n \geq (j-1)^3$ then $s' \in L$

$(j-1)^3 = (3P-1)^3 = 27P^3 - 1 - 27P + 9P$

$= 27P^3 - 18P - 1$

Checking $n \geq (j-1)^3$

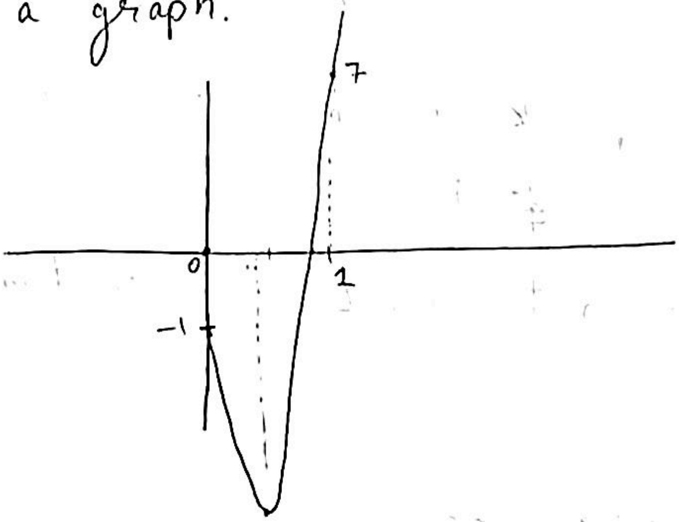
$$p^3 \geq 27p^3 - 18p - 1$$

$(p \geq 0)$
(because pumping length cannot be -ve)

$$0 \geq 26p^3 - 18p - 1$$

(also p is integer)

Forming a graph.



Thus p has no positive integral solution

Thus $s' \notin L$

$\therefore$ L is not a CFL by contradiction

(e) $L = \{ a^{n!} : n \geq 0$

Using pumping lemma for CFL. Let P be the Pumping length.

Let the string $s = a^{P!} = uv^i xy^i z$ for $i \geq 0$

Let $|vy| = k$ where $1 \leq k \leq P$. Thus when string is pumped $s' = a^{P!+k}$

Therefore if $a^{P!+k}$ belongs to L.

Howevever

$$(p+1)! - p! = p(p)! > p \geq k$$

$\therefore$ $(p+1)! - p! > k$

$(p+1)! > p! + k$

$\therefore$ $(p+1)! > p! + k > p!$

Therefore $(p!+k)$ is not a factorial

$\therefore$ $a^{p!+k} \notin L$

$\therefore$ L is not CFL by contradiction

Q2 (i) Let 2 languages be

$L_1 = a^m b^n c^n \quad m,n \geq 0$

$L_2 = a^n b^n c^m \quad m,n \geq 0$

L_1 is a CFL

$S \rightarrow XY$

$X \rightarrow aX \mid \epsilon$

$Y \rightarrow bYc \mid \epsilon$

L_2 is a CFL

$S \rightarrow XY$

$X \rightarrow aXb \mid \epsilon$

$Y \rightarrow cY \mid \epsilon$

However $L_1 \cap L_2 = a^n b^n c^n \quad n \geq 0$ is a very common example, which we know of not a CFL.

Thus CFL is not closed under intersection

$\Rightarrow$ Let 2 Languages be L_1 and L_2 and let CFL be closed under complementation. Then $\overline{L_1}$ and $\overline{L_2}$ are context free.

We know that CFL is closed under union.

