Relation between Permutation and Combination

Mathematical Relation: $^nC_r = \frac{\lfloor n}{\lfloor n-r \lfloor r}$ | $^nP_r = \frac{\lfloor n}{\lfloor n-r}$

(1) So, $\boxed{^nP_r = {}^nC_r \times \lfloor r}$

(2) Total count of characters in power set: power set $= (1+1)^n = 2^n$

$2^n = {}^nC_0 + {}^nC_1 + {}^nC_2 + {}^nC_3 + \cdots + {}^nC_{n-3} + {}^nC_{n-2} + {}^nC_{n-1} + {}^nC_n$

n-char, n-char, n-char, n-chars

Since there are 2^n terms, and 2 terms togeter gives 'n' char.

So, total number of chars in power set = ((2^n)/2)*n = 2^(n-1)*n

Permuation and combination in terms of arrangment :

Permuation: Arranging 'r' distinct items at 'n' positions. Eg. Arranging 2 distinct items(a,b) at 3 positions. $^3P_2 = 6$

Combination: Arranging 'r' identical items at 'n' positions. Eg. Arranging 2 (i,i) at 3 positions ↳ $^3C_2 = 3$

permutation of 2 identical items at 3 positions nCr

$^3C_2 =$ i i _ / i _ i / _ i i

permutation of 2 distinct items at 3 positions nPr

1 2 _ → 2 1 _

1 _ 2 → 2 _ 1

_ 1 2 → _ 2 1

$^3P_2 = 6$

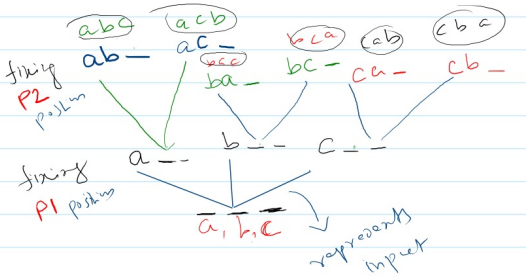
↳ Since r =2, it means against each combination there will be r! copies(2!=2) of permuation.

$2^2 \times -$

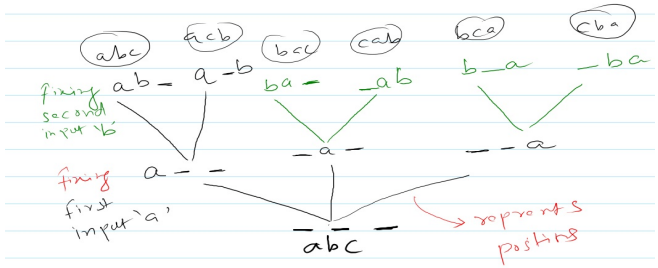
Approach for permuation tree formation:

1. By fixing the position and taking input elements as options
2. By fixing the input elemnt and taking pisitions as options

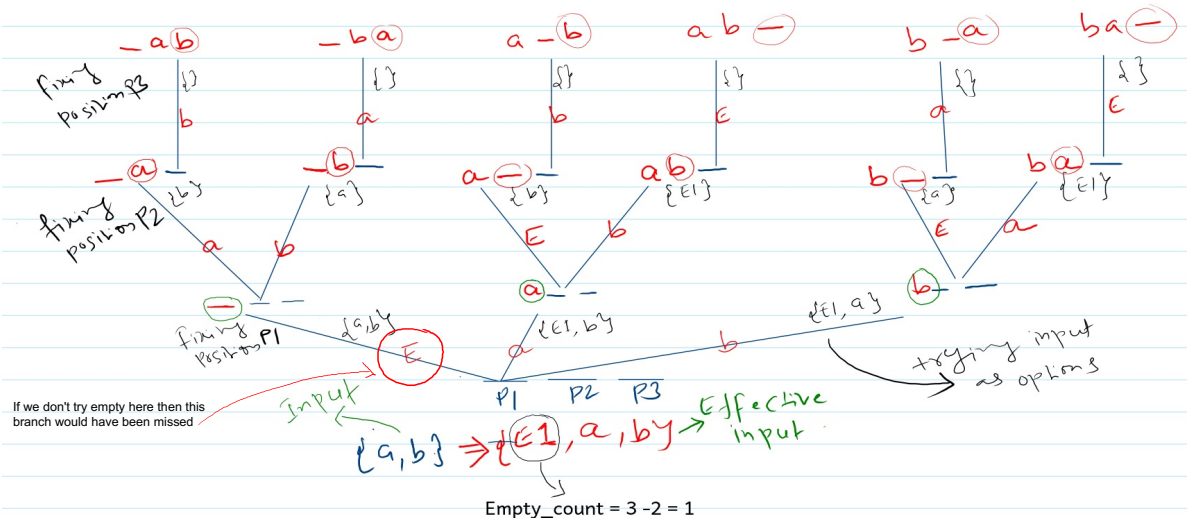
1A. By fixing the position and taking input elements as options where input_count == position_count



