Get Subsequences

Monday, 29 January 2024 10:47 AM

Str = "abc" → Length : n

Subsequences :

need not be continuous

2^n.

_ _ _	0 0 0
_ _ c	0 0 1
_ b _	0 1 0
_ b c	0 1 1
a _ _	1 0 0
a _ c	1 0 1
a b _	1 1 0
a b c	1 1 1

Substring :

need to be continuous

$\frac{n(n+1)}{2}$

a
ab
abc

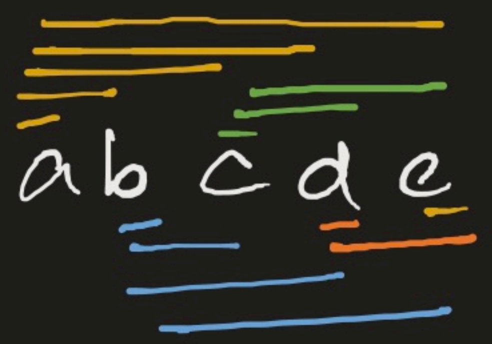
b
bc

c

Proofs:

1. Substrings :

$$\# = n + (n-1) + (n-2) \cdots + 1$$



$$\# = \frac{n(n+1)}{2}$$

2. Sub sequence :

$$\underline{2} \quad \underline{2} \quad \underline{2} \quad \underline{2} \quad \underline{2} \quad \underline{2}$$

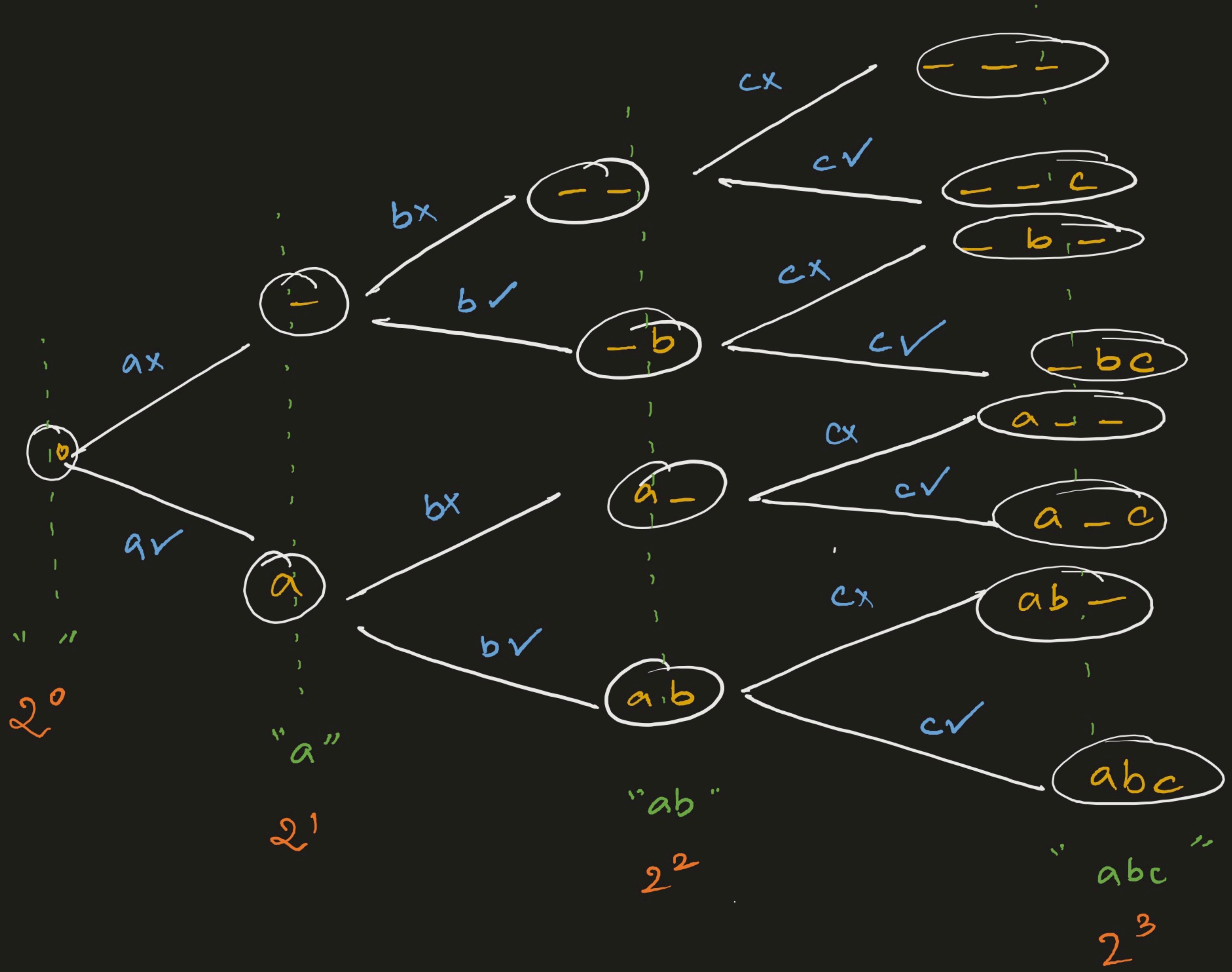
a b e d e f.