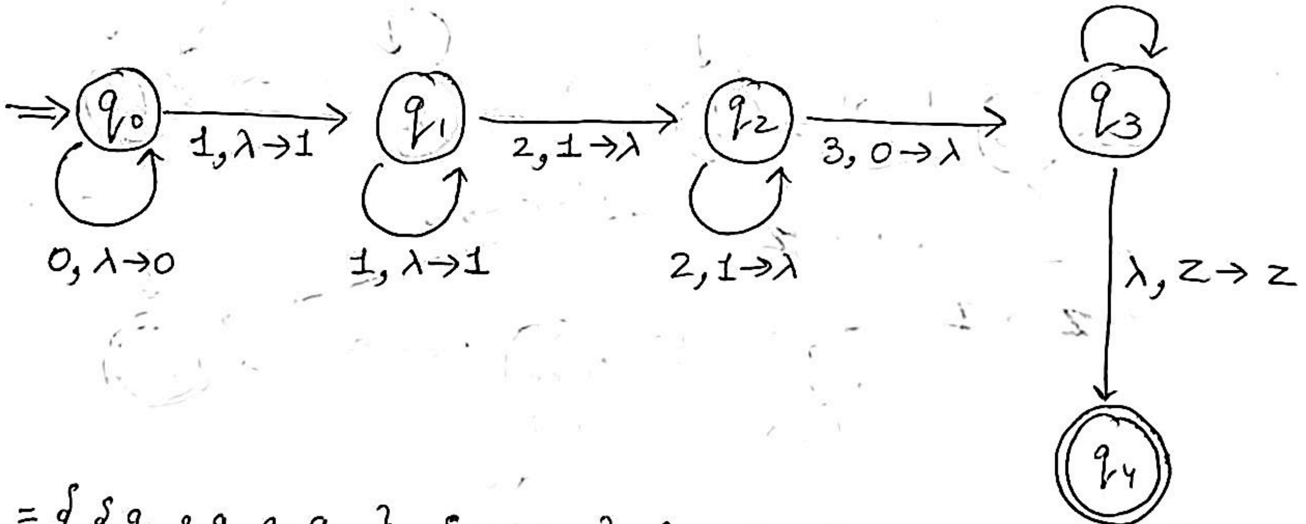
$\therefore$ $\overline{L_1} \cup \overline{L_2}$ is a CFL. Taking is complement, $\overline{\overline{L_1} \cup \overline{L_2}}$ is also a CFL. Using De morgan' Law $\Rightarrow$ $\overline{\overline{A} \cup \overline{B}} = A \cap B$

$\therefore$ $\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$ is also a CFL.

However we know that CFL is not closed under intersection. Thus our assumption of CFL closure under complementation was wrong.

Thus CFL is not closed under complementation.

(ii) PDA for Language $L = 0^n 1^m 2^m 3^n \quad n, m \geq 1$



$M = \{\{q_0, q_1, q_2, q_3, q_4\}, \{0,1,2,3\}, \{z,0,1,2,3\}, \delta, q_0, z, q_4\}$

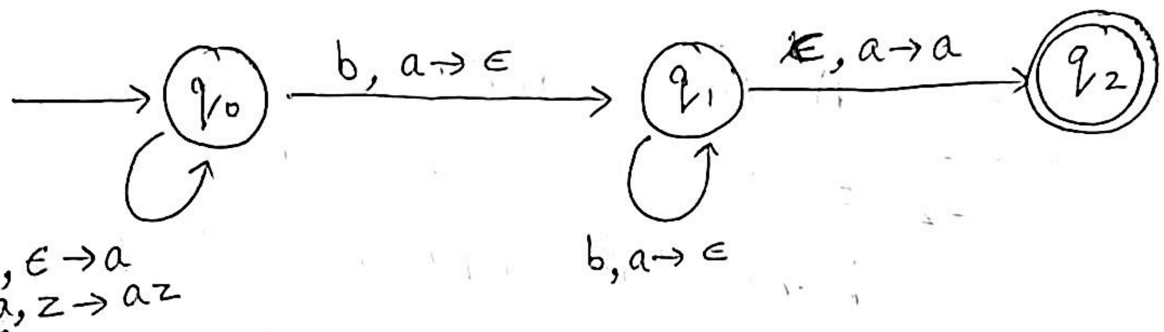
(iii) CFG $\Rightarrow$ $S \rightarrow XY$

