Know more about Recursion :

As we know that for every Recursive Call separate StackFrame would be created.
Working with Recursion is a costly job, as its going to occupy more space and also there is a limit for StackFrames .

Unhandily way of Recursion could cause StackOverFlow.

To avoid this we should use Recursion only when it is needed.

Thumb rule for a Recursion Problem is , for a Recursive problem , we always take a decision based on choice.
Here choice is either include or exclude.

For Ex : given input “ab” find out all the possible subsets.
Output : “” , “a”, “b” , “ab”

If you take a look at output

output	a	b
“”	✗	✗
a	✓	✗
b	✗	✓
ab	✓	✓

1. For subset “” empty String we excluded ‘a’ & ‘b’
2. For subset “a” we include ‘a’ & exclude ‘b’
3. For subset “b” we excluded ‘a’ & included ‘b’
4. For subset “ab” we include ‘a’ & ‘b’

Steps to work with Recursion :

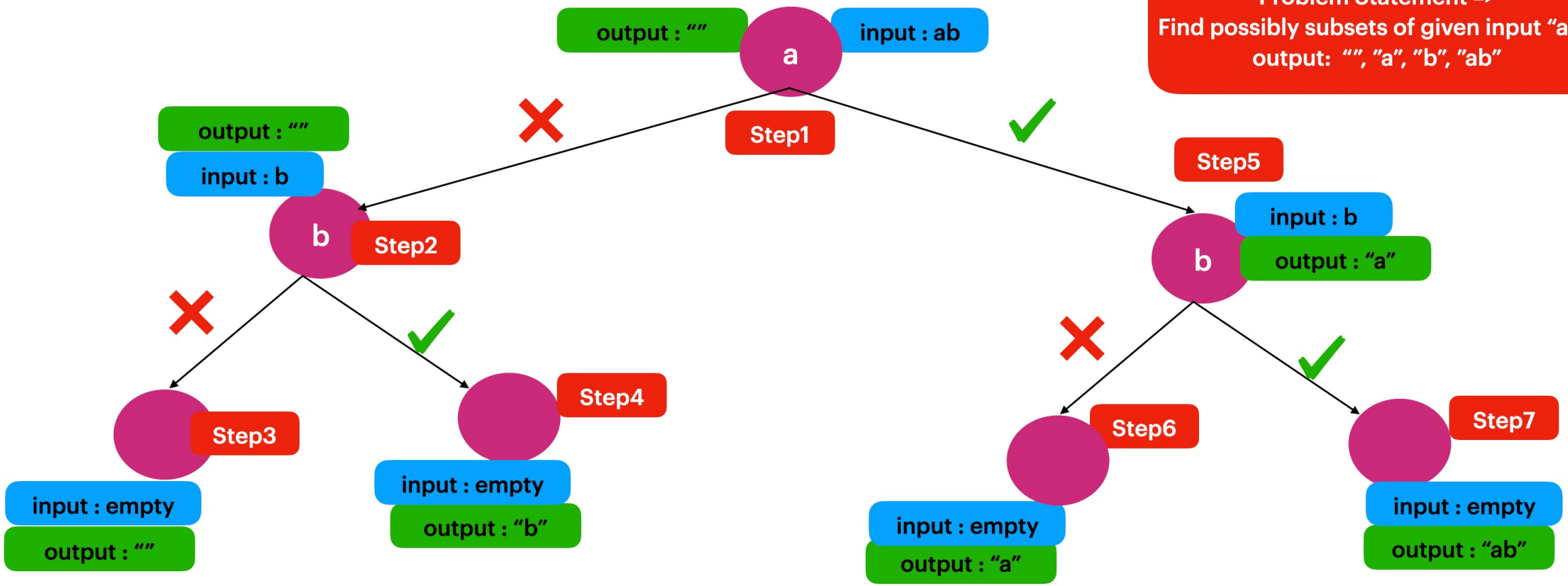
When we are working with Recursion first think about the base condition.

Base Condition => Always be the smallest possible Valid value for a given use case.

Recursion Tree => Draw a Recursion Tree , for every sub problem you will find two possible nodes , one is with exclude and other one is with include.

Write the Code => We can solve any Recursion problem with two lines of code.

