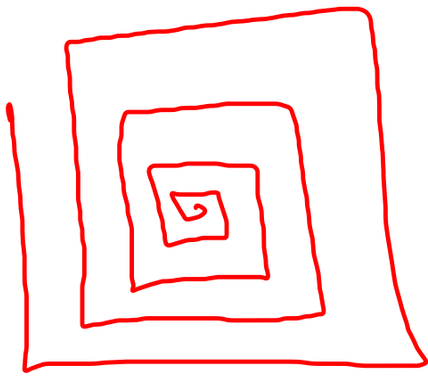
3
5
11 12 13 14 15
21 22 23 24 25
31 32 33 34 35

Output

11
21
31
32
33
34
35
25
15
14
13
12
22
23
24

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 11 | 12 | 13 | 14 | 15 |
| 1 | 21 | 22 | 23 | 24 | 25 |
| 2 | 31 | 32 | 33 | 34 | 35 |



n = 3
m = 5

minr = 0
minc = 0
maxr = n-1
maxc = m-1

minc → 0, 1, 2, 3, 4 (maxc)

minr → 0

maxr → 2

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 11 | 12 | 13 | 14 | 15 |
| 1 | 21 | 22 | 23 | 24 | 25 |
| 2 | 31 | 32 | 33 | 34 | 35 |

lw, bw, rw, tw

11
21
31
32
33
34
35
25
15

```
// right wall
for(int i = maxr-1,j = maxc ; i >= minr ; i--){
    System.out.println(mat[i][j]);
}
```

1 4 0

= maxc ; j++){
;

Count = 31

minc

maxc

minr

maxr

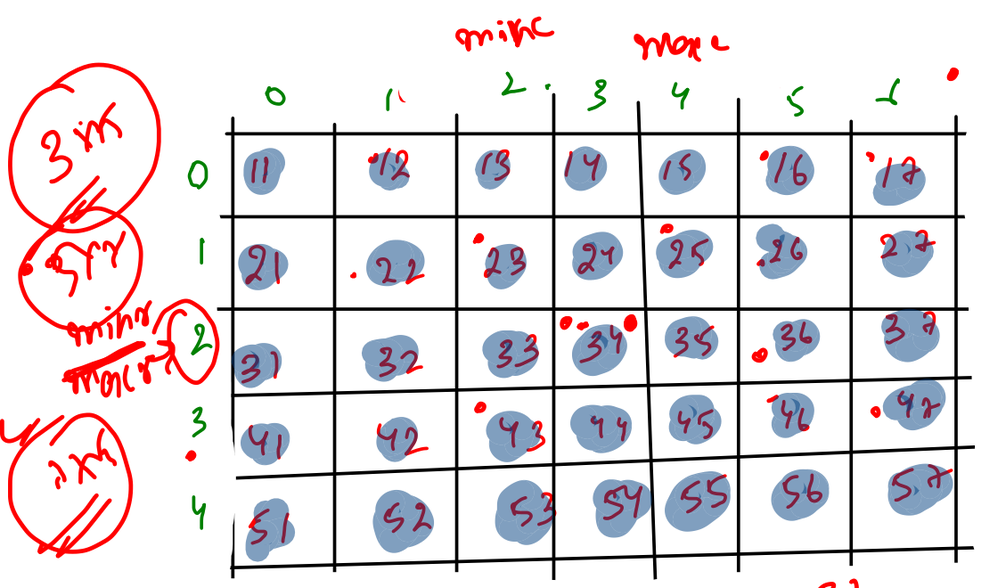
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 1 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 2 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| 3 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 4 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |

| | | | |
|---|---|---|---|
| 11 | 52 | 47 | 11 |
| 21 | 53 | 37 | 15 |
| 31 | 54 | 27 | 14 |
| 41 | 55 | 17 | 13 |
| 51 | 56 | | 12 |
| | 57 | | |

minr ++;
minc ++;
maxr --;
maxc --;

```
int minr = 0 , minc = 0, maxr = n-1, maxc = m-1;
```

```
// left wall
for(int i = minr,j = minc ; i <= maxr ; i++){
    System.out.println(mat[i][j]);
}

// bottom wall
for(int i = maxr, j = minc+1; j <= maxc ; j++){
    System.out.println(mat[i][j]);
}

// right wall
for(int i = maxr-1,j = maxc ; i >= minr ; i--){
    System.out.println(mat[i][j]);
}

// top wall
for(int i = minr,j = maxc-1 ; j >= minc+1 ; j--){
    System.out.println(mat[i][j]);
}
```




Count: ~~27~~ ~~29~~ ~~32~~ ~~33~~ 35

tEle = 35

5 x 7

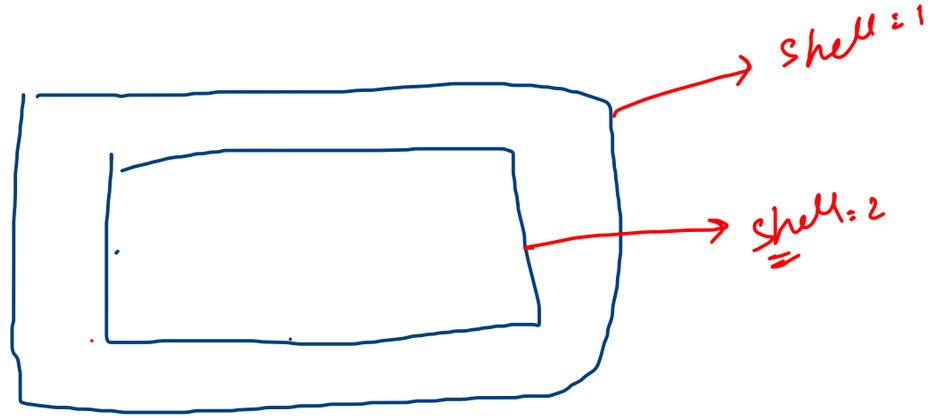
check
A → check
B → check
C → check
D → check