$X \rightarrow aX \mid a$

$Y \rightarrow aYb \mid \epsilon$

Language generated by above CFG =>

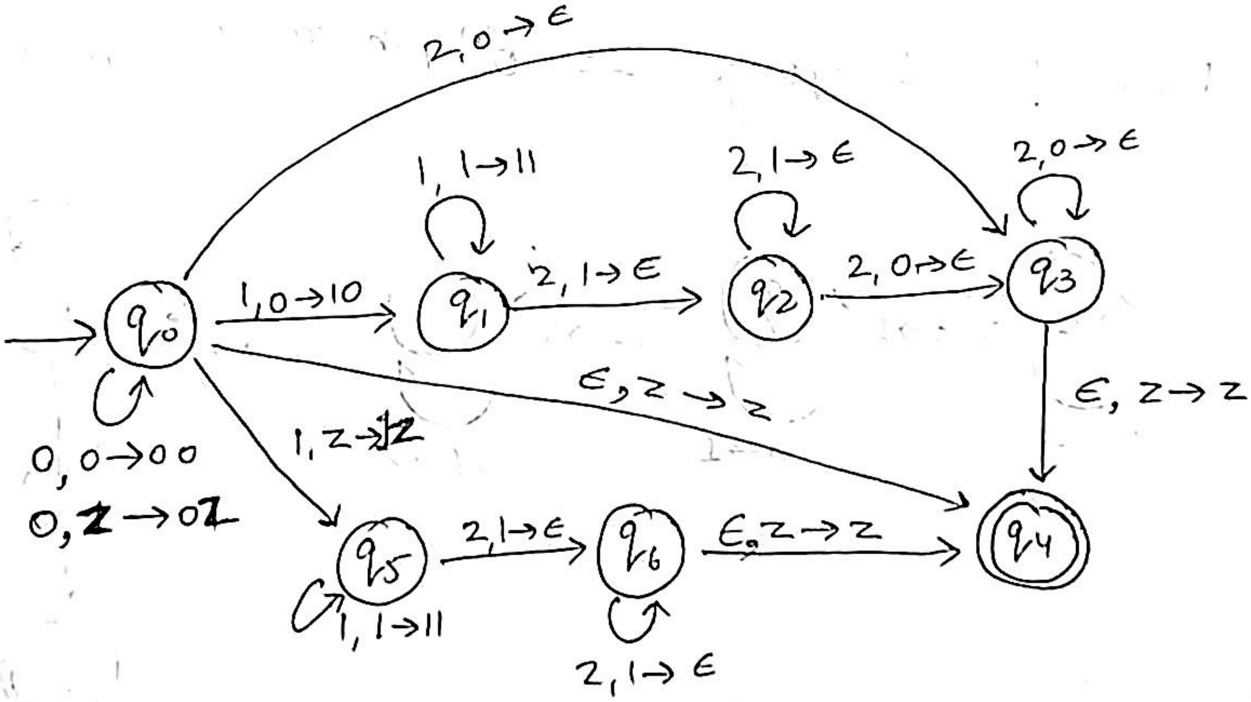
$L = a^n b^m \quad \underline{\underline{n > m}}$



$M = \{\{q_0 q_1 q_2\}, \{a, b\}, \{z, a\}, \delta, q_0, z, q_2\}$

Q3

(i) $\boxed{\text{Let } a=0,\ b=1 \text{ and } c=2}$



Required PDA for
$L = 0^i 1^j 2^{i+j} \quad i, j \geq 0$

$M = \{\{q_0, q_1 q_2 q_3 q_4 q_5\}, \{0, 1, 2\} \{z, 0, 1\}, \delta, q_0, z, q_4\}$

$\delta \Rightarrow$

q_0
- $(0, Z \to 0Z)\ q_0$
- $(0, 0 \to 00)\ q_0$
- $(1, 0 \to 10)\ q_1$
- $(1, Z \to 1Z)\ q_5$
- $(2, 0 \to \epsilon)\ q_3$
- $(\epsilon, Z \to Z)\ q_4$

q_1
- $(1, 1 \to 11)\ q_1$
- $(2, 1 \to \epsilon)\ q_2$

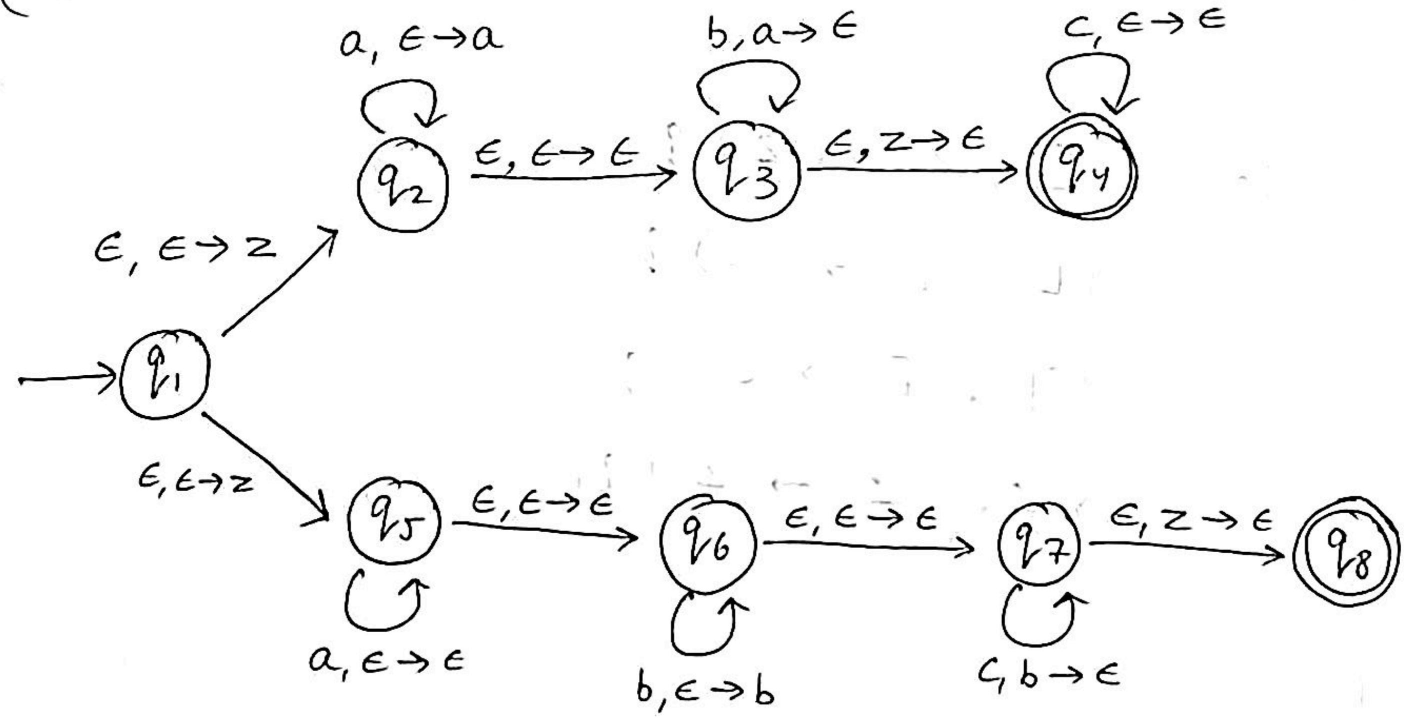
q_2
- $(2, 1 \to \epsilon)\ q_2$
- $(2, 0 \to \epsilon)\ q_3$

