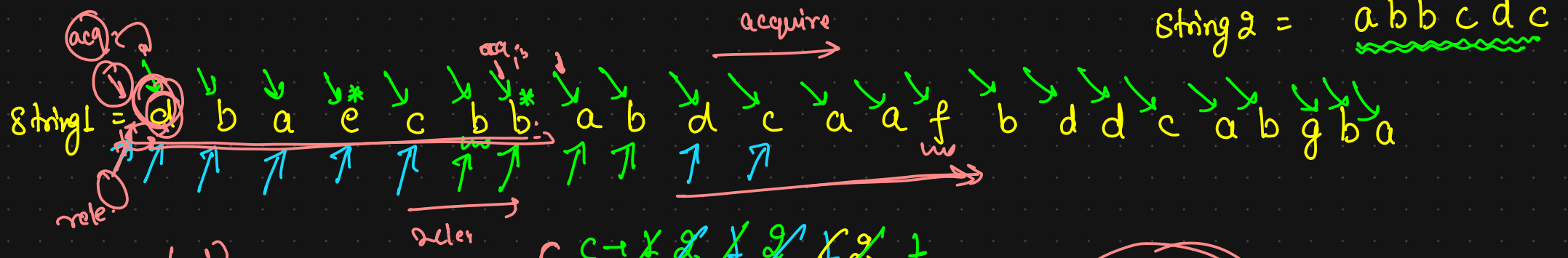
Date: 4th January 2022

• Agenda •

Acquire and Release group

- Smallest Substring Of A String Containing All Characters Of Another String
- Smallest Substring Of A String Containing All Unique Characters Of Itself
- Longest Substring With Non Repeating Characters
- Count Of Substrings Having All Unique Characters
- Longest Substring With Exactly K Unique Characters
- Count Of Substrings With Exactly K Unique Characters

1. Smallest Substring Of A String Containing All Characters Of Another String



s. Substring (a, b)
include
excluded

→ "acquire" till valid

Start "release" when invalid g → 1

map

c → X 2 X 2 X 2 1
f → 1
d → X 2 X X 2 2
b → X 2 2 X 2 X 2 2 X 3
a → X 2 X 2 X 2 2 4

freq. map string 2.

a → 1
b → 2
c → 2
d → 1

Requirement = 6

count = 0 X 2 2 X 3 4 5 5 5 5 6

ans = c b b a b d c

acquire.

Release.

ans.

cnt --

while(true)

flag1 = false
flag2 = false
if(count < requirement)

loop

acquire
1. add char. in map.
2. conditional increment in count [if map.get(ch) <= fmap.get(ch)]

flag1 = true

loop

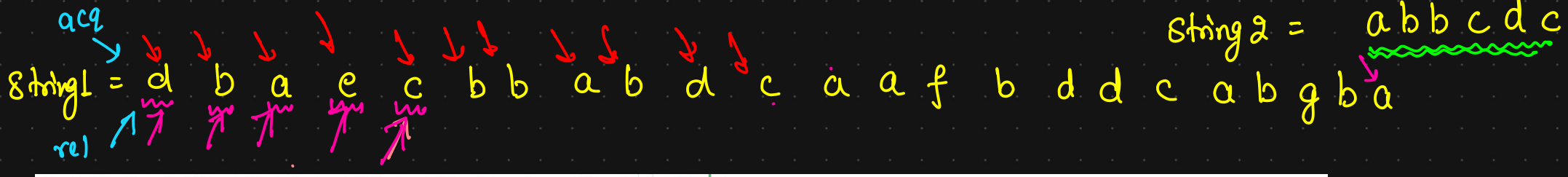
if(count == requirement)
1. ✓ Hold answer, if substring is smaller

loop

release
2. Hold releasing char. and decrement its freq,
if freq. == 1 → remove key from map as well
3. with invalid release decrease count
↳ condition of invalid.
if(map.get(ch) < fmap.get(ch)) count--;

flag2 = true

if(flag1 == false && flag2 == false)
{ break;




```java
// smallest substring of string containing all characters of anothe
public static String smallestSubStringAllChar(String s1, String s2)
    // frequency map for string 2
    HashMap<Character, Integer> fmap = new HashMap<>();
    for(int i = 0; i < s2.length(); i++) {
        char ch = s2.charAt(i);
        fmap.put(ch, fmap.getOrDefault(ch, 0) + 1);
    }

    // move toward solution
    int acq = -1; // acquire
    int rel = -1; // release
    int count = 0;
    int requirement = s2.length();
    HashMap<Character, Integer> map = new HashMap<>();
    String ans = "";
    while(true) {
        boolean acFlag = false; // acquire flag
        boolean reFlag = false; // release flag
        // acquire
        while(acq < s1.length() - 1 && count < requirement) {
            acq++;
            char ch = s1.charAt(acq);
            map.put(ch, map.getOrDefault(ch, 0) + 1);

            // conditional increment in count
            if(map.get(ch) <= fmap.getOrDefault(ch, 0)) {
                count++;
            }
            acFlag = true;
        }

        // release
```

```java
    }

    // release
    while(count == requirement) {
        // hold ans, and if smallest then update the result
        String tempAns = s1.substring(rel + 1, acq + 1);
        if(ans.length() == 0 || tempAns.length() < ans.length()) {
            ans = tempAns;
        }
        // get the character and remove from map(either dec. freq.
        rel++;
        char ch = s1.charAt(rel);
        map.put(ch, map.get(ch) - 1);
        if(map.get(ch) == 0) map.remove(ch);

        // decrement in count if release is invalid
        if(map.getOrDefault(ch, 0) < fmap.getOrDefault(ch, 0)) {
            count--;
        }
        reFlag = true;
    }

    // conditional break from loop
    if(acFlag == false && reFlag == false) {
        break;
    }
}
urn ans;

Run | Debug
static void main(String[] args) {
```

