4. Two Pointer Approach

In Two Approach:

Determine Three Conditions Always
Con1: KIS CONDITION PAR RUKNA HAI
Con2: START KO KAISE MOVE KARNA HAI *(start++/start--)*
Con3: END KO KAISE MOVE KARNA HAI *(end++/end--)*

Problem 1: Two Sum (Leetcode-1)

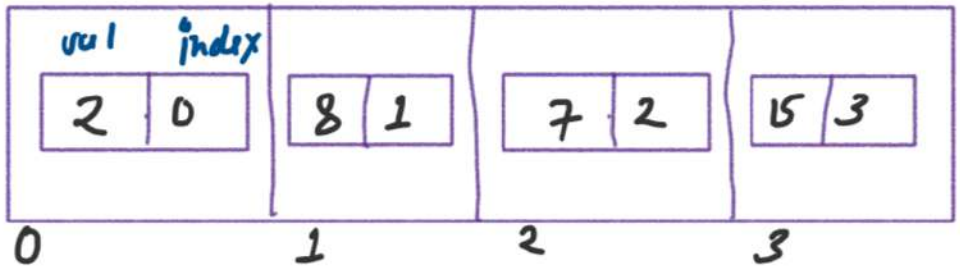
1. Two Sum (Leetcode-1)

Example 1:
Input: nums = [2,8,7,15], target = 9
Output: [0,2]

Explanation: Because nums[0] + nums[2] == 9, we return [0, 2].

STEP1

Create the array of **pair<value, index>** due to not shorted input array

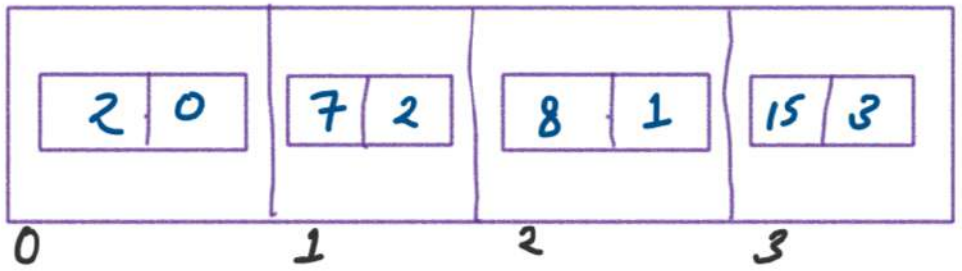


vector<pair<int,int>> temp

STEP2

Short the temp array based on value

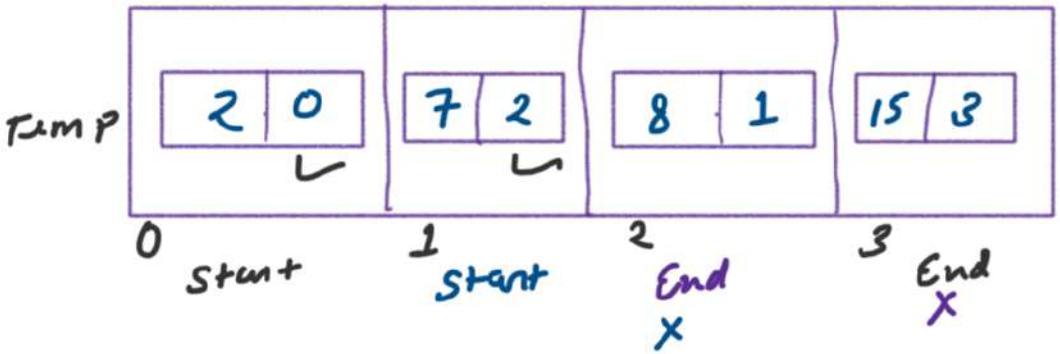
sort(temp.begin(), temp.End());



STEP 3

Apply two pointer approach to get the intent output

target = 9



1.

start = 0
End = 3
sum = 2 + 15
= 17

sum > target
↳ 17 > 9
⇒ End--

2.