q_3
- $(2, 0 \to \epsilon)\ q_3$
- $(\epsilon, Z \to Z)\ q_4$

q_6
- $(2, 1 \to \epsilon)\ q_6$
- $(\epsilon, Z \to Z)\ q_4$

q_5
- $(1, 1 \to 11)\ q_5$
- $(2, 1 \to \epsilon)\ q_6$

(b) $(a^i b^j c^k \mid i\ j\ k \geq 0 \quad i = j \text{ or } j = k)$



$M = \{\{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}, \{a, b, c\}, \{a, b, z\}, \delta, q_1, z, q_4 q_8\}$

$\delta \Rightarrow$

At q_1 we have a nondeterministic branch. If string is $a^i b^i c^k$ $(i=j)$ then PDA goes from $q_1 \to q_2$. If string is $a^i b^j c^j$ $(j=k)$ then PDA goes from $q_1 \to q_5$.

q_2 $(a, \epsilon \to a)\, q_2$
$(\epsilon, \epsilon \to \epsilon)\, q_3$

q_3 $(b, a \to \epsilon)\, q_3$
$(\epsilon, z \to \epsilon)\, q_4$

q_4 $(c, \epsilon \to \epsilon)\, q_4$

q_5 $(a, \epsilon \to \epsilon)\, q_5$
$(\epsilon, \epsilon \to \epsilon)\, q_6$

q_6 $(b, \epsilon \to b)\, q_6$
$(\epsilon, \epsilon \to \epsilon)\, q_7$

q_7 $(c, b \to \epsilon)\, q_7$
$(\epsilon, z \to \epsilon)\, q_8$

Q4

$M = \{\{q_0, q_1\}, \{[,]\}, \{z, [\}, \delta, q_0, z, q_1\}$

$\delta \Rightarrow$

q_0 $([, z \to [z)\, q_0$
$([, [\to [[)\, q_0$
$(], [\to \epsilon)\, q_0$
$(\epsilon, z \to z)\, q_1$

given string [[][]][]

$(q_0, [[][]][], Z) \vdash (q_0, [][]][], [Z)$

$\vdash (q_0,][]][], [[Z) \vdash (q_0, []][], [Z)$

$\vdash (q_0,]][], [[Z) \vdash (q_0][], [Z) \vdash (q_0, [], Z)$

$\vdash (q_0,], [Z) \vdash (q_0, \epsilon, Z) \vdash (q_1, \epsilon, Z)$

Q6

(a) Every language accepted by a multitape TM is recursively enumerable.

Sol Suppose language L is accepted by a n-tape Turing machine "M". Let's try to simulate it with a one tape TM "T" whose tape has 2n tracks. Half these tracks holds the tapes of M and the other half of the tracks each hold only a single marker that indicates where the head for corresponding tape M is currently located.