2.By fixing the input elemnt and taking pisitions as options where input_count <= position_count



1B. By fixing the position and taking input elements as options where input_count < position_count



Empty_count = 3 -2 = 1

Note: since input_count is smaller than position_count so, input options to try at level will get exhausted before reaching to leaf level. This is why 'empty' need to be treated as special input.

EMPTY_COUNT = POSITION_COUNT - INPUT_COUNT

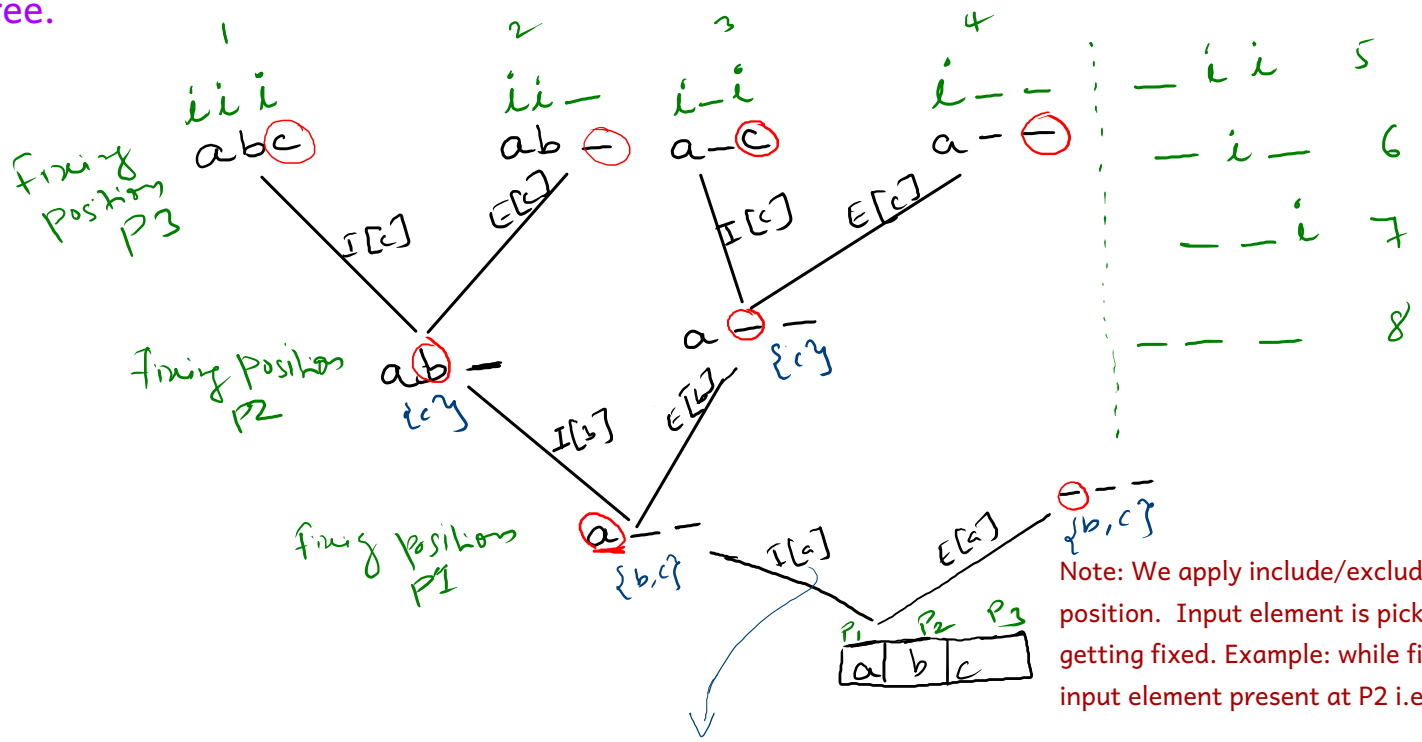
Approaches for Combination tree formation:

1. Pascal_Identity based Include_Exclude_Tree tree by fixing position
2. Pascal_Identity_Expansion based Include_Tree by fixing position

1. Pascal_Identity based Include_Exclude_Tree tree by fixing position where

input_count <= postion_count

Note: Position is fixed at each level, and include(i) & exclude(i) are taken as options i.e. branches of tree.



