

```

while(ep<s.length()){
    //introduce
    char sep = s.charAt(sp);
    hm.put(sep, hm.getOrDefault(sep, 0) + 1); //increment the freq of this person in hashmap
    if(hm.get(sep) == 1) noUnique++;
    if(hm.get(sep) == k) noRepk++;

    //shrink if im having extra chars
    while(noUnique > k){
        char ssp = s.charAt(sp);
        hm.put(ssp, hm.get(ssp) - 1);

        if(hm.get(ssp) == 0) noUnique--;
        if(hm.get(ssp) == k-1) noRepk--;

        sp++;
    }

    //update answer
    if(noUnique == k && noRepk == noUnique) ans = Math.max(ans, ep-sp+1);

    //expand
    ep++;
}
return ans;

```

sp  
X A B C A B C

ep

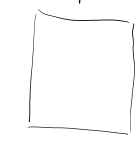
0 → 1 no. Unique = 1  
→ k no. Rep k = 0  
→ k-1

ans = 0

- Zero
- shrink no > 0
- update
- expand

sp=0  
ep=0  
noUnique=0  
noRepk=0  
ans=0

unique=3 F A B B C D X A C X A C C A D  
k=2  
noUnique=0  
noRepk=0  
ans=0



Zero

while(ep < n) {  
 hm[sep]++;  
 if(hm.get(sep) == 1) noUnique++;  
 if(hm.get(sep) == k) noRepk++;

shrink

while(noUnique > limit) {  
 hm[sp]--;  
 if(hm.get(sp) == 0) noUnique--;  
 if(hm.get(sp) == k-1) noRepk--;  
 sp++;

ans update

if (ans == limit + noRepk == unique) ans = max(ans, ep-sp+1);  
 ep++

X X X Y Y

```

// Solution: Sliding Window
// Problem: Longest Substring Without Repeating Characters
// Input: s = "abcabcbb"
// Output: 3
// Explanation: The characters 'a', 'b', 'c' form the longest substring without repeating characters.

```

S = R C X B X Y A P R R D B A F R C X B X Y A P R R D B A F  
T = P A B B R  
shrink [P, B] [B, R] [R, D] [D, B] [B, A] [A, F]  
if R C X B X Y A P R R D B A F > P A B B R  
ans = A P R R D B  
maxLen = 6

13-1-35367

do  
add(i < k)  
consider ans for minCo  
sp = 0  
ep = k  
while(ep < n)  
//shrink  
//null ptr  
//ans  
//sp = ep+1

k=3  
ep  
1 3 -1 -3 2 1 6 7  
0 1 2 3 4 5 6 7  
3 3 2 2 6 7  
3

sp  
1 3 -1 -3 2 1 6 7  
0 1 2 3 4 5 6 7  
ep

4  
maxLen  
3 3