start = 0
End = 2
sum = 2 + 8
= 10

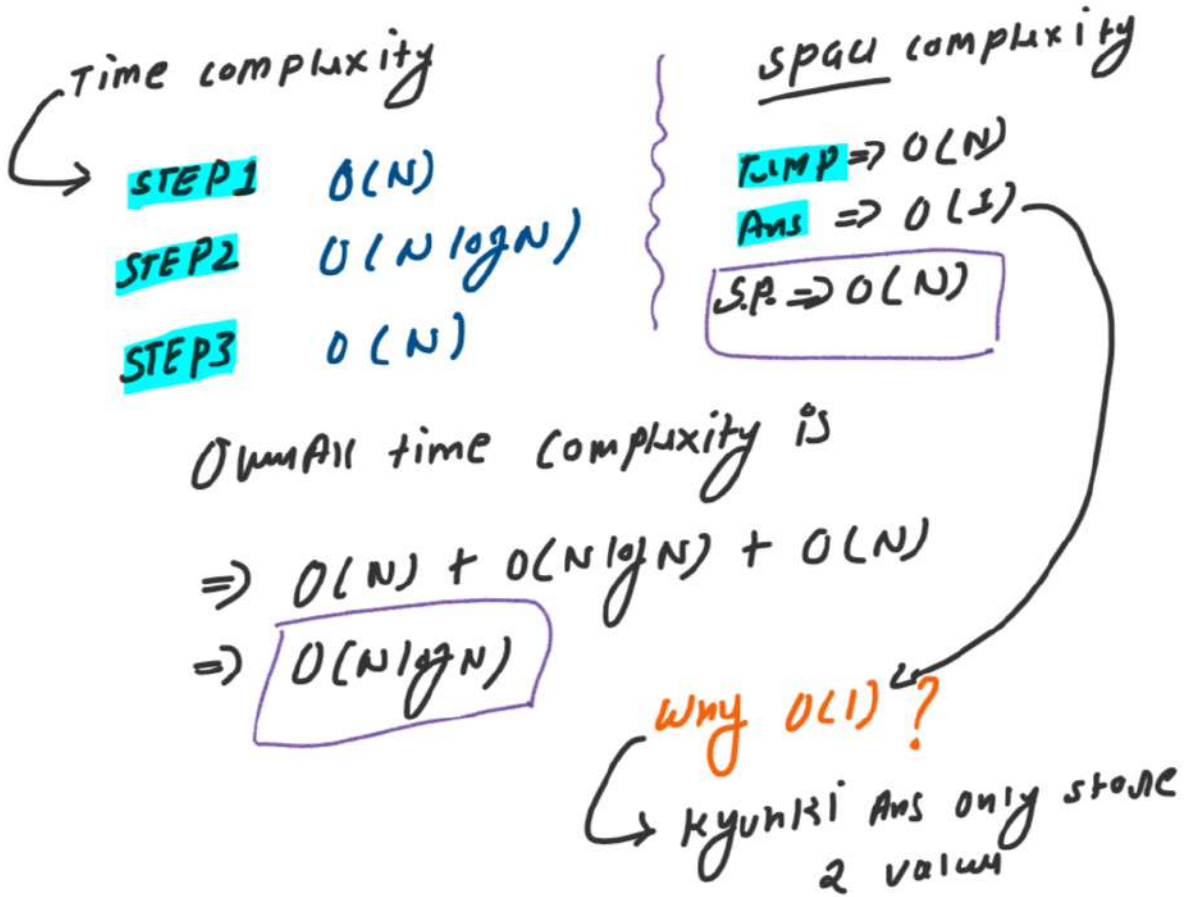
sum > target
↳ 10 > 9
⇒ End--

3.

start = 0
End = 1
sum = 2 + 7
= 9

sum == target
↳ return Index pair
[0, 2] ← output
STOP

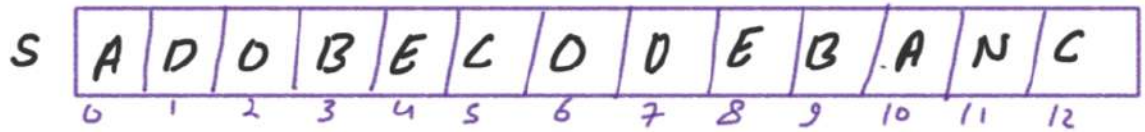
```cpp
// Problem 01: Two Sum (Leetcode-1)

class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        // Step 1: Create the array of pair<value, index> due to not
shorted input array
        vector<pair<int,int>> temp;
        for(int i=0; i<nums.size(); i++){
            temp.push_back({nums[i], i});
        }

        // Step 2: Short the temp array based on value
        sort(temp.begin(), temp.end());

        // Step 3: Apply two pointer approach to get the intent output
        int start = 0;
        int end = nums.size()-1;
        vector<int> ans;

        while(start < end){
            int sum = temp[start].first + temp[end].first;
            if(sum == target){
                ans.push_back(temp[start].second);
                ans.push_back(temp[end].second);
                break;
            }
            else if(sum > target){
                end--;
            }
            else{
                start++;
            }
        }
        return ans;
    }
};
```

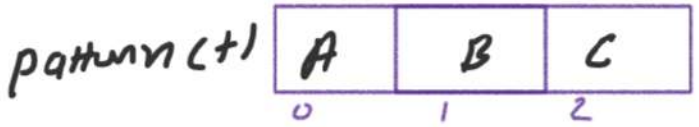


5. Optimization Approach

Problem 01: Minimum Window Substring (Leetcode-76)

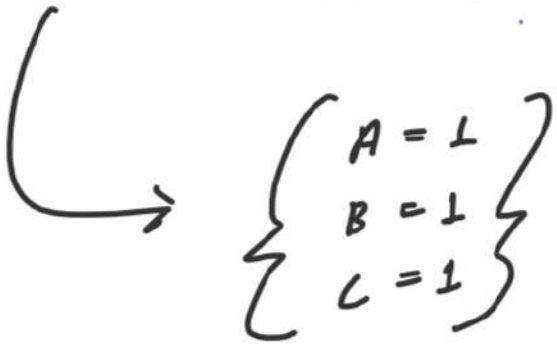
Example 1:
Input: s = "ADOBECODEBANC", t = "ABC"
Output: "BANC"