To simulate a move of M, T's head must visit the n head markers. To maintain how many head markers are to its left at all times, the count is stored as a component of T's finite control. After visiting each head marker and storing the scanned symbol in a component of its finite control, T knows which tape symbols are being scanned by each of M's head. T also knows the state of M which is stored in T's own finite control. Thus T knows all moves of M.

T now revisits each head marker on its tape, changes the symbol in the track representing the corresponding tapes of M and moves the head markers left or right if required. Finally T changes the state of M as recorded in its own finite control. At this point T has simulated one move of M.

We select T's accepting state as all those states that record M's state as one of the accepting state. Thus whenever M accepts, T also accepts and vice-versa.

Hence Language accepted by a Multitape TM is accepted by a one tape turing machine and is recursively enumerable.

(b) If M_1 is a NTM, then there is a deterministic TM M_2 such that $L(M_1) = L(M_2)$

Sol

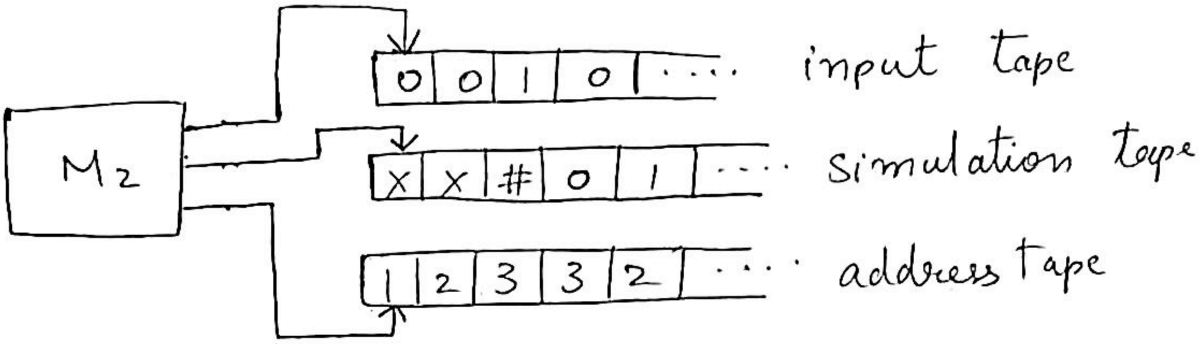
An NTM M_1 can be simulated with DTM M_2

It will be designed as a Multitape (3 tapes)

Tape1 (read only) – Records the input string. Its reused many times.

Tape2 – Used as M_1's tape

Tape3 (a counter) – It stores an integer number $d_1 d_2 d_3 \dots d_n$ where each d_i indicates a choice to make at step i. Ex- pick the 1st branch, 2nd branch etc. The max. value of d_i is the largest no. of choices given by δ_{M_1}.



(an example of M_2)

1.) The DTM M_2 begins with tape 2, 3 empty and tape 1 holds the input.

2.) Wipe tape 2 and copy the input string from tape 1 to tape 2. Simulate NTM M_1 on tape 2.

2.1) At each step i, determine the value v of cell d_i on tape 3

2.2) If v is valid, then update tape 2.

2.3) If not, abort the branch; goto step 3

2.4) abort if transistion represent rejects.

3.) Increment the value on tape 3; goto step 2.

It clearly will accept if it finds that NTM M_1 can enter an accepting state. To confirm it :-

Let m be the maximum no. of choices NTM M_1 has for any configuration. Thus after n moves M_1 can reach at most $1 + m + m^2 \ldots m^n$ points i.e nm^n points. These points are searched in a breadth first search manner and this bound is sufficient to assure us that the accepting state will be considered eventually at some point of time

Thus if M_1 accepts so does M_2. Since we already know that if M_2 accepts it does only so because M_1 accepts Thus we conclude $L(M_1) = L(M_2)$

Although time taken by DTM M_2 can be exponentially larger than NTM M_1.

Q7

Sol (a) Since we have a NTM, we can have a nondeterministic branch at initial state to either move left or right, entering one of 2 different states on either side of initial state. Each of this state can proceed in its own direction i.e left or right, and will keep moving until it sees a \$ as it enters state P.

The pointer has to move off the \$ entering another state and then move back to \$ To enter into state P when it moves back.

(b) Since we have a deterministic TM we can't have 2 branches. Instead we will now oscillate left and right and will use some left marker X and right marker Y, to keep note of how far we have moved in that direction.

Start moving one cell right and mark Y then move 2 cell left and mark X. Repeatedly now move to Y move Y one cell right, go left to X and move it one cell left and repeat until \$ is found.

This way we can find our symbol \$.