| | | |
|---|---|---|
| 11 | 27 | 32 |
| 21 | 17 | 26 |
| 31 | 16 | 25 |
| 41 | 15 | 24 |
| 51 | 14 | 23 |
| 52 | 13 | 33 |
| 53 | 12 | 34 |
| 54 | 22 | 35 |
| 55 | 32 | |
| 56 | 42 | 34 |
| 57 | 43 | |
| 47 | 44 | |
| 37 | 45 | |
| | 41 | |

```java
public static void spiralDisplay(int mat[][]){
    int n = mat.length , m = mat[0].length;
    int minr = 0 , minc = 0, maxr = n-1, maxc = m-1, count = 0 , tEle = n*m;
    while(count < tEle){
        // left wall
        for(int i = minr,j = minc ; i <= maxr ; i++){
            System.out.println(mat[i][j]);
            count++;
        }

        // bottom wall
        for(int i = maxr, j = minc+1; j <= maxc && count < tEle; j++){
            System.out.println(mat[i][j]);
            count++;
        }

        // right wall
        for(int i = maxr-1,j = maxc ; i >= minr && count < tEle; i--){
            System.out.println(mat[i][j]);
            count++;
        }

        // top wall
        for(int i = minr,j = maxc-1 ; j >= minc+1 && count < tEle; j--){
            System.out.println(mat[i][j]);
            count++;
        }

        minr++;
        minc++;
        maxr--;
        maxc--;
    }
}
```

n : 5
m : 7 ] ⇒ minimum = 5/2 = 2

Input




Output

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 1 | 21 | 25 | 26 | 36 | 46 | 45 | 27 |
| 2 | 31 | 24 | 33 | 34 | 35 | 44 | 37 |
| 3 | 41 | 23 | 22 | 32 | 42 | 43 | 47 |
| 4 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |

k : 3

① fill 1D array

arr[idx] = mat[i][j]

Kn

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 32 | 42 | 43 | 44 | 45 | 46 | 36 | 26 | 25 | 24 | 23 |

spiral display

② rotate

P1 | P2

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 32 | 23 | 22 | 32 | 42 | 43 | 44 | 45 | 41 | 36 | 26 |

reverse(P2)
reverse(P1)
reverse(arr)

⇒ n

③ fill 2D array

mat[i][j] = arr[idx]

Challenge

① Shell

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 1 | 21 | 25 | 26 | 36 | 46 | 45 | 27 |
| 2 | 31 | 24 | 33 | 34 | 35 | 44 | 37 |
| 3 | 41 | 23 | 22 | 32 | 42 | 43 | 47 |
| 4 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |

Output

Input

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 1 | 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 2 | 31 | 32 | 33 | 34 | 35 | 36 | 37 |
| 3 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 4 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |

String

StringBuilder

structure

"hello"

1. Usage
2. memory management
3. Performance

ArrayList

String

str.length() = 5

str = "hello"

0 1 2 3 4

idx = (0, length()-1)

str.charAt(2) => 'l'

1. Declaration / definition
2. Output
3. func.
   - length() -> no. of characters
   - charAt(idx) => character according to idx.
   - Concat.
     - string + ____ = string

char → operation → Compiler → ascii

character

ch = "M"

('A' ≤ ch && ch ≤ 'Z') if

65 ≤ 77 ≤ 90

('a' ≤ ch && ch ≤ 'z') if

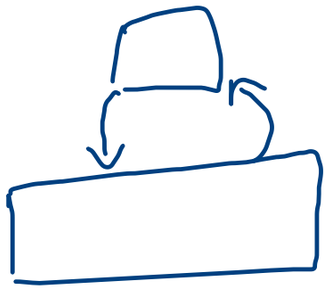
('0' ≤ ch && ch ≤ '9') if

char → ascii

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|---|---|---|
| 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

char = 2 byte

int = 4 byte



wwwwaaadexxxxxx

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

res = "wade x"

```
public static String compression1(String str){
    String res = str.charAt(0) + "";
    for(int idx = 1; idx < str.length(); idx++){
        char curr = str.charAt(idx);
        char prev = str.charAt(idx-1);

        if(curr != prev){
            res = res + curr;
        }
    }
    return res;
}
```

= "aabbccc de aa ghiiijkll"

eod

git

var

length

length()