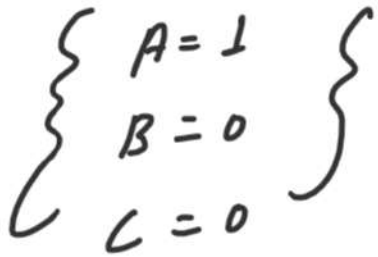
Valid Ans
↳ minimize : start ++

Invalid Ans
↳ Expand : end ++



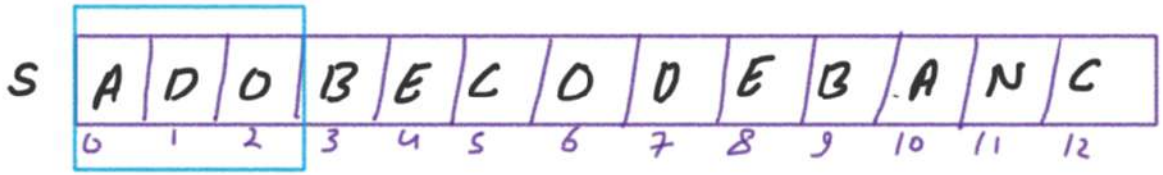
DRY RUN

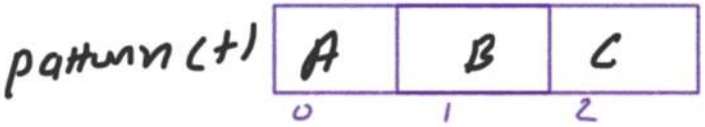
window 1 start = 0 end = 2

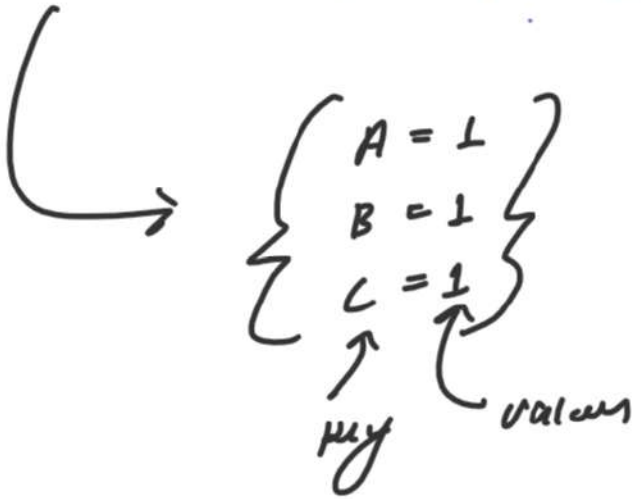


Invalid Ans

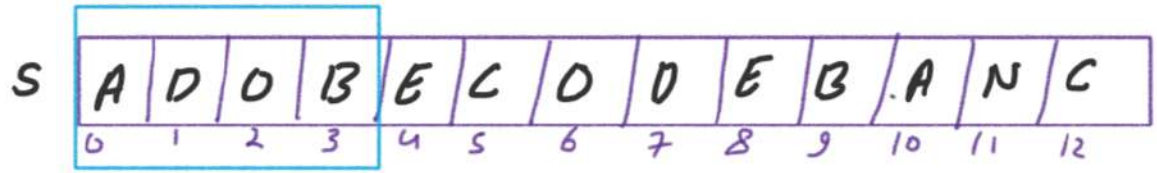
End++

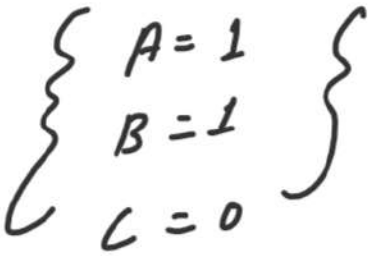






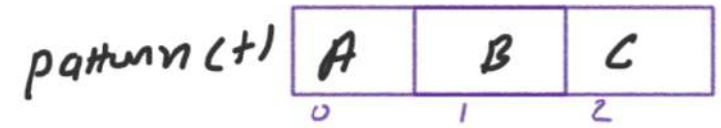
window = 2 start = 0 end = 3

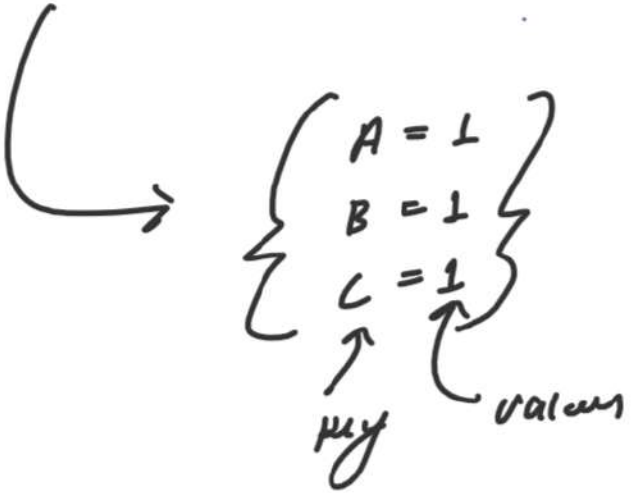