Note: We apply include/exclude option at each level while fixing the position. Input element is picked from the same position that is getting fixed. Example: while fixing P2 at level 2, we need to pick input element present at P2 i.e. 'b' .

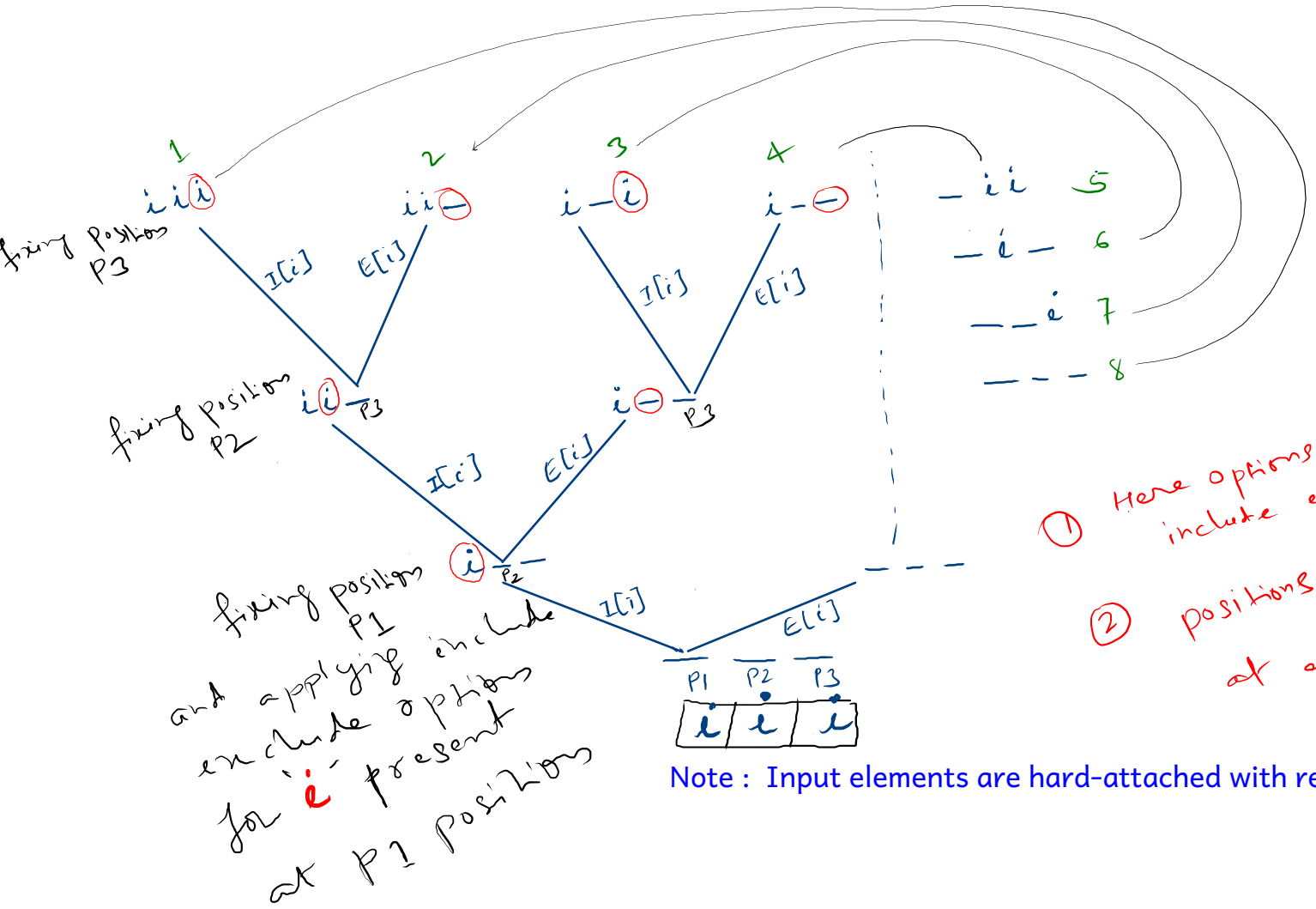
performing opertation include[a] and exclude[a] on the input element persent at postion 'p1' (since we are fixing position P1 at level 1).

Note : Input elements are hard-attached with respective positions.

