

Min. No. of Software developers:

Skills \rightarrow a, b, c, d, e

0 dev. \rightarrow a, c

1 dev. \rightarrow b, d

2 dev. \rightarrow a, b, c, d

2 dev. \rightarrow e

Min no. of dev.
Such that all
skill set can be
grasp from them.

Ex \rightarrow we can grasp all skills

Date: 21st Jan 2022.

1. Minimum Number Of Software Developers
2. Number Of Valid Words
3. All Repeating Except One
4. All Repeating Except Two
5. One Repeating And One Missing
6. All Repeating Three Times Except One

① Map Skills with bit index.

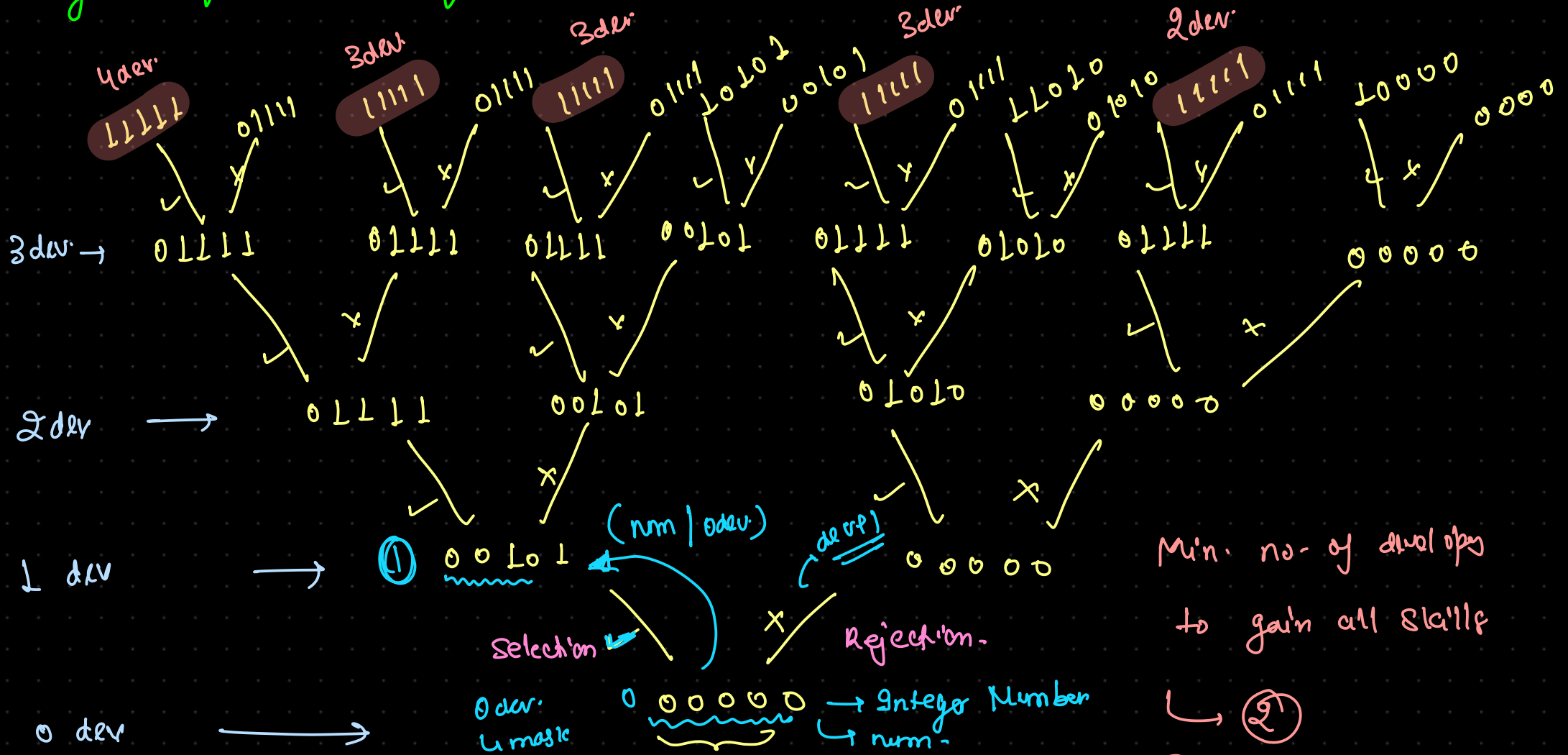
| | e | d | c | b | a | Skills |
|--|---|---|---|---|---|-----------|
| | 4 | 3 | 2 | 1 | 0 | bit index |