1

wwwwaaadexxxxxx

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

my = w4a3dex6

Count = 1 2 3 4 1 2 3 1 2 3 4 5 6

→ Spiral Display

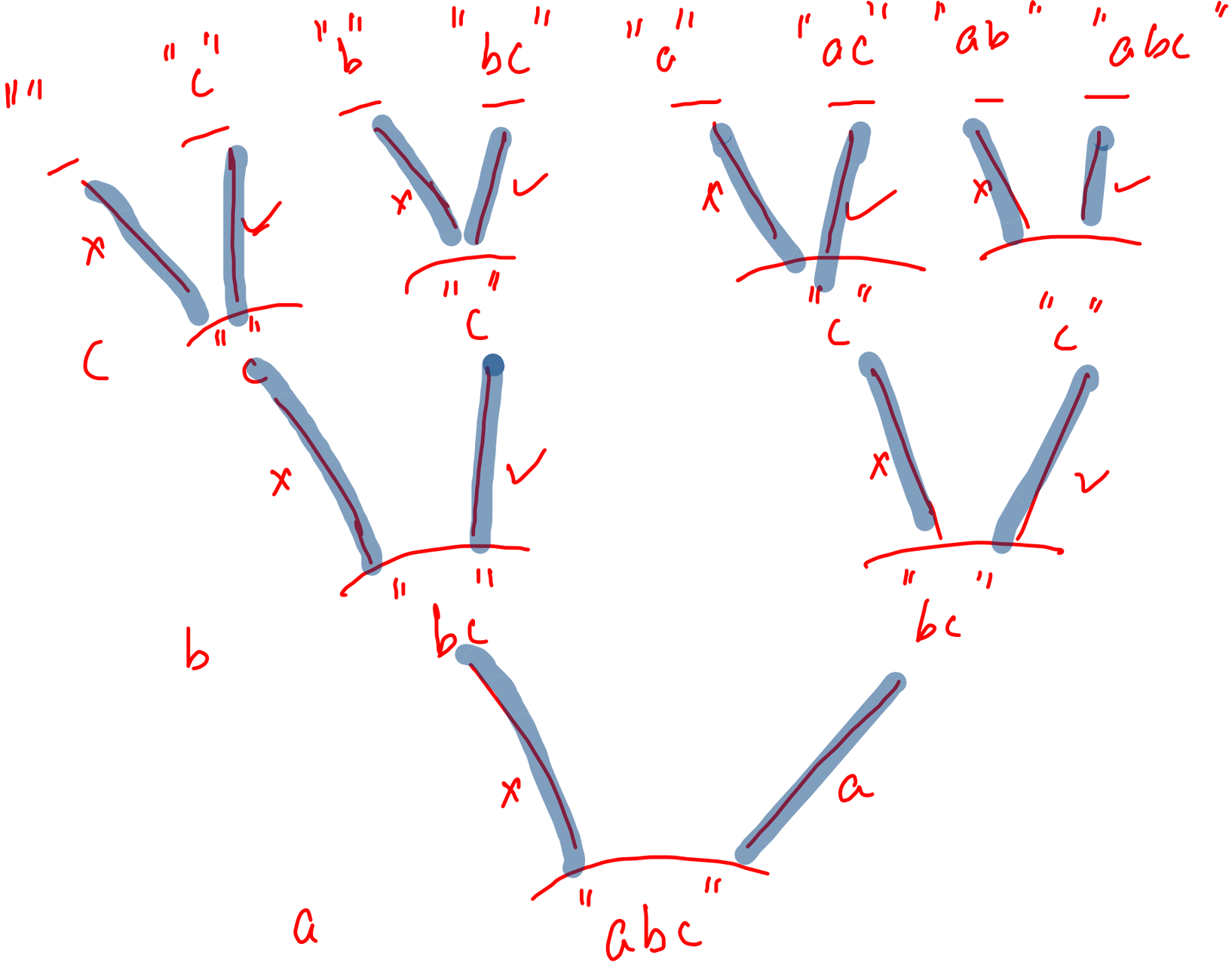
→ String Rotate

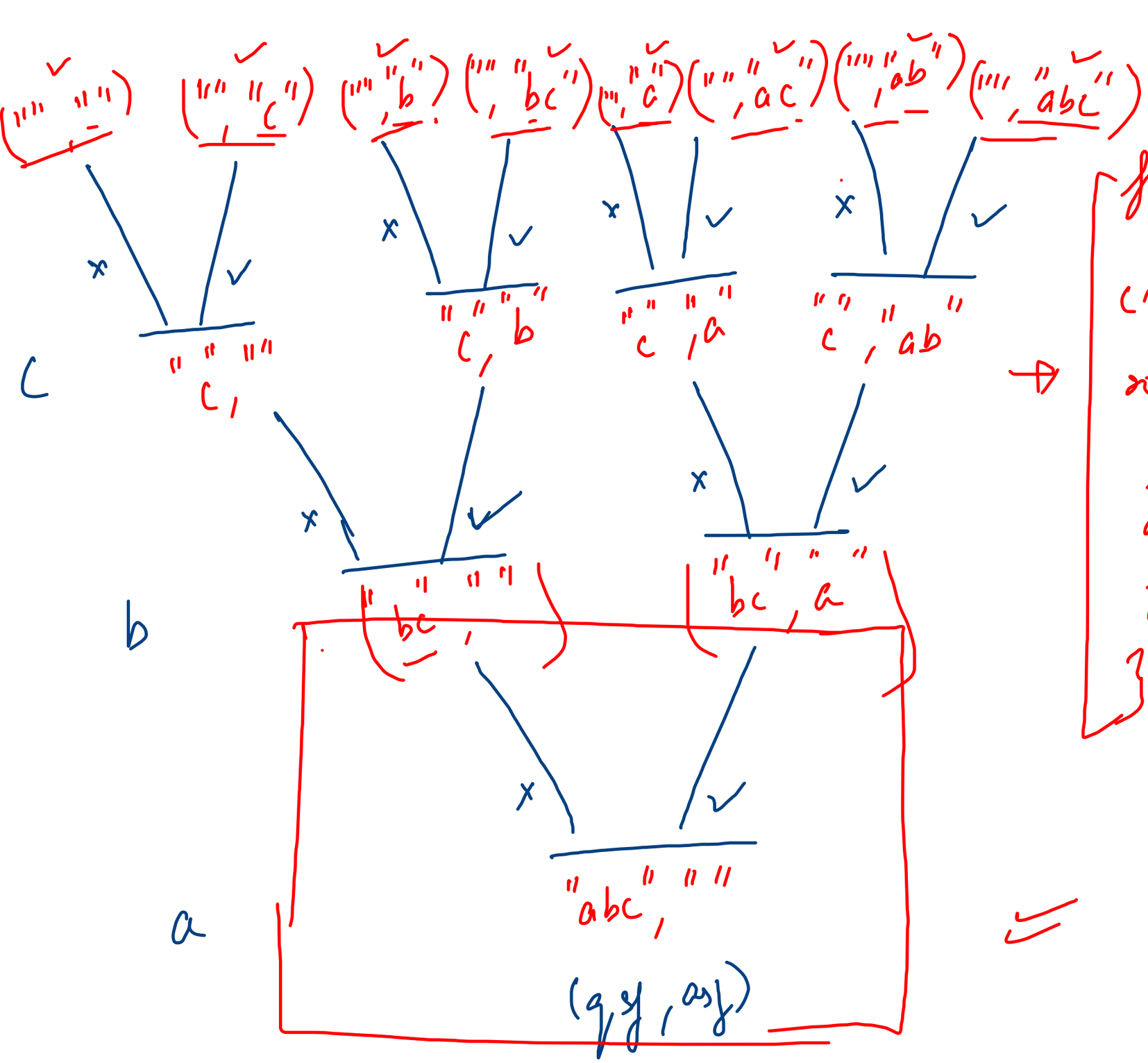
→ String intro, ASCII → char

→ Compression-1, Compression-2

Batch 1

Recursion or the way up



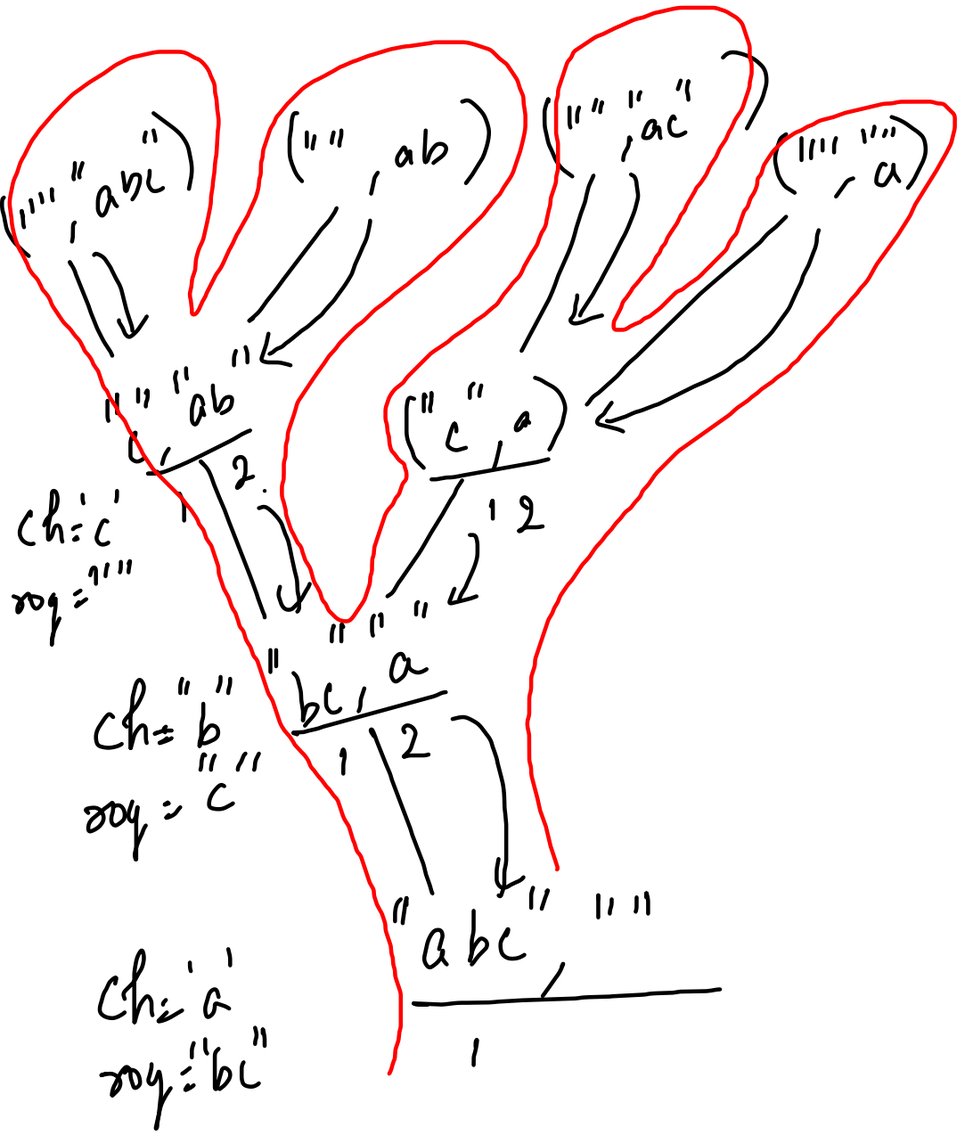



void

```
fun(qsf, asf) {
ch ← str.charAt(0)
roq ← str.substring(1)
fun(roq, asf);
fun(roq, asf+ch);
}
```

Subsequence → Inc, Exc

a bc




```java
public static void printSS(String qsf, String asf) {
    if(qsf.length() == 0){
        System.out.println(asf);
        return;
    }
    char ch = qsf.charAt(0);
    String roq = qsf.substring(1);

    printSS(roq,asf+ch);    — 1
    printSS(roq,asf);       — 2
}
```