1. Pascal_Identity based Include_Exclude_Tree tree by fixing position where

input_count <= position_count

power set by placing 'i' on n given position



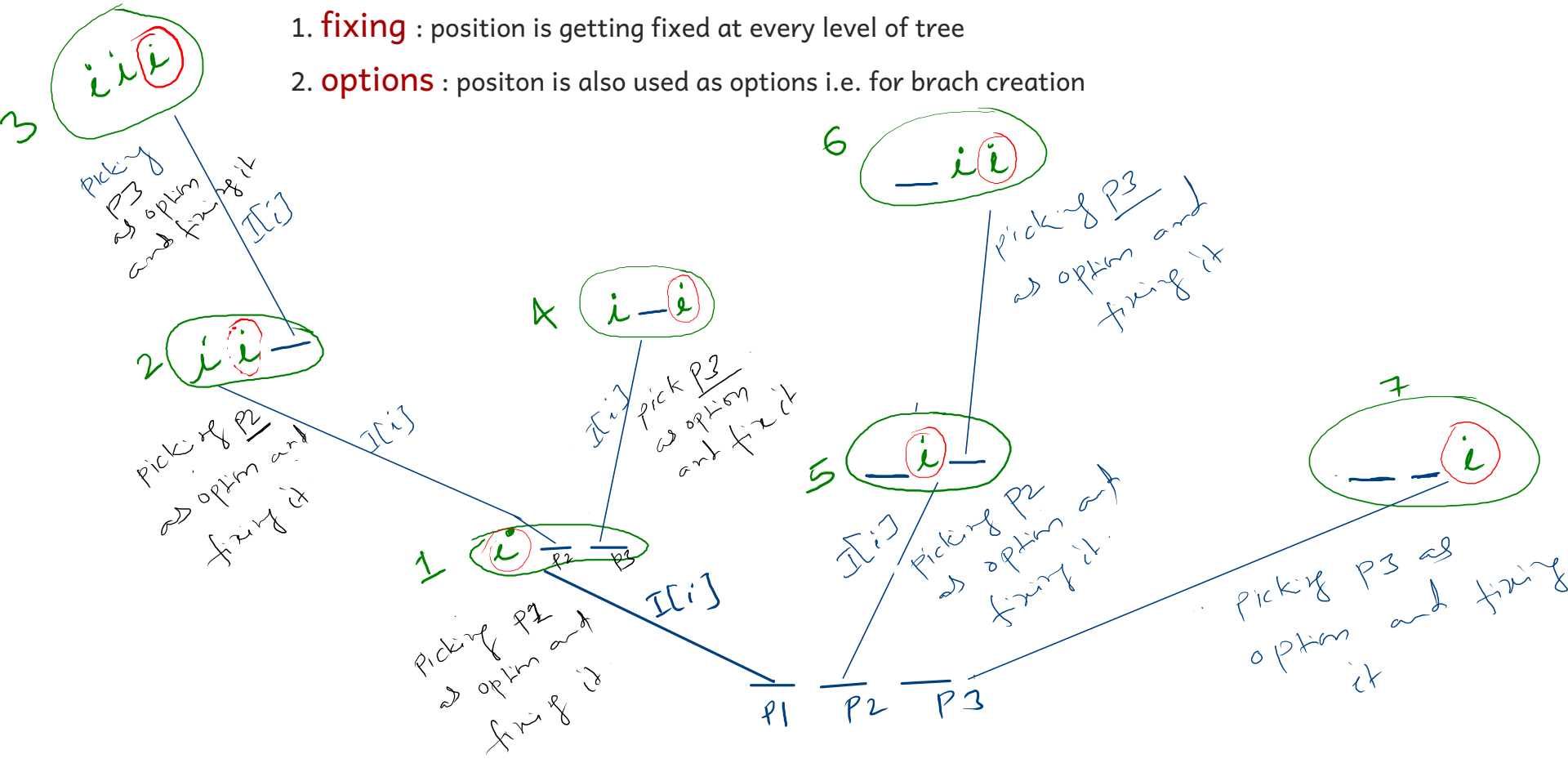
Note : Input elements are hard-attached with respective positions.