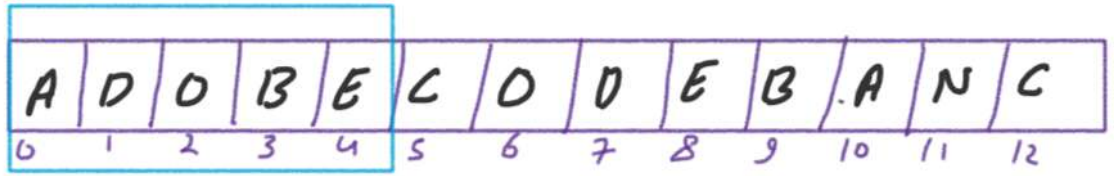
Invalid Ans

end++





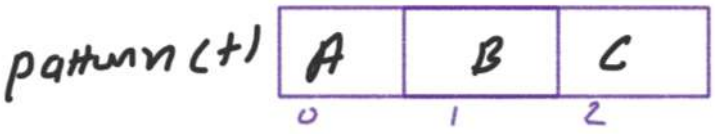
window = 3 start = 0 end = 4

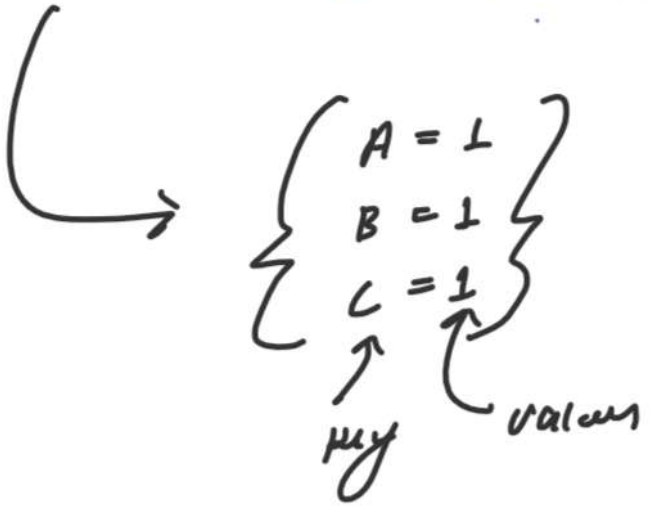


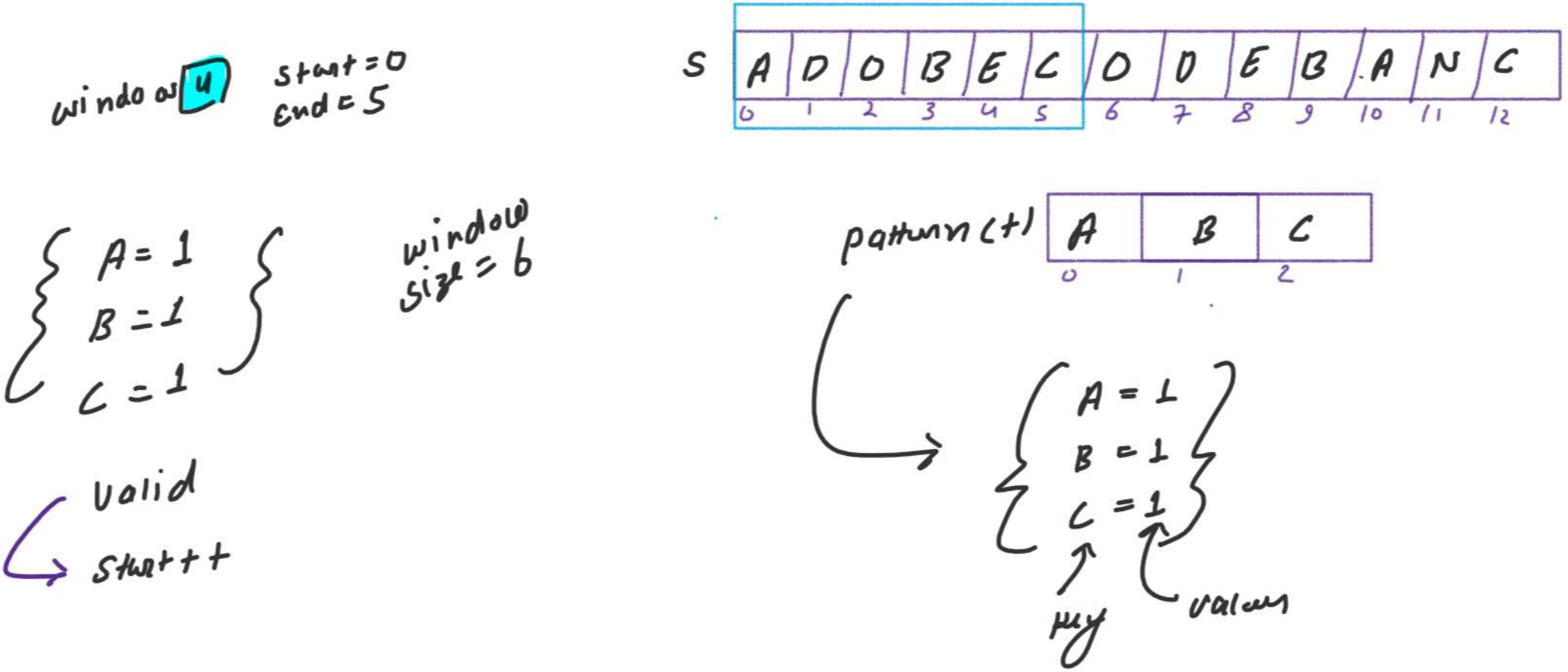
{ A = 1
B = 1
C = 0 }

Invalid Ans

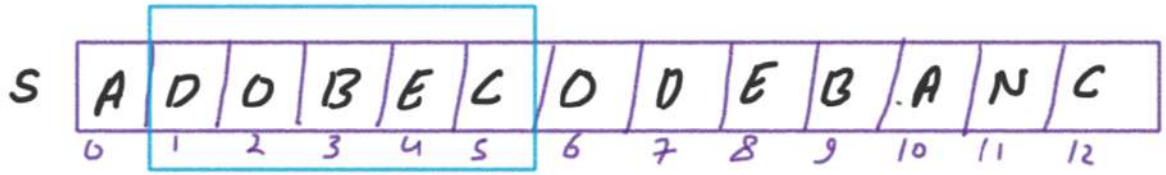
↳ end++

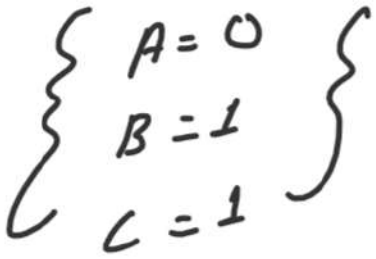





window 4
start = 0
end = 5
S
A D O B E C O D E B A N C
0 1 2 3 4 5 6 7 8 9 10 11 12
A = 1
B = 1
C = 1
window size = 6
valid
start++
pattern
A B C
0 1 2
A = 1
B = 1
C = 1
key
value