abc

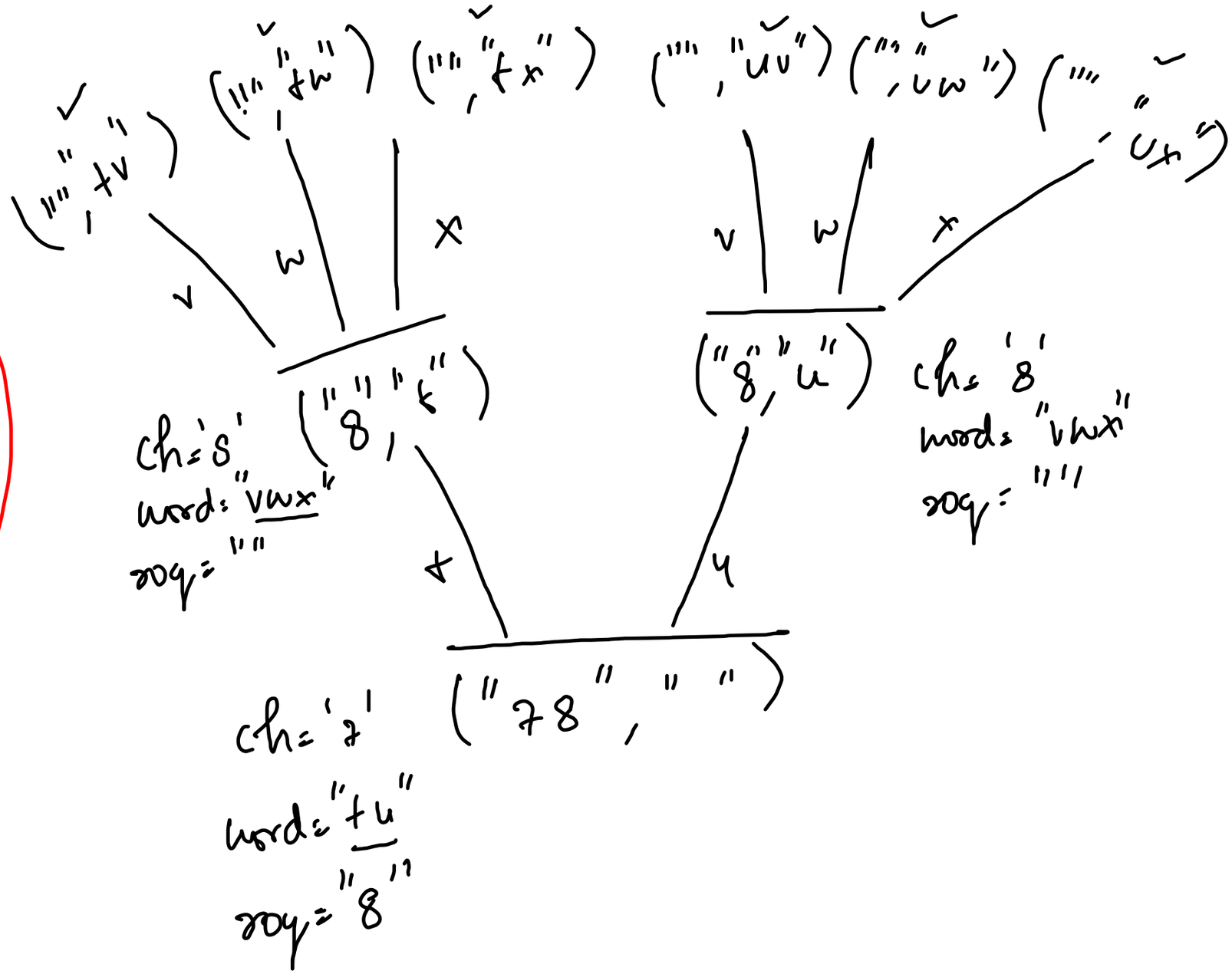
ab

ac

a

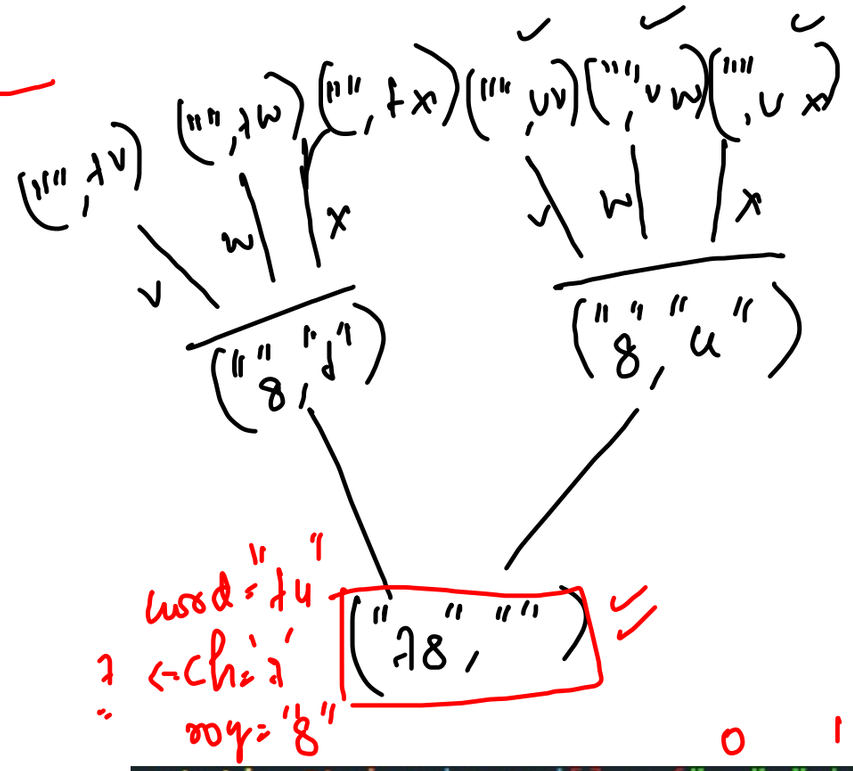
0 -> .;
1 -> abc
2 -> def
3 -> ghi
4 -> jkl
5 -> mno
6 -> pqrs
7 -> tu
8 -> vwx
9 -> yz

78




```
static String keypad[] = {".;","abc","def","ghi" , "jkl","mno","pqrs","tu","vwx","yz"};
```

0 1 2 3 4 5 6 7 8 9




ch = '8', idx = 8
roq = ""
word = vwx
0 1 2

tv
tw
tx
uv
uw
ux

"ta"

idx = ~~0 → 3~~ : 0, 1, 2

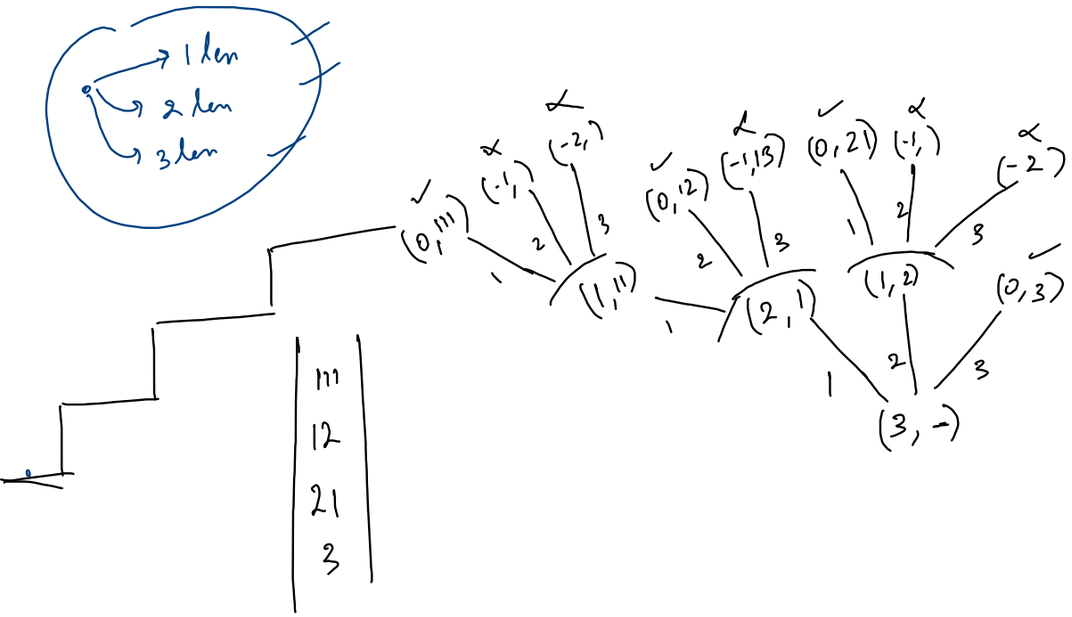
word = "abc"

asf = "t"