2. Pascal_Identity_Expansion based Include_Tree by fixing position where input_count <= position_count

Power set by placing 'i' at 'n' given positions

In pascal identity Expansion strategy position is used for both :

1. fixing : position is getting fixed at every level of tree
2. options : positon is also used as options i.e. for brach creation

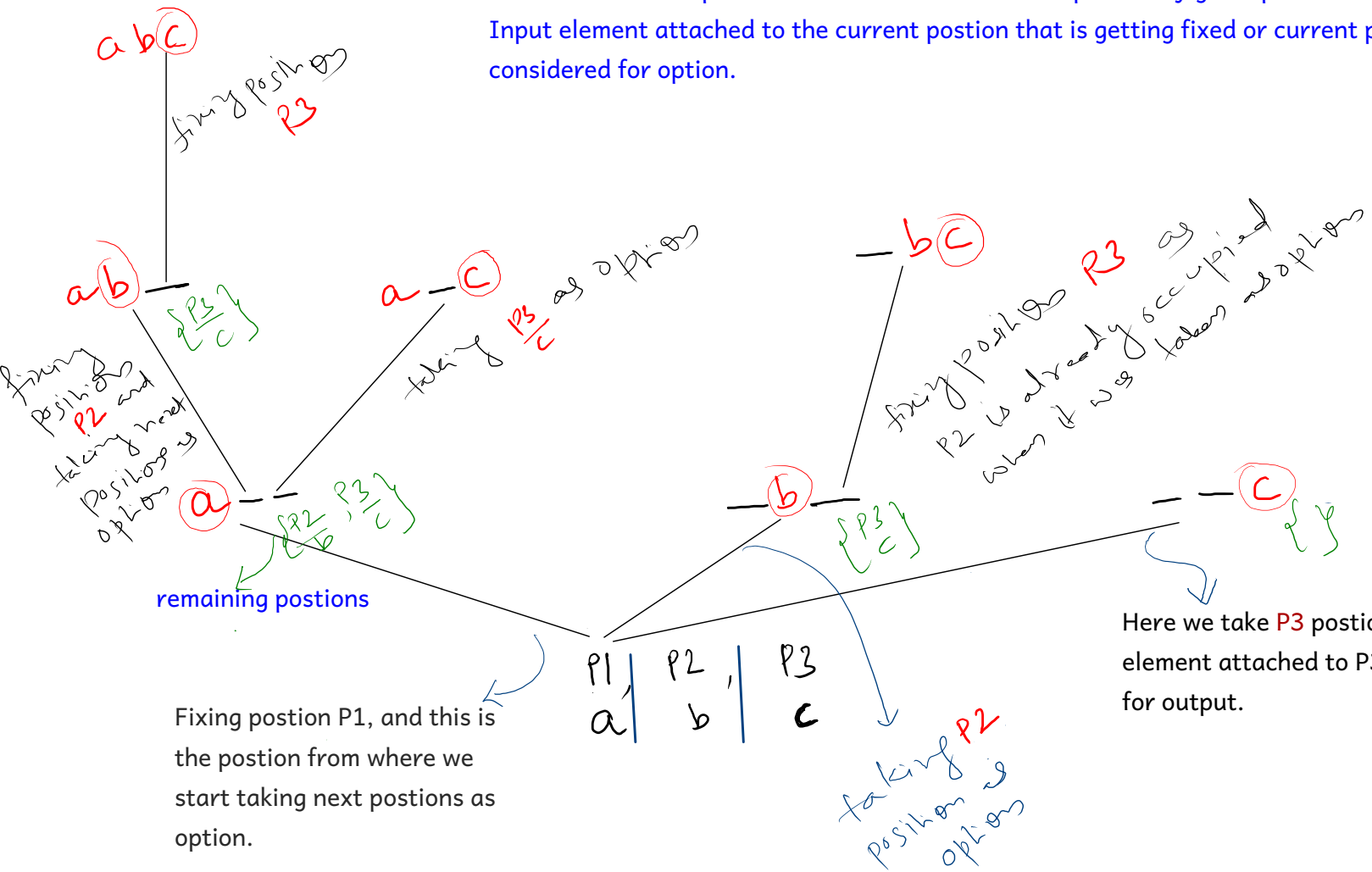


Power set by placing 'abc' at '3' given positions

Note : Input elements are hard-attached with respective positions.

Question: Which input element is considered for output at any given postion ?

Input element attached to the current postion that is getting fixed or current postion considered for option.