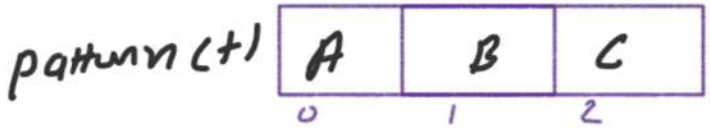
window 5 start = 1 end = 5

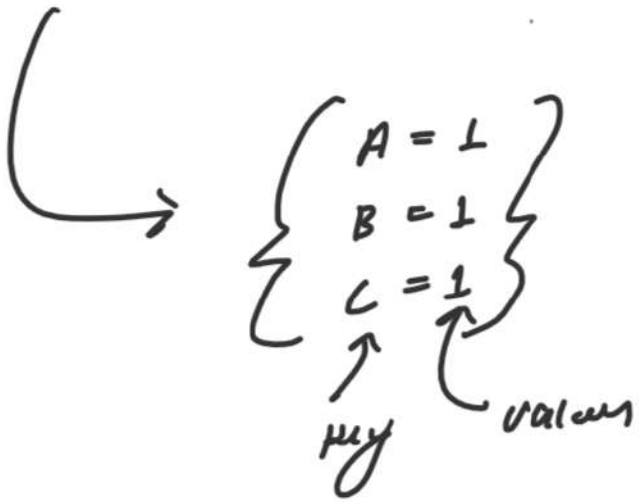




Invalid

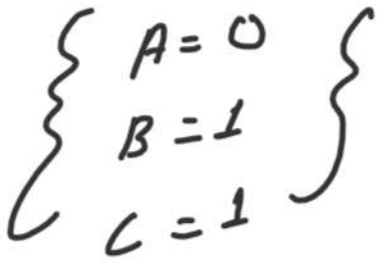
End++

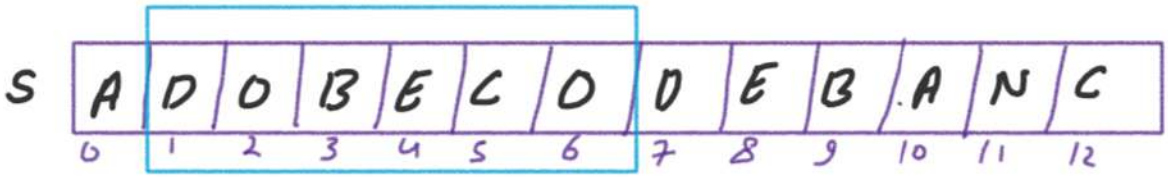


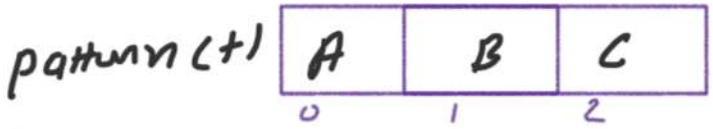


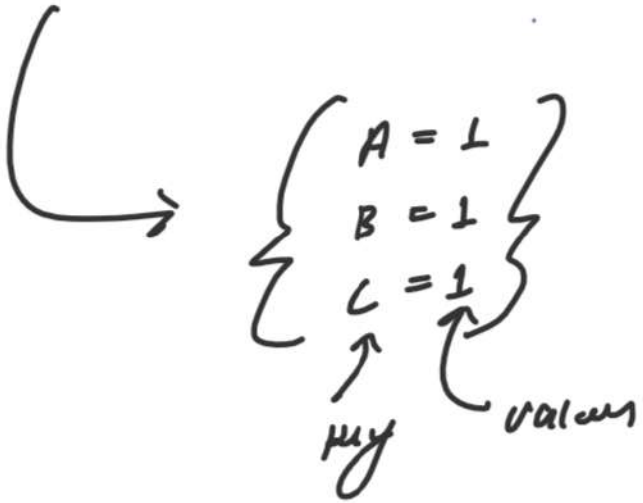
window = 6

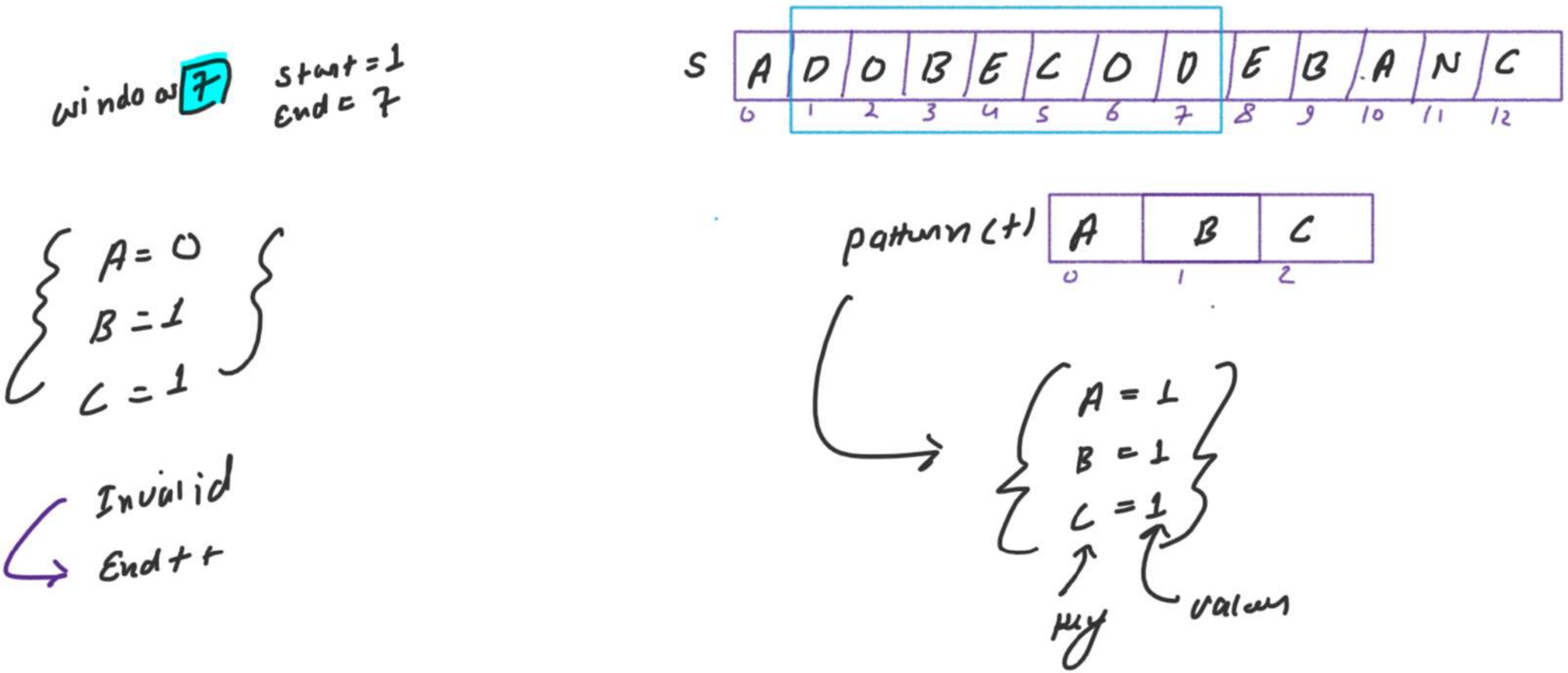
start = 1
end = 6



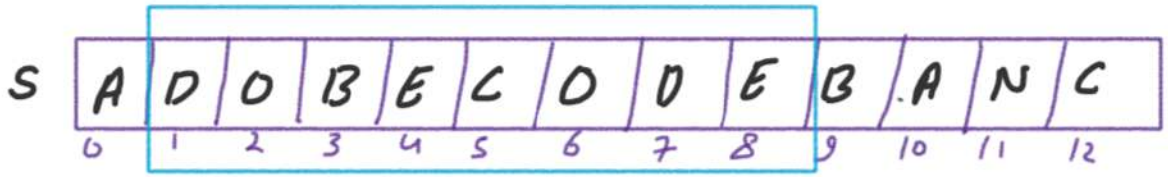


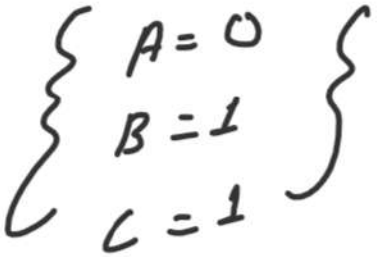


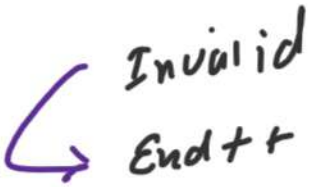


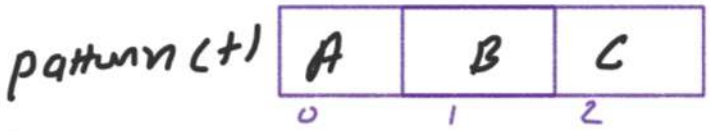

window 7
start = 1
end = 7
S
A D O B E C O D E B A N C
0 1 2 3 4 5 6 7 8 9 10 11 12
A = 0
B = 1
C = 1
Invalid
End++
pattern (+1)
A B C
0 1 2
A = 1
B = 1
C = 1
key
value