submit

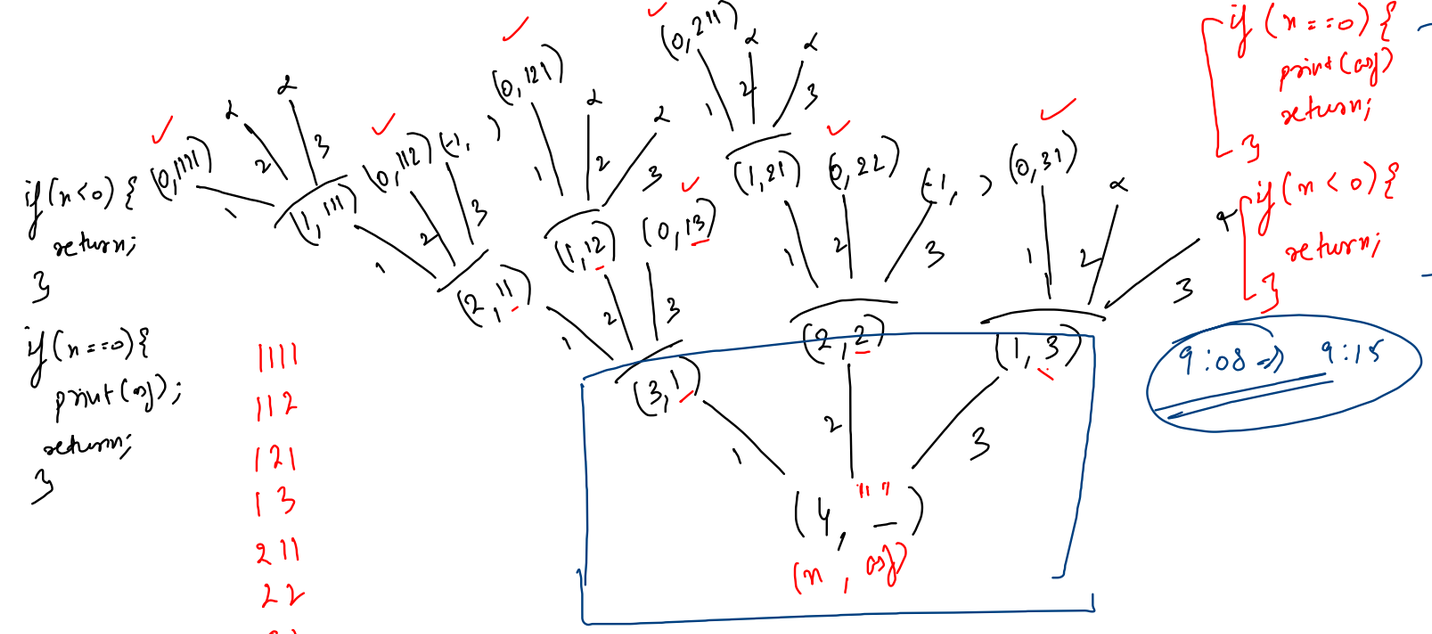
0 1 2 3 4 5 6 7 8 9

```
static String keypad[] = {".;","abc","def","ghi" , "jkl","mno","pqrs","tu","vwx","yz"};
public static void printKPC(String qsf,String asf) {
    if(qsf.length() == 0){
        System.out.println(asf);
        return;
    }

    char ch = qsf.charAt(0);
    String roq = qsf.substring(1);
    int idx = Integer.parseInt(ch+"");
    String word = keypad[idx];

    for(int idx = 0 ; idx < word.length() ; idx++){
        printKPC(roq,asf+word.charAt(idx));
    }
}
```



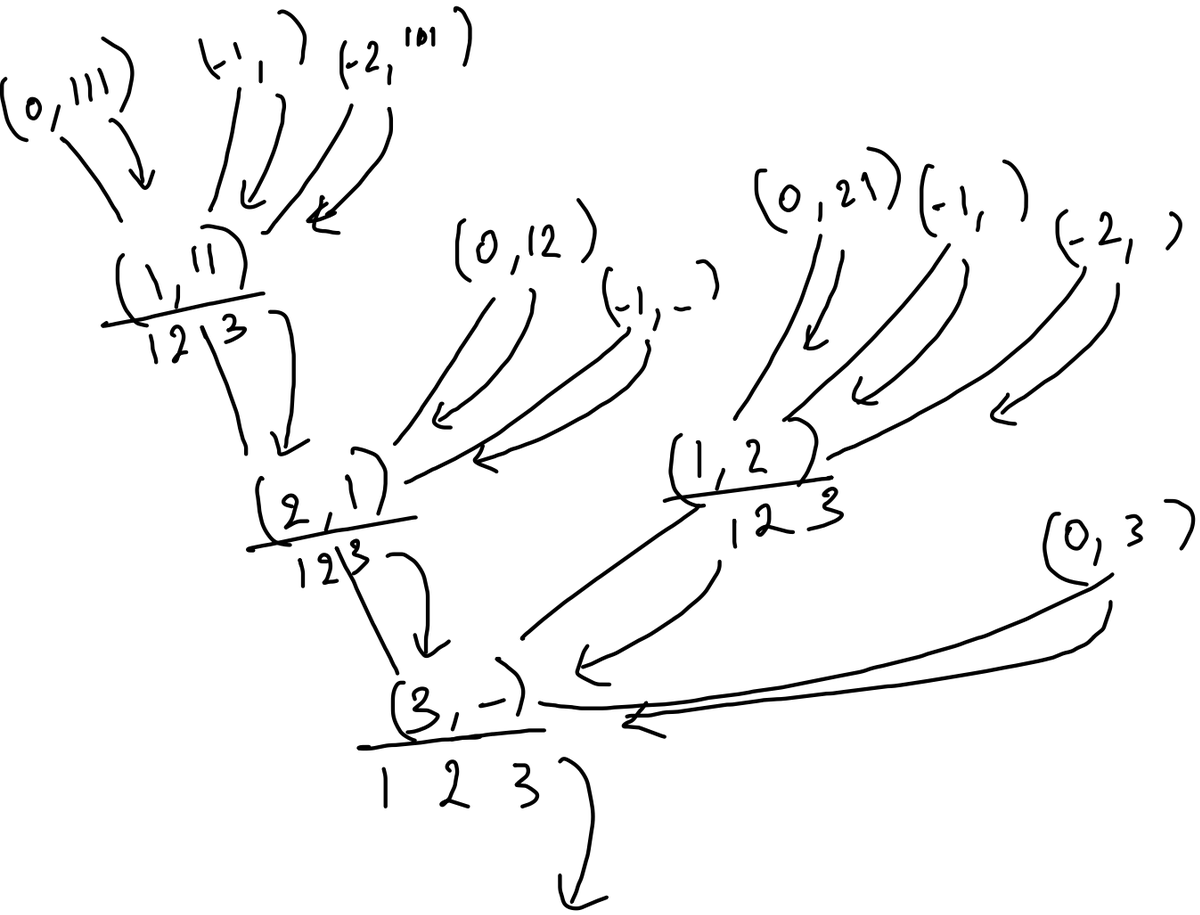

```
if(n<0){
  return;
}
if(n==0){
  print(ans);
  return;
}
```