Problem Statement =>
Find possibly subsets of given input “ab”.
output: “”, ”a”, ”b”, ”ab”

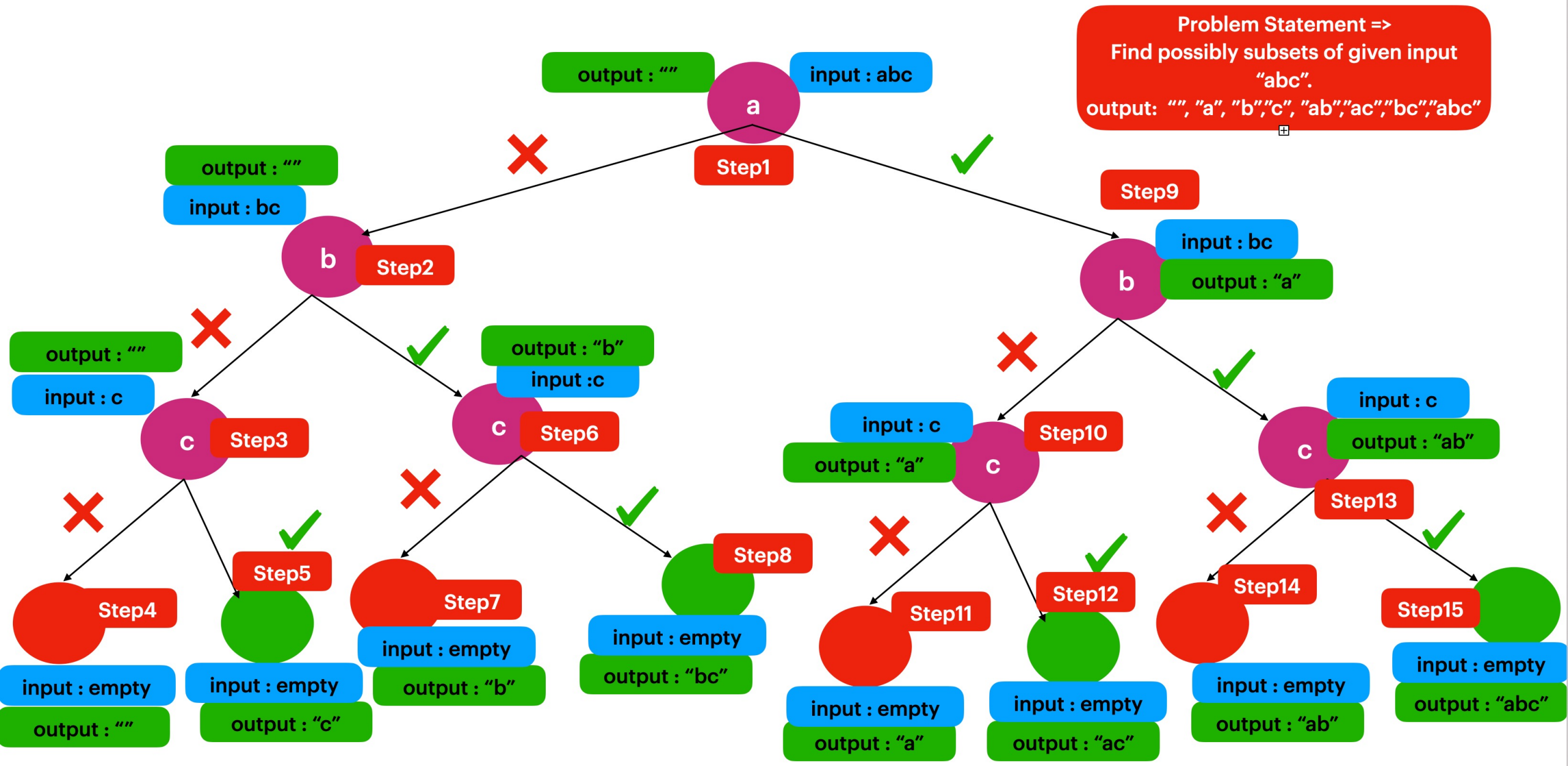


Time Complexity : 2^n
Space Complexity : O(n)

```
public void subSets(String input, String output)
{
if(input.isEmpty())
{
System.out.println(output);
return;
}

//Exclude
subSets(input.substring(1), output);

//Include
subSets( input.substring(1), output+input.charAt(0) );
}
```


Problem Statement =>
Find possibly subsets of given input "abc".
output: "", "a", "b","c", "ab","ac","bc","abc"
output : ""
a
input : abc
Step1
output : ""
input : bc
b
Step2
Step9
input : bc
b
output : "a"
output : ""
input : c
c
Step3
output : "b"
input :c
c
Step6
input : c
output : "a"
c
Step10
input : c
c
output : "ab"
Step13
Step4
input : empty
output : ""
Step5
input : empty
output : "c"
Step7
input : empty
output : "b"
Step8
input : empty
output : "bc"
Step11
input : empty
output : "a"
Step12
input : empty
output : "ac"
Step14
input : empty
output : "ab"
Step15
input : empty
output : "abc"