② mask for dev. with their skills

| | | | | | | |
|---------------------------------|---------------|---|---|---|---|---|
| 0 dev. \rightarrow a, c | \Rightarrow | 0 | 0 | 1 | 0 | 1 |
| 1 dev. \rightarrow b, d | \rightarrow | 0 | 1 | 0 | 1 | 0 |
| 2 dev. \rightarrow a, b, c, d | \rightarrow | 0 | 1 | 1 | 1 | 1 |
| 2 dev. \rightarrow e | \rightarrow | 1 | 0 | 0 | 0 | 0 |

these are bit mask

Using Subsequence technique-



Min. no- of envelopes
to gain all skills

②

[2dev, 3dev]

① Mask for dev

Integer Number

num-

skill set gained

Argument

dev - 2nd

dev - 1st

dev - 0th

dev - 3rd

dev - 4th

dev - 5th

dev - 6th

dev - 7th

dev - 8th

dev - 9th

dev - 10th

dev - 11th

dev - 12th

dev - 13th

dev - 14th

dev - 15th

dev - 16th

dev - 17th

dev - 18th

dev - 19th

dev - 20th

dev - 21st

dev - 22nd

dev - 23rd

dev - 24th

dev - 25th

dev - 26th

dev - 27th

dev - 28th

dev - 29th

dev - 30th

dev - 31st

dev - 32nd

dev - 33rd

dev - 34th

dev - 35th

dev - 36th

dev - 37th

dev - 38th

dev - 39th

dev - 40th

dev - 41st

dev - 42nd

dev - 43rd

dev - 44th

dev - 45th

dev - 46th

dev - 47th

dev - 48th

dev - 49th

dev - 50th

dev - 51st

dev - 52nd

dev - 53rd

dev - 54th

dev - 55th

dev - 56th

dev - 57th

dev - 58th

dev - 59th

dev - 60th

dev - 61st

dev - 62nd

dev - 63rd

dev - 64th

dev - 65th

dev - 66th

dev - 67th

dev - 68th

dev - 69th

dev - 70th

dev - 71st

dev - 72nd

dev - 73rd

dev - 74th

dev - 75th

dev - 76th

dev - 77th

dev - 78th

dev - 79th

dev - 80th

dev - 81st

dev - 82nd

dev - 83rd

dev - 84th

dev - 85th

dev - 86th

dev - 87th

dev - 88th

dev - 89th

dev - 90th

dev - 91st

dev - 92nd

dev - 93rd

dev - 94th

dev - 95th

dev - 96th

dev - 97th

dev - 98th

dev - 99th

dev - 100th

```

req_skills =
["algorithms", "math", "java", "reactjs", "csh  

arp", "aws"], people =
[["algorithms", "math", "java"],  

["algorithms", "math", "reactjs"],  

["java", "csharp", "aws"],  

["reactjs", "csharp"], ["csharp", "math"],  

["aws", "java"]]