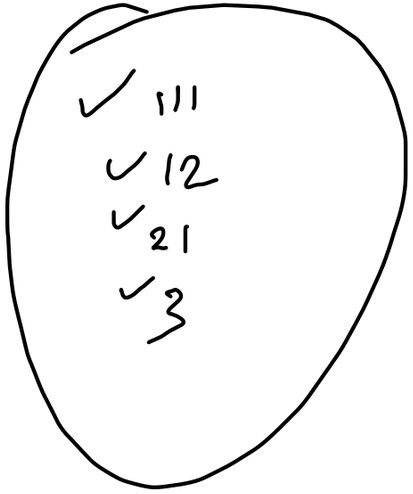
1111
112
121
13
211
22
31




```
if(n==0){
  print(ans)
  return;
}
if(n<0){
  return;
}
```

9:08 => 9:15



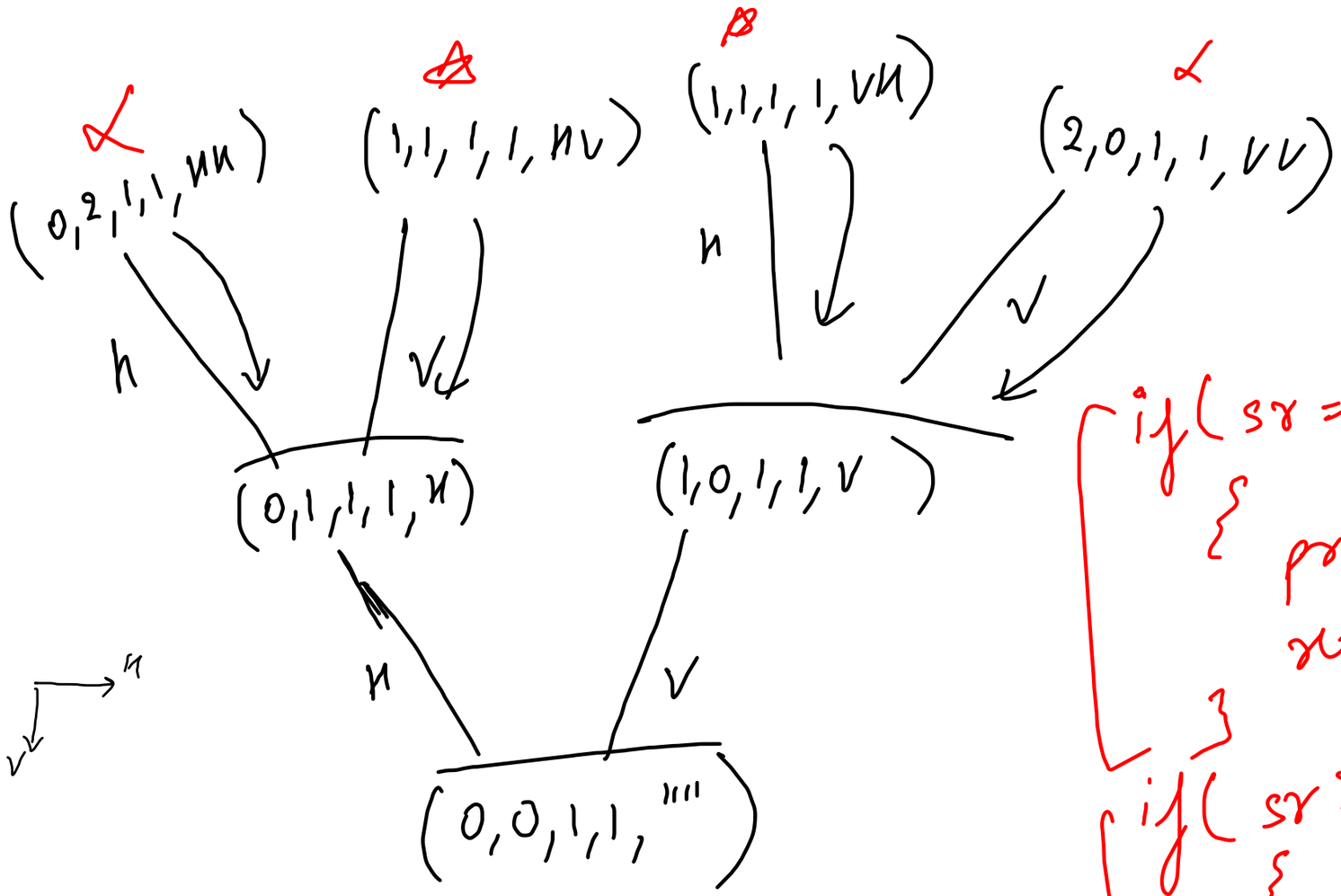



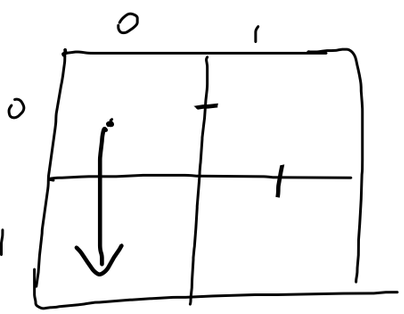

```java
public static void printStairPaths(int n, String psf) {
    if(n < 0){
        return;
    }
    if(n == 0){
        System.out.println(psf);
        return;
    }
    printStairPaths(n-1,psf+"1");  — 1
    printStairPaths(n-2,psf+"2");  — 2
    printStairPaths(n-3,psf+"3");  — 3
}
```