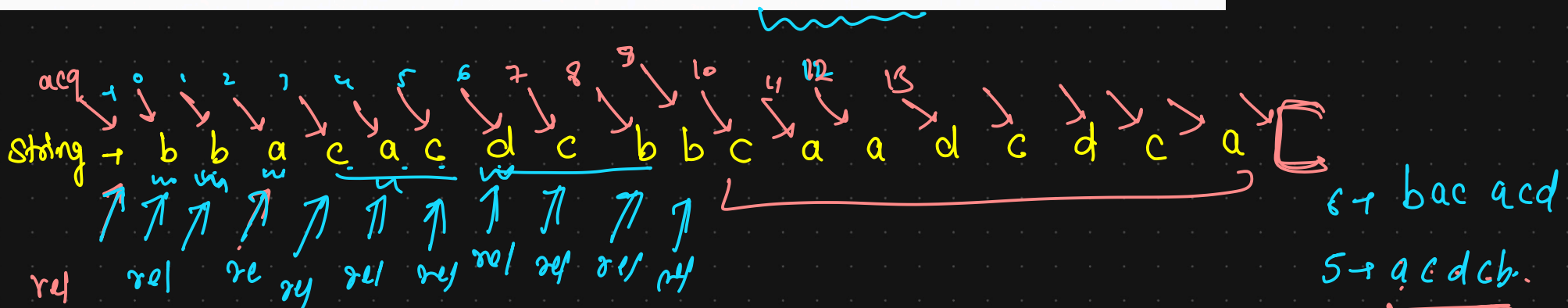
c→1 2 1
d→1 2 1
b→1 2 2 1 2
a→1 2 1
c→1 0
count=0
2

req=6

a→1
b→2
c→2
d→1

ans = c b b a b d c

2. Smallest Substring Of A String Containing All Unique Characters Of Itself



templen = acq - rel = ~~7~~ ~~6~~ 5

ansLen = ~~∞~~ ~~7~~ ~~6~~ 5

a → ~~1~~ ~~2~~ 3

c → ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ ~~2~~ 3

d → ~~1~~ 2

requirement = set.size();

count = map.size();

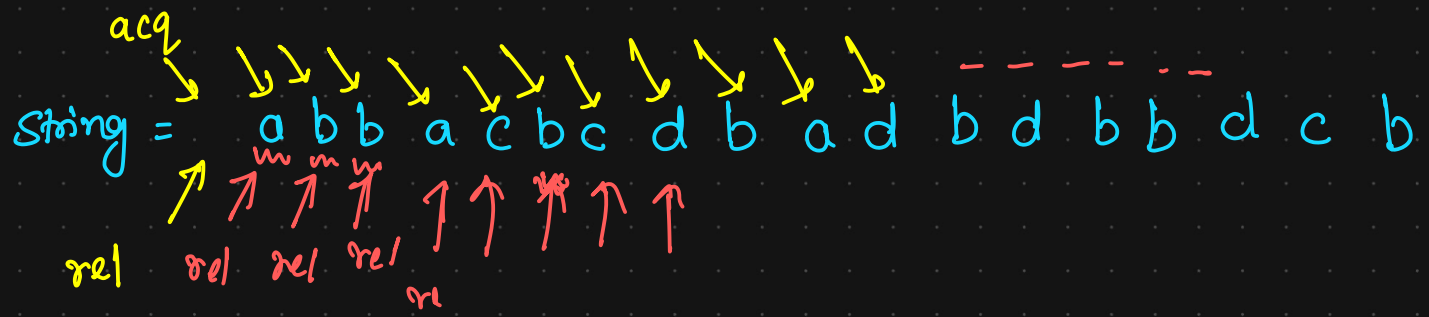
uniqueElemt = map.size

(Because keys are unique in map)

HashSet

- b
- a
- c
- d
- e

2. Longest Substring With Non Repeating Characters



templen = 4

anslen = ~~1~~ ~~2~~ ~~3~~ 4

fmap →

a → ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1

b → ~~1~~ ~~2~~ ~~1~~ ~~2~~ ~~1~~ ~~2~~ 1

c → ~~1~~ ~~2~~ ~~1~~ 0

d → ~~1~~ ~~2~~ 1

Schedule from this week [4th Jan 2022]

DSA

Tuesday → 9:00 - 12:00

Thursday → 9:00 - 12:00

Saturday → 11:00 - 3:00

Sunday → 11:00 - 3:00

LLD System Design

HLD → Halt

If system design start then

4:00 - 8:00

LLD → Sumeet Sir

- 8:00 - 12:00
- 8:00 - 12:00