Every character has 2 options, to be a part of our ssq. or to not be a part of our ssq.

└> present

└> not-present.

2^n ssq


0
ax
a✓
_
a
bx
b✓
bx
b✓
_ _
_ b
a _
a b
cx
c✓
cx
c✓
cx
c✓
cx
c✓
_ _ _
_ _ c
_ b _
_ b c
a _ _
a _ c
a b _
a b c
" "
2⁰
"a"
2¹
"ab"
2²
"abc"
2³

Expectation.

get ssq.(abc)

_ _ _

_ _ c

_ b _

_ b c

a _ _

a _ c

a b _

a b c

Faith.

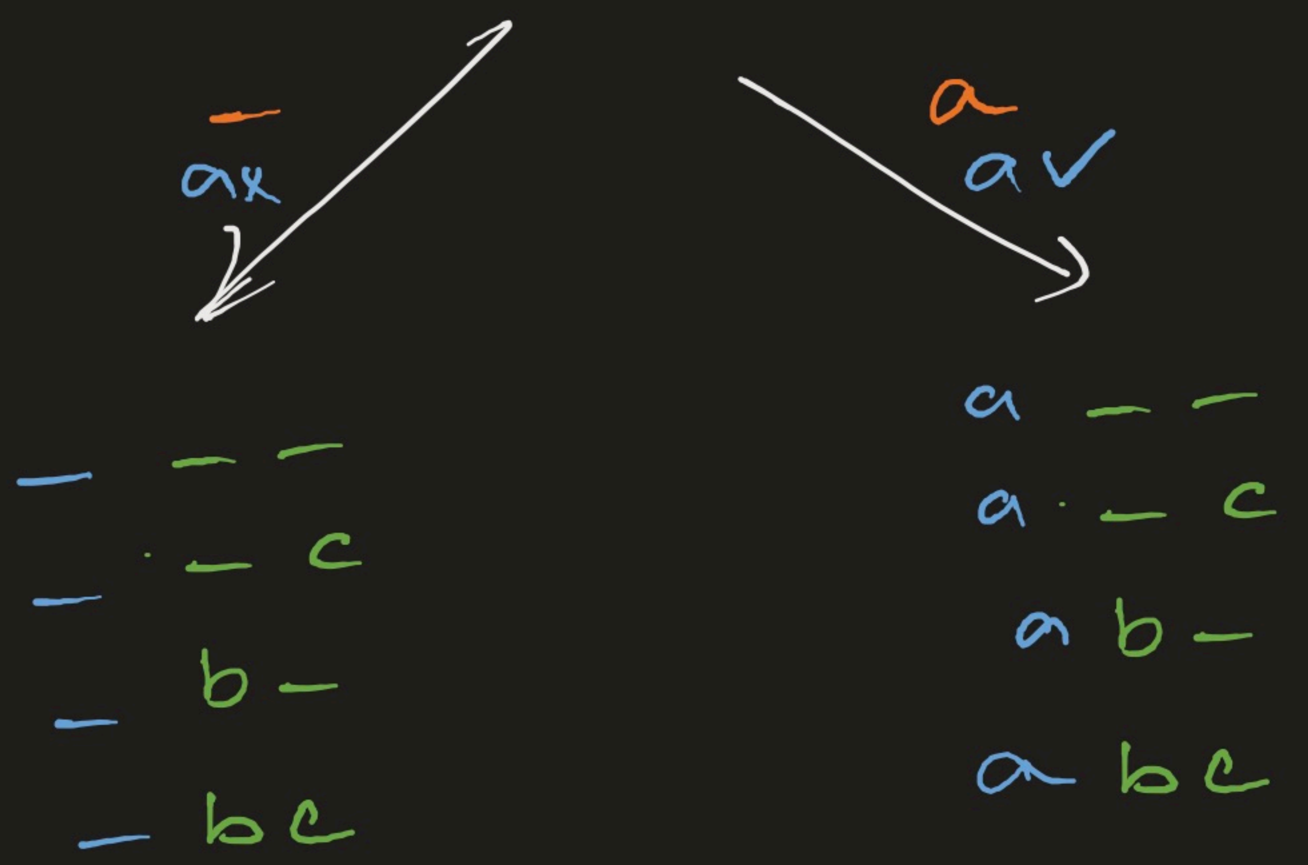
getssq(bc)

_ _

_ c

b _

bc.



_ _ _

_ _ c

_ b _

_ bc

a _ _

a _ c

a b _

a bc

Meet expectations using the faith

getssq (abc)

||

_ getssq (bc)

+

a getssq (bc)

Meeting expectations using faith : Can we generate all the ssq. of "abc" if somebody has already given me all the ssq of "bc" ?