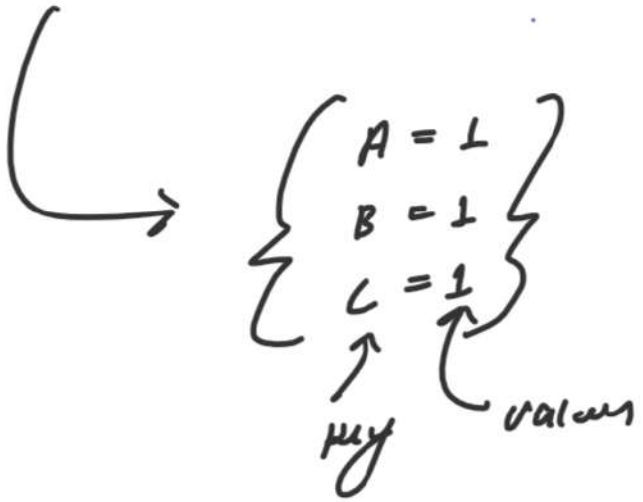
window = B start = 1, end = 8



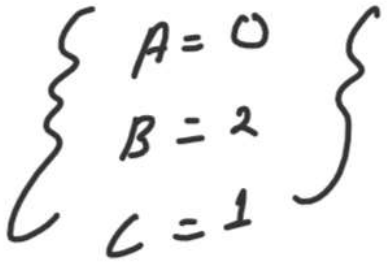


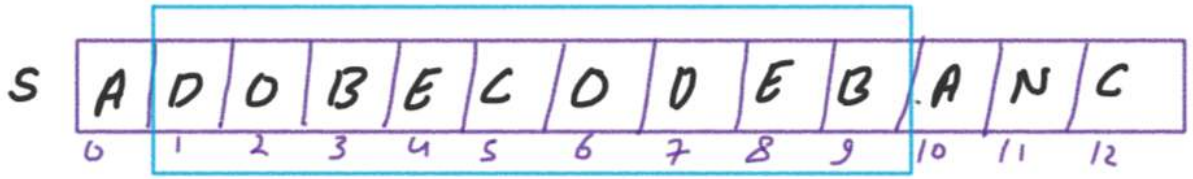


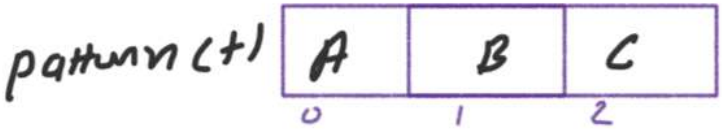


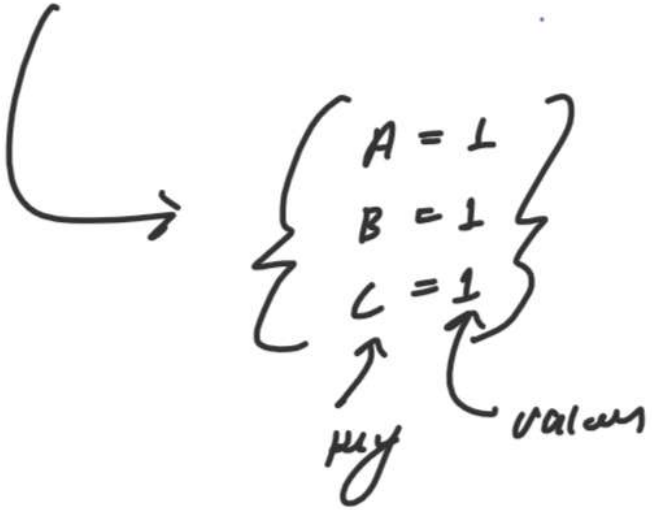


window = 9 start = 1 end = 9

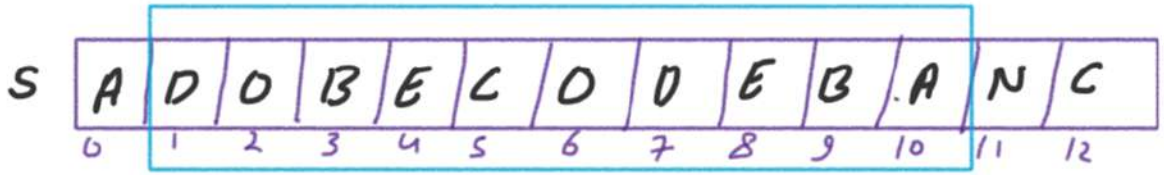


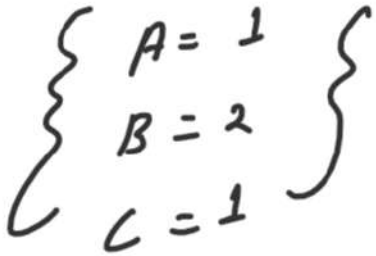




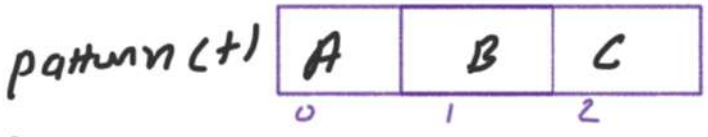