```

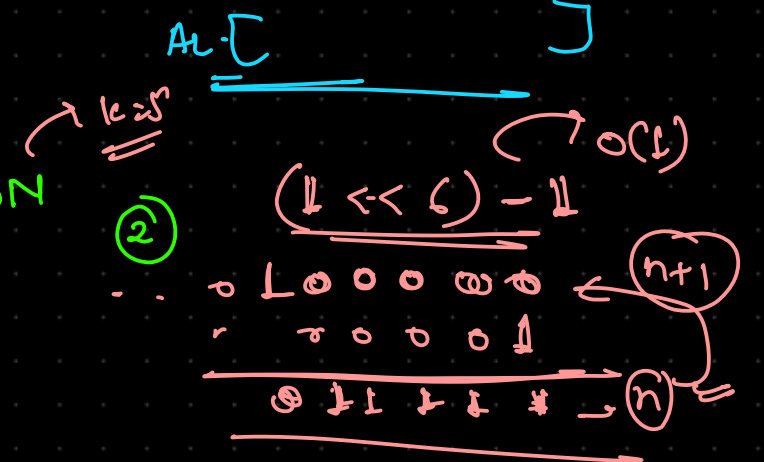
Skill to bit map → Map

| | dev | 5 | 4 | 3 | 2 | 1 | 0 | Integer No |
|----------------|-------|---|---|---|---|---|---|-----------------------|
| algorithms → 0 | → 0 → | 0 | 0 | 0 | 1 | 1 | 1 | 2 |
| math → 1 | → 1 → | 0 | 0 | 1 | 0 | 1 | 1 | 4-1 → 3 |
| java → 2 | → 2 → | 1 | 1 | 0 | 1 | 0 | 0 | (11) ₂ → 3 |
| reactjs → 3 | → 3 → | 0 | 1 | 1 | 0 | 0 | 0 | 100 → 4 |
| csharp → 4 | → 4 → | 0 | 1 | 0 | 0 | 1 | 0 | 111 → 7 |
| aws → 5 | → 5 → | 1 | 0 | 0 | 1 | 0 | 0 | 1000 → 8 |

0 0 1 1 1 1 1

Right to left
First k bits ON

① Iterate and ON bit.



No. of valid words:

7 → words

aaaa asas able ability actt actor access

6 → puzzle

aboveyz abrodyz absolute absoryz actresz gaswxyz

For a given puzzle word is valid if

① word contains first letter of puzzle

② each letter of word should present in puzzle.

word = aabc
puzzle = abcd

Valid

word & puzzle = ⇒ even

| Puzzle | words → | aaa | asas | able | ability | actt | actor | access. |
|--------------|---------|-----|------|------|---------|------|-------|---------|
| aboveyz → ① | aaa | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| abrodyz → ① | aaa | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| absolute → ③ | aaa | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| absoryz → ② | aaa | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| actresz → ④ | aaa | ✓ | ✓ | ✗ | ✗ | ✓ | ✗ | ✓ |
| gaswxyz → ⑥ | aaa | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

Suppose → a b c d e f g h i j

puzzle

↓

d g c j b

b g j i

a f i e

c g h f

HashMap < char, Something > → ArrayList<Integer>

a → hiba, jba f

b → begi, hiba, jba f

c → begi, cegh

d → di h j

e → cegh

f → jba f

g → begi, cegh

h → di h j, hiba, cegh

i → begi, di h j, hiba

j → di h j, jba f

How to store this

bits
z h g e d c b a
-- 1 1 0 1 0 1 0 0

words

L

bcgi

di h j

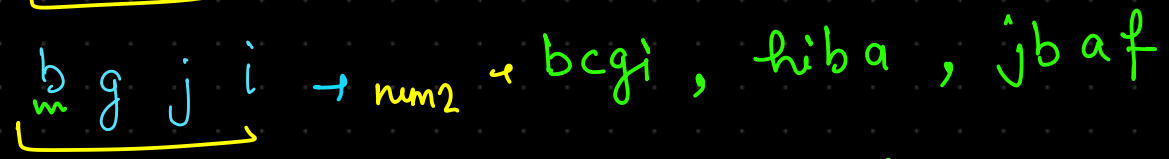
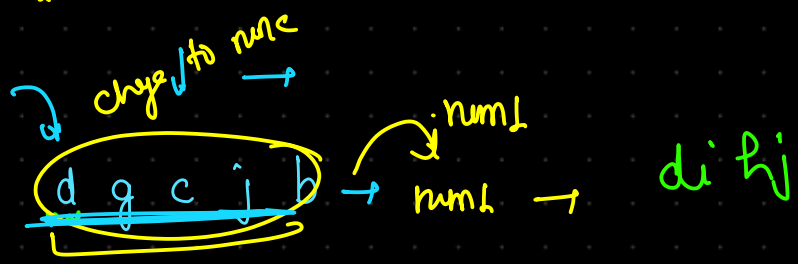
hiba

cegh

jba f

Map change to num

- a → hiba, jba f
- b → b c g i, hiba, jba f
- c → b c g i, c e g h
- d → d i h j
- e → c e g h
- f → jba f
- g → b c g i, c e g h
- h → d i h j, hiba, c e g h
- i → b c g i, d i h j, hiba
- j → d i h j, jba f



