Q.5 → Pascal's Triangle :-

Method 1 :- Iterative Sol^{n} :-

Step 1 :- Pascal's Triangle में हमारा हर row का 1st और last Element 1 होता है। इसलिए इस step में हम edge case देखेंगे।

(i) अगर given rows '0' है तो '[]' (empty array) return कर देंगे।

(ii) और हर row के 1st और last element पर 1 रख देंगे।

e.g. :-

row 1 → [1]
row 2 → [1, 1]
row 3 → [1, _, 1]
row 4 → [1, _, _, 1]
row 5 → [1, _, _, _, 1]

∴ इसमें देखिए हमें अब सिर्फ middle वाले elements find करने हैं।

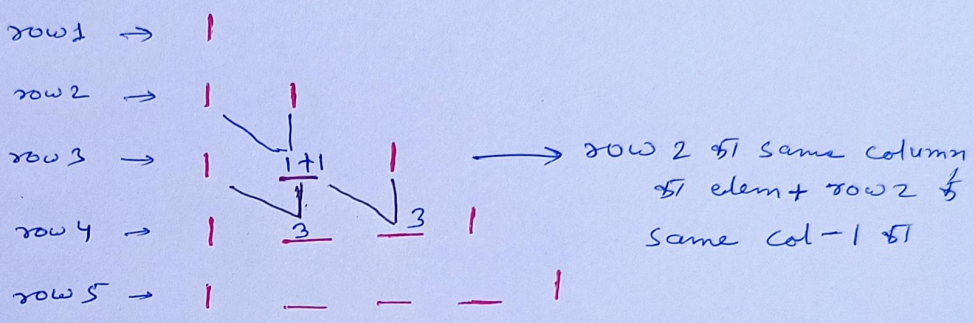
Step 2 :- हमने triangle name का variable बनाकर उसमें 1 push कर दिया है, क्योंकि अगर हमारा इसमें given rows '1' है तो इस triangle को return कर देंगे।

∴ Let triangle = [[1]];

इसके बाद हम हर rows को traverse करने के लिए, एक loop चलाएँगे।

Step3 :- इसमें हम एक और loop का use करेंगे, जो current row के हर elements को traverse करेगा। और इसी loop के अंदर हम हर row के middle elements को fill करेंगे।

eg :-



∴ हर row के middle elements को fill करने के लिए — उसके previous Row के same col. (मतलब row3 में 2nd col. fill करना है तो row2 का भी 2nd col) + previousRow के same col − 1 का Value put कर देंगे।

Note :- जितना space middle में खाली है, inner loop (row के elements को traverse) उतनी बार ही चलेगा।

Step4 :- इस step में हम last element मतलब 'i' को currentRow में push कर देंगे।

∴ इसके बाद currentRow को हम triangle में push कर देंगे।

Dry Run :-

numOfRows : 5 , triangle = [[1]]

For i=1 , 1 < 5 (true)

↳ prevRow = triangle [i-1] = triangle [1-1]
= triangle [0]
= [1].

currentRow = [1]

j=1 , 1 < 1 (false)

↳ loop terminates.

currentRow = [1,1]

triangle = [[1], [1,1]]

For i=2 , 2 < 5 (true)

↳ prevRow = triangle [i-1] = triangle [1]
= [1,1]

currentRow = [1]

j=1, 1 < 2 (true)

↳ currentRow.push = prevRow [1-1] + prevRow [1];

= 1 + 1
= 2

currentRow = [1,2]

j=2, 2 < 2 (false) → loop terminates.

currentRow = [1,2,1]

triangle = [[1], [1,1], [1,2,1]]

For i=3, 3<5 (true)

↳ prevRow = triangle[3-1] = triangle[2]
= [1,2,1]

currRow = [1]

j=1, 1<3 (true)

↳ ⇒ prevRow[0] + prevRow[1];

⇒ 1 + 2

currentRow = [1,3].

j=2, 2<3 (true)

↳ ⇒ prevRow[1] + prevRow[2]:

⇒ 2 + 1

⇒ 3

currentRow = [1,3,3]

j=3, 3<3 (false)

↳ Loop terminates

currentRow = [1,3,3,1]

triangle = [[1], [1,1], [1,2,1], [1,3,3,1]]

For i=4, 4<5 (true)

↳ prevRow = triangle[4-1] = [1,3,3,1]

currentRow = [1]

j=1, 1<4 (true)

↳ prevRow[1-1] + prevRow[1]

⇒ 1 + 3 = 4

currentRow = [1,4]

j=2 , 2 < 4 (true)

↳ prevRow[1] + prevRow[2]

→ 3 + 3 = 6

currentRow = [1,4,6]

j=3, 3 < 4 (true)

↳ prevRow[2] + prevRow[3]

→ 3 + 1 = 4

currentRow = [1,4,6,4]

j=4, 4 < 4 (false) → loop terminates.

∴ currentRow = [1,4,6,4,1]

∴ triangle = [[1], [1,1], [1,2,1], [1,3,3,1], [1,4,6,4,1]]

for i=5 , 5 < 5 (false)

↳ loop terminates.

2