window 10 start = 1 end = 10



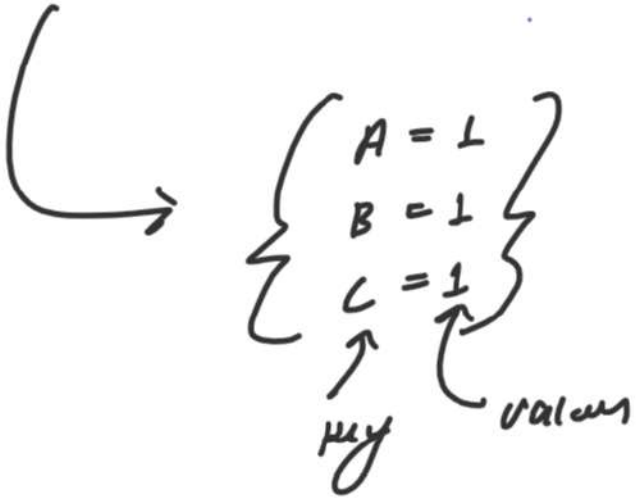


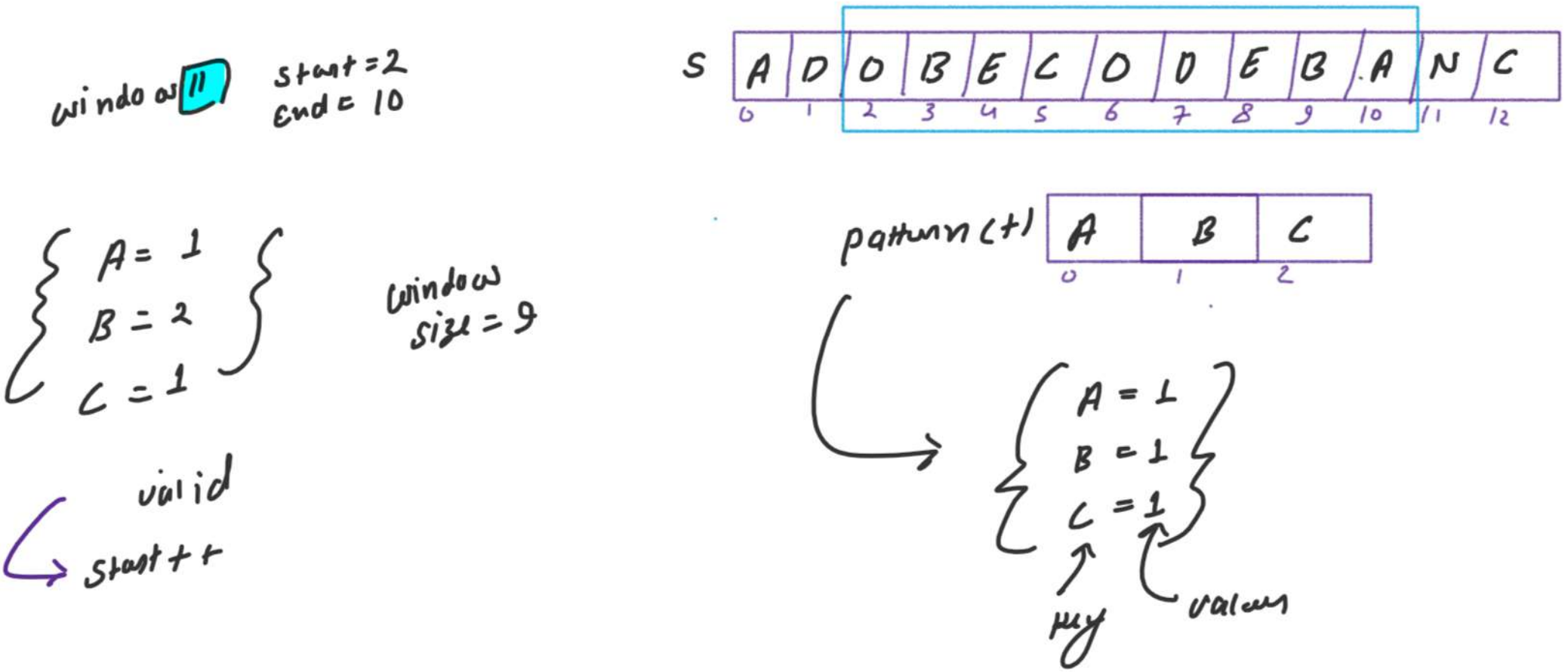
window size = 10



valid

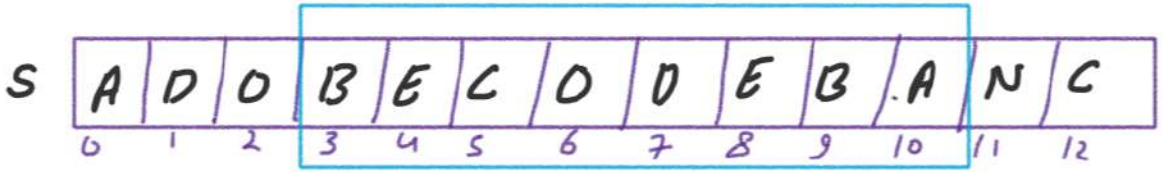
start ++

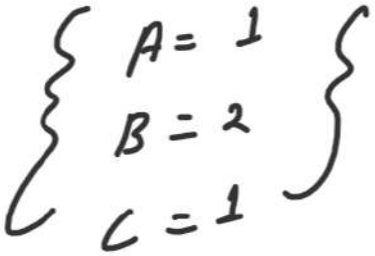



window 11
start = 2
end = 10
S
A D O B E C O D E B A N C
0 1 2 3 4 5 6 7 8 9 10 11 12
pattern (t)
A B C
0 1 2
A = 1
B = 2
C = 1
window size = 9
valid
start++
A = 1
B = 1
C = 1
key
value