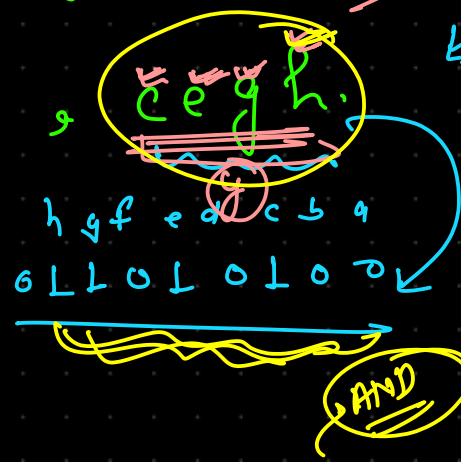
freeze

num1's → bit mask having bit are ON on character index

h g f e d c b a

0 1 1 0 1 0 1 0 0

c g h f →



How to check if x have some bit as y??

may be

0 1 1 0 1 0 1 0 0 x

have more ON bit

$((y \& x) == y)$ → same bit are ON

→ all character of y are present in x

① Map < Character, AL < Integer >>
 char bit mask of word

```

HashMap<Character, ArrayList<Integer>> map = new
for(int i = 0; i < 26; i++) {
    char ch = (char)(i + 'a');
    map.put(ch, new ArrayList<>());
}

for(String word : words) {
    for(int i = 0; i < word.length(); i++) {
        char ch = word.charAt(i);
        map.put(ch, word);
    }
}

```

a @ a a

a → aaaa, aaaa, aaaa

→ to avoid this problem

HashSet < Char >

a → [] ←

Output for this section

Map change to make
 a → hiba, jba f

b → b c g i, h i b a, j b a f

c → b c g i, c e g h

d → d i h j

e → c e g h

f → j b a f

g → b c g i, c e g h

h → d i h j, h i b a, c e g h

i → b c g i, d i h j, h i b a

j → d i h j, j b a f

word are
 present in

terms of mask

// map of char vs bit mask of word

```
HashMap<Character, ArrayList<Integer>> map = new HashMap<>();
```

```
for(int i = 0; i < 26; i++) {
    char ch = (char)(i + 'a');
    map.put(ch, new ArrayList<>());
}
```

```
for(String word : words) {
    HashSet<Character> set = new HashSet<>();
    // prepare mask for word
    int mask = 0;
    for(int i = 0; i < word.length(); i++) {
        char ch = word.charAt(i);
        int k = (int)(ch - 'a');
        mask = (mask | (1 << k));
    }
    // Add in map corresponding to letter
    for(int i = 0; i < word.length(); i++) {
        char ch = word.charAt(i);
        if(set.contains(ch)) continue;
        set.add(ch);
        map.get(ch).add(mask);
    }
}
```

a → [1] [3] [69]
 b → [3] [26] [98]
 c → [9]
 d → [26]
 e → [26]
 f → [98]
 g → [98] [19]

a b c d e f g
 Assumption: Suppose Alphabets are

puzzle
 ↓ word → a a a a a b d e b f g b g a c
 → b c f g m₁ → 3, 26, 98
 c b e a m₂
 d e a a m₃
 f c a b m₄
 f g e d m₅
 g a c b m₆

8 & m₁ = 3
 Count ↑

Return count.

a a a → (0 0 0 0 0 1)₂ = (1)₁₀

a a b → (0 0 0 1 1)₂ = (3)

f g b = (1 1 0 0 0 1 0)₂ = (98)

```
// prepare count
ArrayList<Integer> counts = new ArrayList<>();
for(String puzzle : puzzles) {
    // prepare mask of puzzle
    int mask = 0;
    for(int i = 0; i < puzzle.length(); i++) {
        char ch = puzzle.charAt(i);
        int k = (int)(ch - 'a');
        mask = (mask | (1 << k));
    }
    // check in valid word and find count
    char firstChar = puzzle.charAt(0);
    int count = 0;
    for(int wordMask : map.get(firstChar)) {
        if((wordMask & mask) == wordMask) {
            count++;
        }
    }
    counts.add(count);
}
return counts;
```

deb → 1 1 0 1 0 1
 (0 1 1 0 1 0)₂ = (26)₁₀
 1 1 1 1 + 8 + 2
 mas = 26
 22 64 + 2 = 66