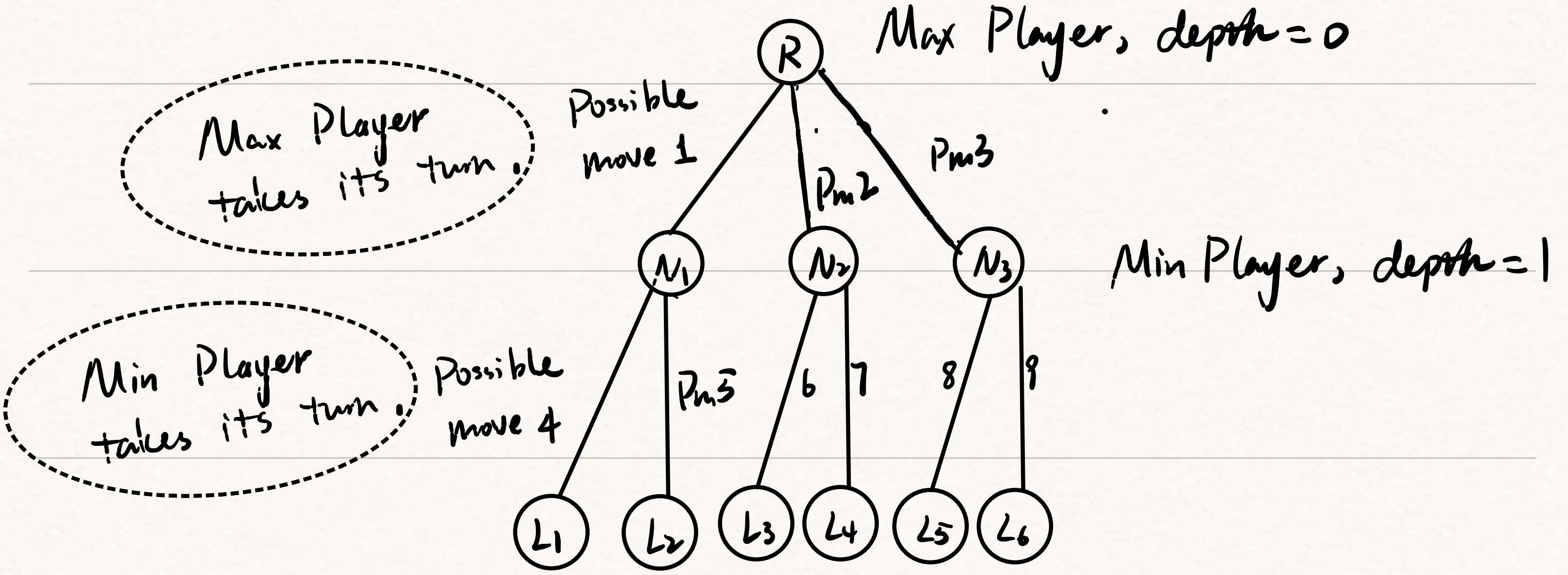
Min-Max ~ Alpha-Beta prunning



First, by DFS, we arrive L_1 node through $R - N_1 - L_1$

At L_1, by MCT, it would calculate the score of current situation (denoted as l_1), and return " move = None, score = l_1 to N_1

N_1 update : min_score = l_1 # Current best situation for Min Player

best_move = Pm4 # Current best choice for

Min Player

beta = l_1 # parameter passed to the parent node N-level

Assume $l_2 > l_1$, so L_2 doesn't update N_1.

Now, we go back from L_2 to N1 to R.

R update: max_score = l_1

best_move = Pm1.

alpha = l_1

Then we arrive L_3 through R - N_2 - L_3.

Assume $l_3 < l_1$. N_2 update: min_score = l_3

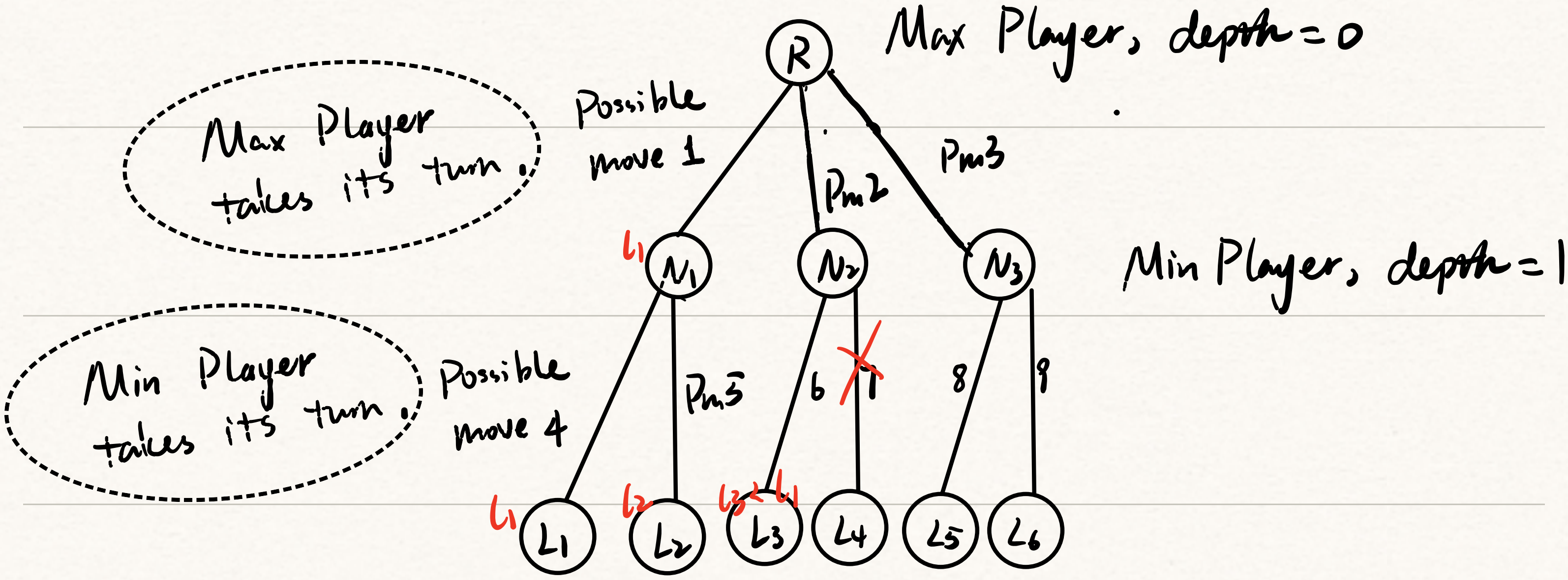
best_move = Pm6

beta = l_3.

$\because \underset{L_3}{beta} < \underset{L_1}{alpha}$

$\therefore$ break. back to R.

Meaning:



Since N_2 is the stage that Min Player takes its turn, therefore the value return by $N_2 \le L_3 < L_1$. So at Node R, Max Player will not choose Pm 2. Thus, given L_3, we can tell that N_2 will never be reached in reality. We don't have to search its children nodes then.