nv
vn

$n_r = 2$

$n_c = 2$

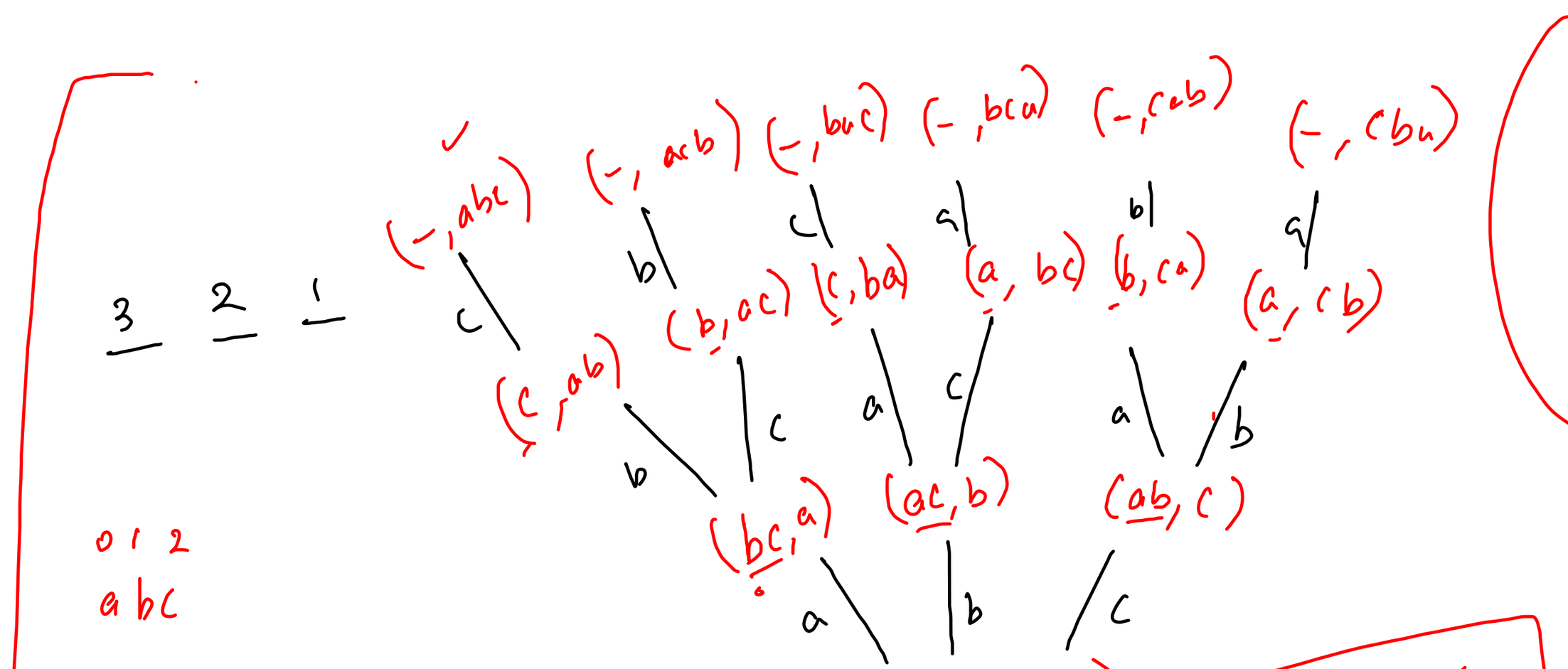






```
if( sr == dr && sc == dc)
  { print(psf);
    return;
  }
if( sr > dr || sc > dc)
  { return; }
```

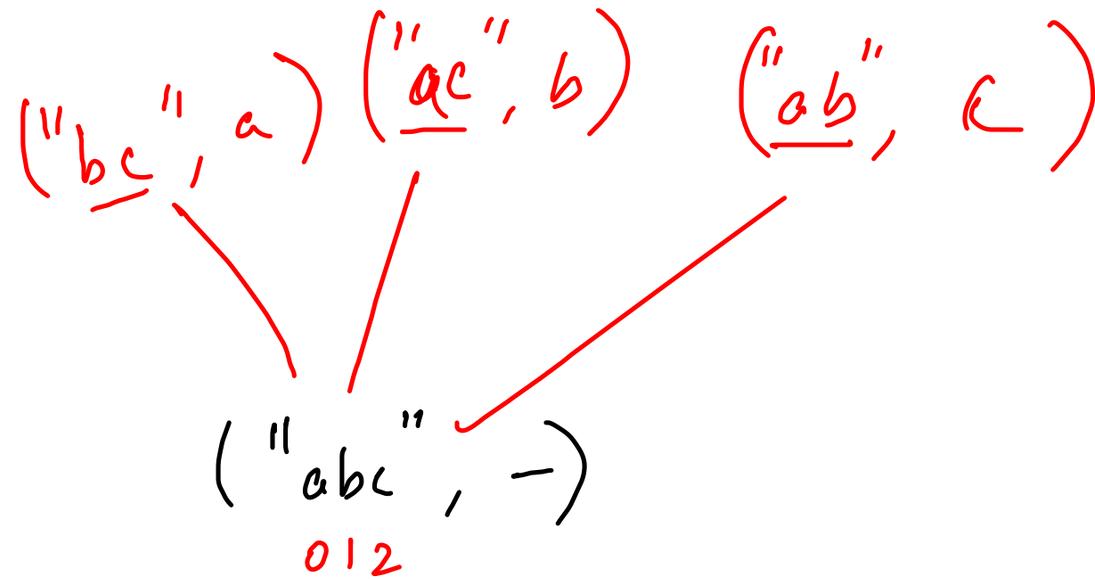



0 1 2
a b c

| idx | ch |
|---|---|
| 0 | a |
| 1 | b |
| 2 | c |

| l part | r part | row |
|---|---|---|
| ss(0,0) = "" | ss(1) = "b" | "bc" |
| ss(0,1) = "a" | ss(2) = "c" | "ac" |
| ss(0,2) = "ab" | ss(3) = "" | "ab" |

```java
public static void printPermutations(String qsf,String asf) {
    if(qsf.length() == 0){
        System.out.println(asf);
        return;
    }
    for(int i = 0 ; i < qsf.length() ; i++){
        char ch = qsf.charAt(i);

        String lPart = qsf.substring(0,i);
        String rPart = qsf.substring(i+1);

        printPermutations(lPart+rPart,asf+ch);
    }
}
```




String = abc

| i | ch | lpart | rpart |
|---|---|---|---|
| 0 | a | "" | "bc" |
| 1 | b | "a" | "c" |
| 2 | c | "ab" | "" |