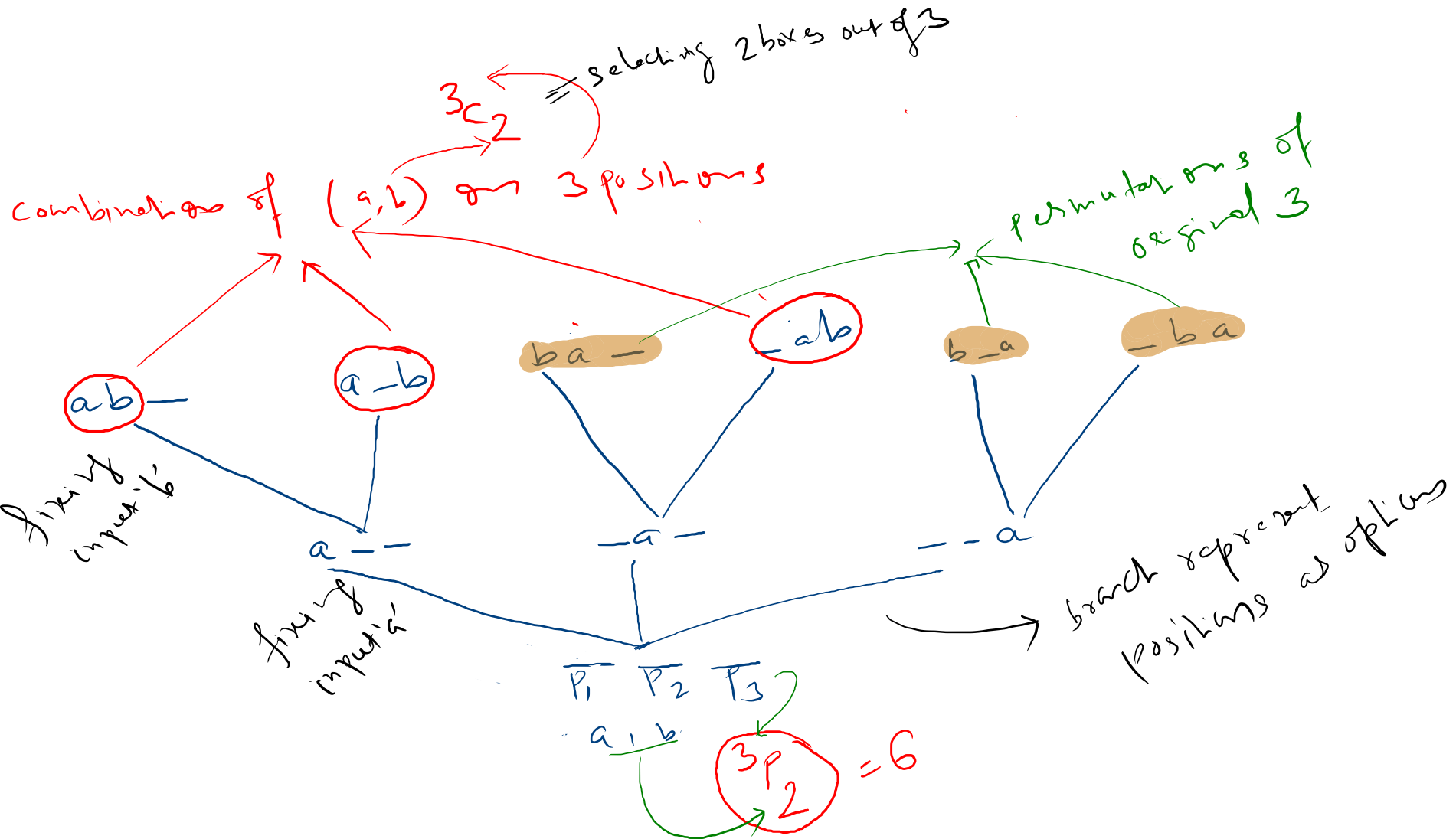
Fixing postion P1, and this is the postion from where we start taking next postions as option.

Here we take P3 postion as option, so the element attached to P3 i.e. 'C' is considered for output.

Relation between Permutation and Combination

Question : Print combination using prermuation strategy of fixing input and taking position as options