Base Case :

input = " . " ← empty string.

getssq() = {

[ ] empty vector ✗

[" "] a vector containing an empty string. ✓

An empty string : a string containing no characters will still

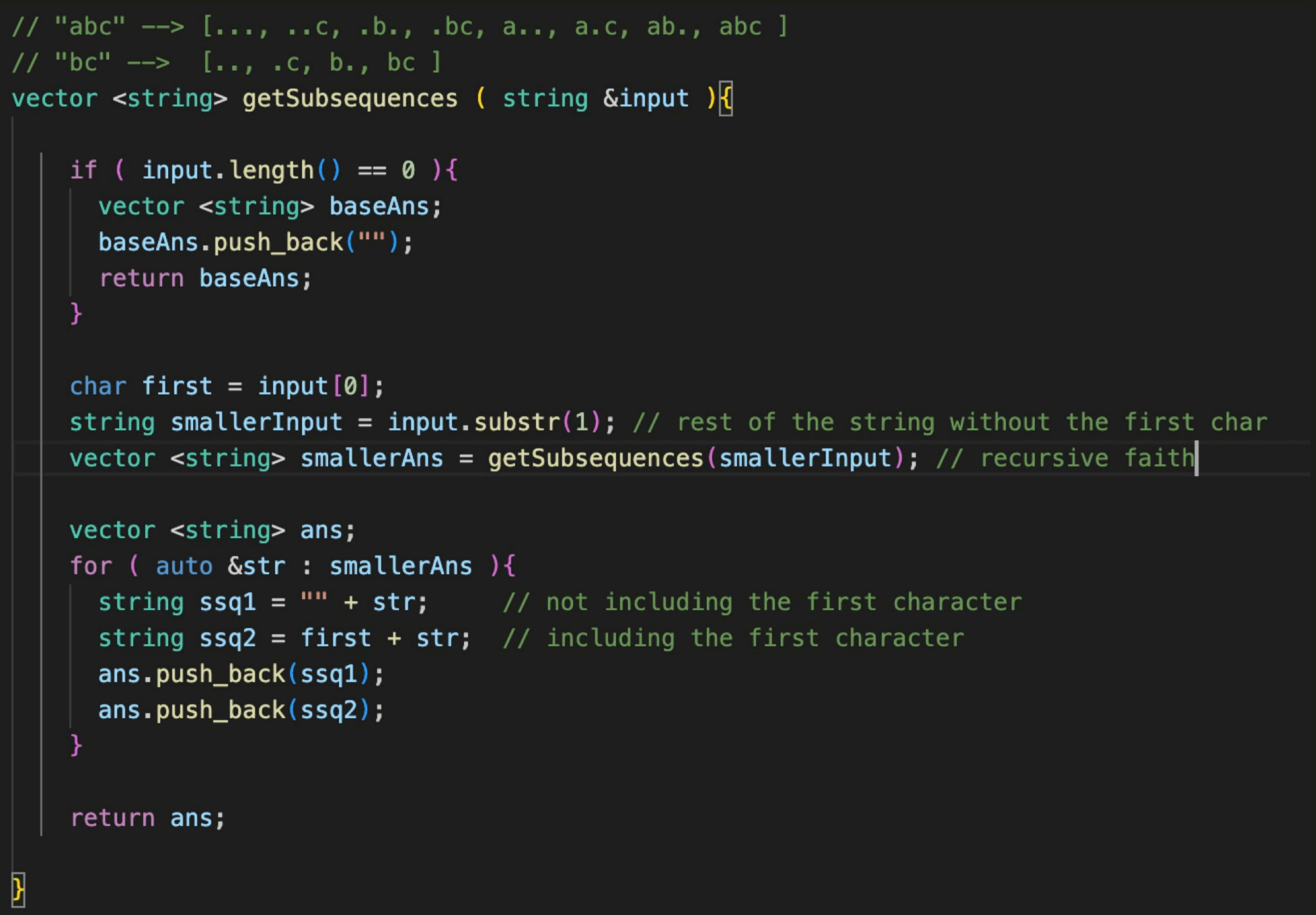
An empty string : a string containing no characters will still has a subsequence which is empty. i.e the subsequence wont have any character. But an empty ssq. is still a valid ssq. Its not has we arent getting any subsequences, instead we are getting a subsequence which contains no characters.

'n' char string $\xrightarrow{ssq.}$ 2^n ssq.

if $n = 0$ $\longrightarrow$ $2^0 = \underline{\underline{1}}$

[ ] → 0

[" "] → 1



```cpp
// "abc" --> [..., ..c, .b., .bc, a.., a.c, ab., abc ]
// "bc" -->  [.., .c, b., bc ]
vector <string> getSubsequences ( string &input ){

    if ( input.length() == 0 ){
      vector <string> baseAns;
      baseAns.push_back("");
      return baseAns;
    }

    char first = input[0];
    string smallerInput = input.substr(1); // rest of the string without the first char
    vector <string> smallerAns = getSubsequences(smallerInput); // recursive faith

    vector <string> ans;
    for ( auto &str : smallerAns ){
      string ssq1 = "" + str;     // not including the first character
      string ssq2 = first + str;  // including the first character
      ans.push_back(ssq1);
      ans.push_back(ssq2);
    }

    return ans;

}
```