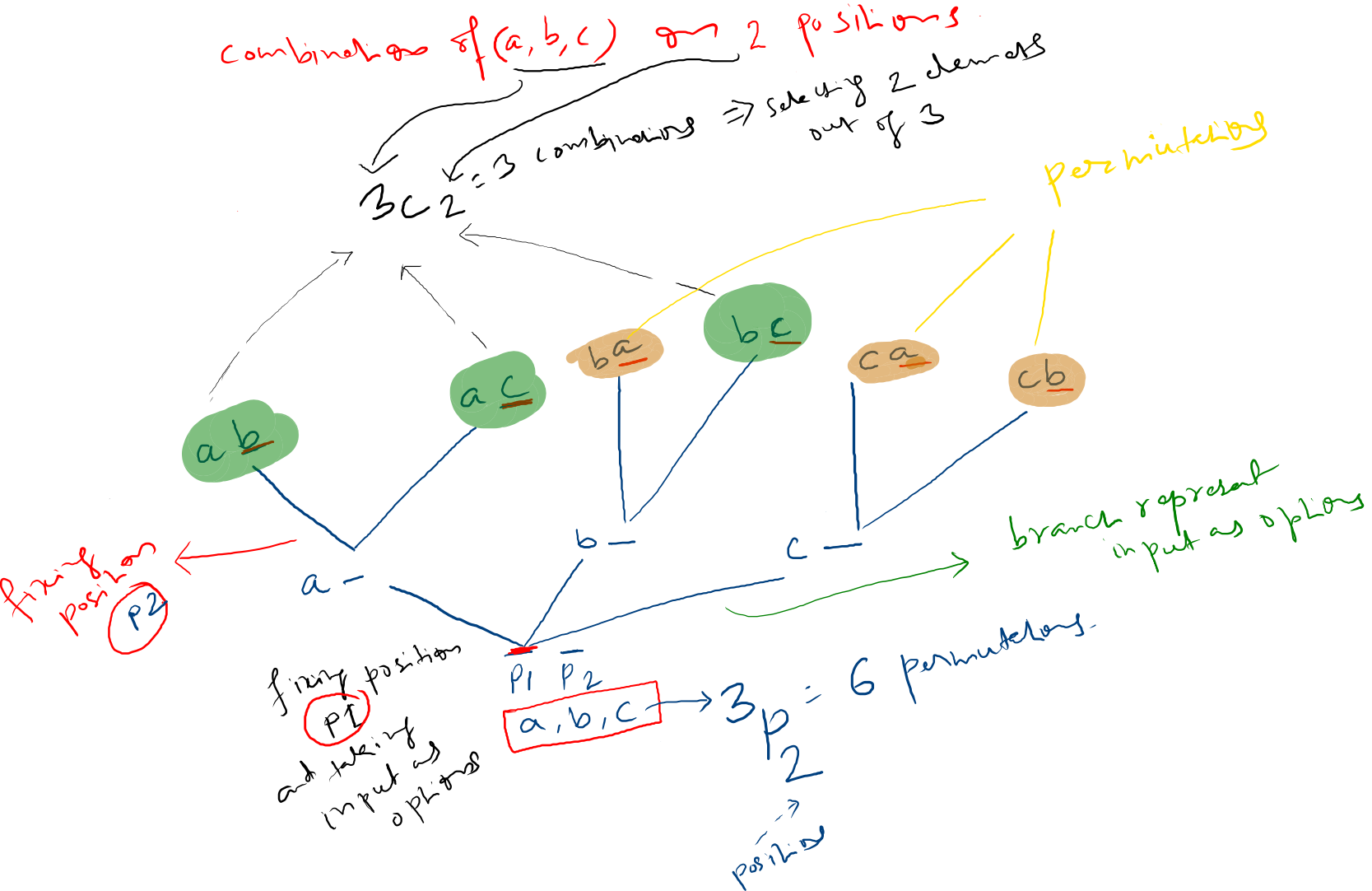
Note: If we allow the input to be placed only in increasing order or in lexicographic order i.e. we don't allow smaller input to be placed before large input, then we will get combinations from permutation strategy.



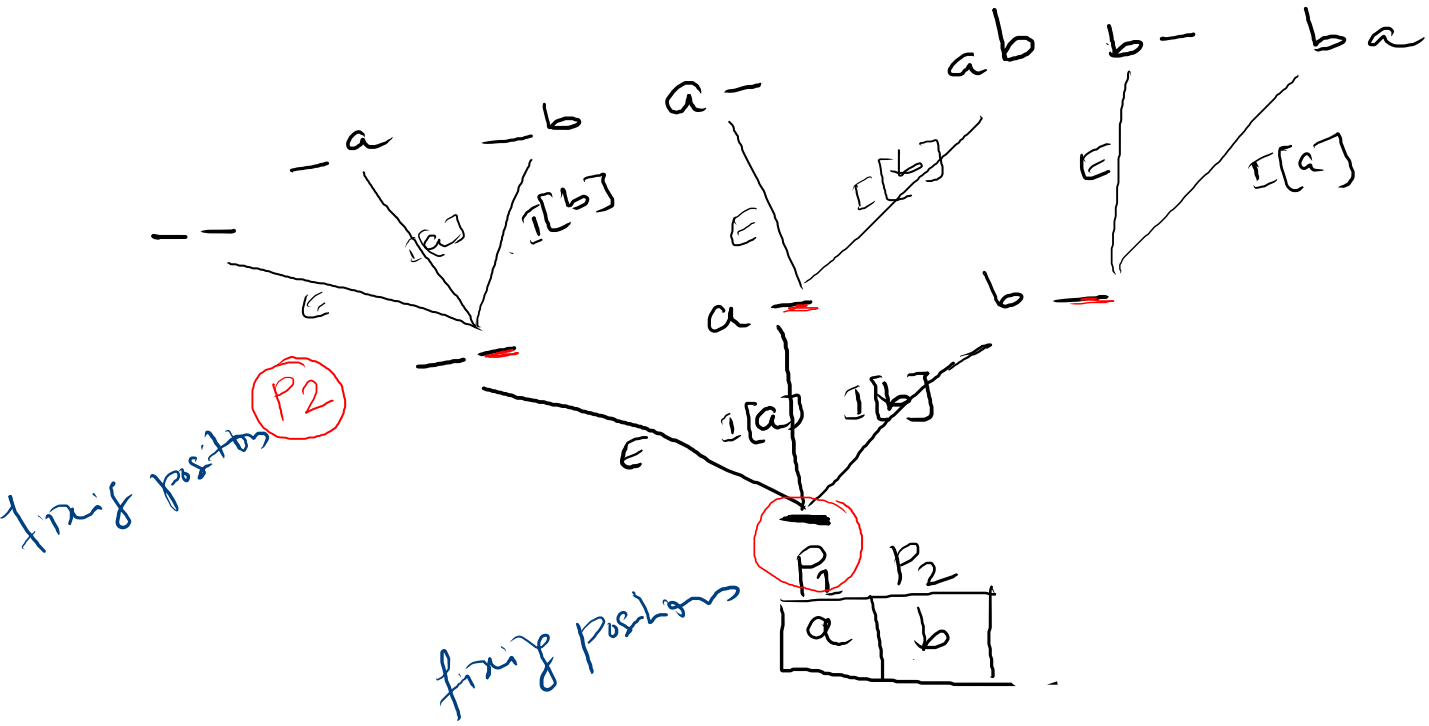
Relation between Permutation and Combination

Question : Print combination using prermuation strategy of fixing position and taking input as options

Note: If we allow the input to be placed only in increasing order or in lexicographic order i.e. we don't allow smaller input to be placed before large input, then we will get combinations from permutation strategy.



Generating Permutation using Pascal Identity Incldue-Exclude recursion-tree strategy



We are generating the permutations by fixing the positions and taking include-exclude as options.

Here in include step we pick all the remaining inputs as include-options.

Summing-Up the recursion generation methods

Recursion-Tree Generation methods used for generating PowerSet, Permutation and Combination

1. Pascal Identity Incldue-Exclude recursion-tree strategy
2. Pascal Identity Expansion recursion-tree strategy
3. Permutation recursion-tree strategy by fixing input
4. Permutation recursion-tree strategy by fixing position

PowerSet

The natural way to generate PowerSet recursion-tree is by using :

1. Pascal Identity Incldue-Exclude recursion-tree strategy
2. Pascal Identity Expansion recursion-tree strategy

Combination

The natural way to generate Combination recursion-tree is by using :

a. Pascal Identity Incldue-Exclude recursion-tree strategy

b. Pascal Identity Expansion recursion-tree strategy

Combination can be generated by tweaking the Permutation recursion-tree strategy:

a. Permutation recursion-tree strategy by fixing input

b. Permutation recursion-tree strategy by fixing position

Permutation

The natural way to generate Permutation recursion-tree is by using :

a. Permutation recursion-tree strategy by fixing input

b. Permutation recursion-tree strategy by fixing position

Permutation can be generated by tweaking the Pascal Identity Include-Exclude recursion-tree strategy:

-- Pascal Identity Incldue-Exclude recursion-tree strategy

Rule of thumb for picking what to be fixed and what to be taken as options in recursion tree while working with permutation-strategy

Observation:

1. For case where options to try at level get exhausted before leaf level:
 - We have to cover lots of corner cases with lots of if and buts.
2. For case where options to try at level get exhausted at leaf level or remain unexhausted:
 - Solution remain simple and straight-forward.

Rule of Thumb to pick the approach to tackle the recursive problem:

The parmeter whose count is smaller than the other need to be picked as fixing at levels.

Example: input :{a,b}; positions:{_,_,_,_} ;

Here, inputs count are 2 and positions count are 4 and since input_count is smaller than the position_count so we will pick 'input' as to fix at levels.