window 12

start = 3
end = 10

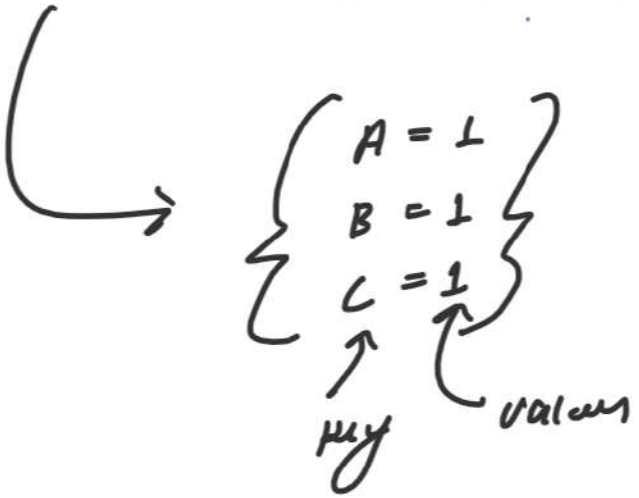




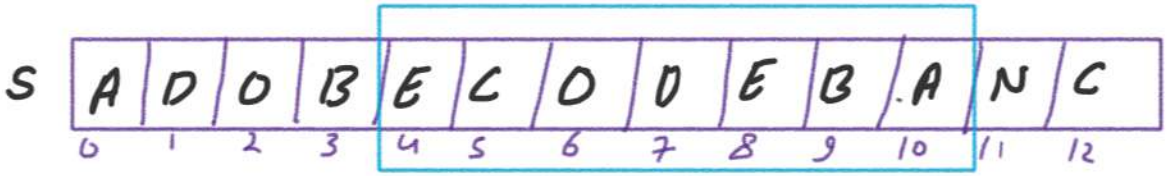
window size = 6

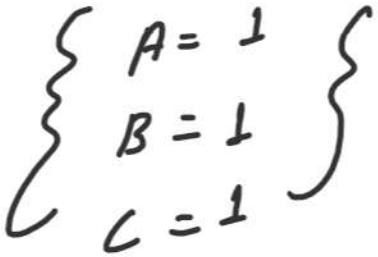
valid

start++

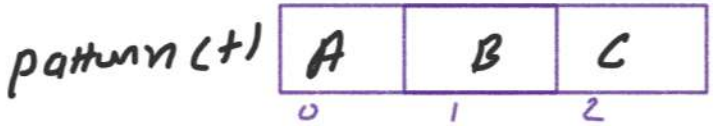


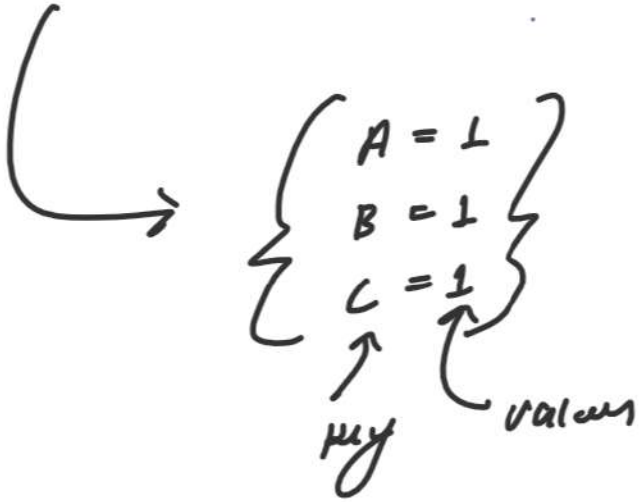
window 13 start = 4 end = 10

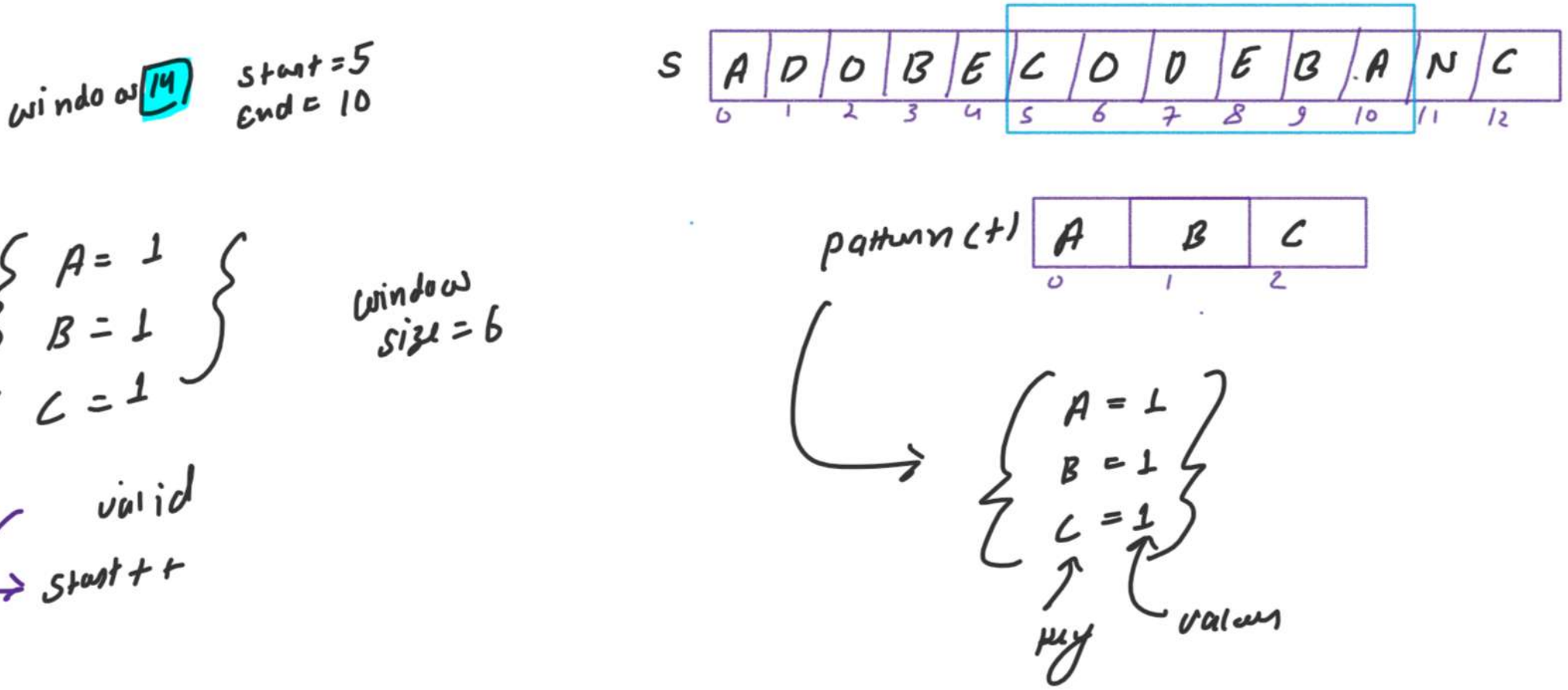




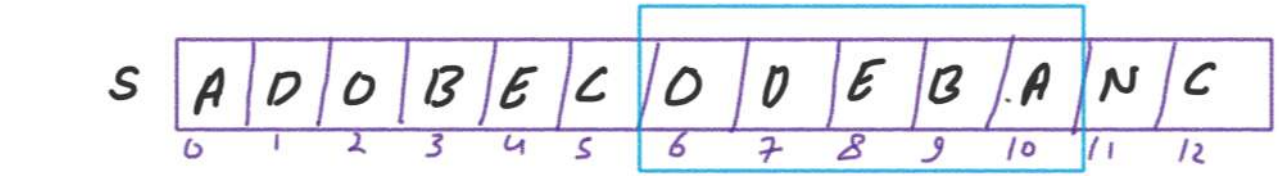
window size = 7

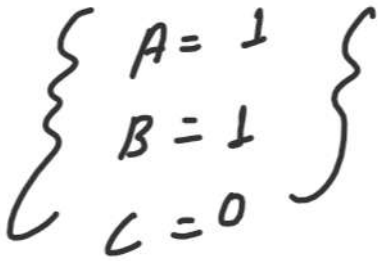





window 14
start = 5
end = 10
S
A D O B E C O D E B A N C
0 1 2 3 4 5 6 7 8 9 10 11 12
A = 1
B = 1
C = 1
window size = 6
valid
start++
pattern(t)
A B C
0 1 2
A = 1
B = 1
C = 1
key
values

window at 15 start = 6 end = 10

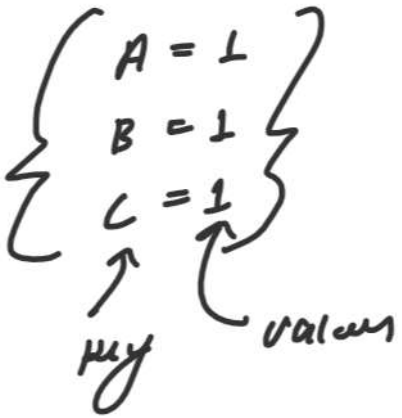




Invalid

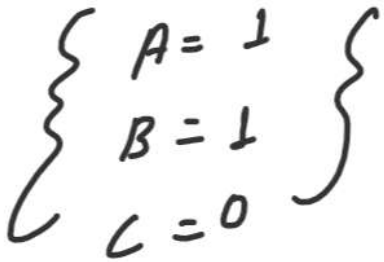
End++

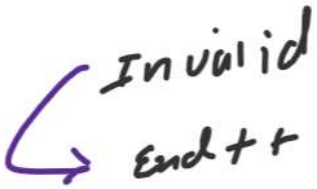
pattern(t)

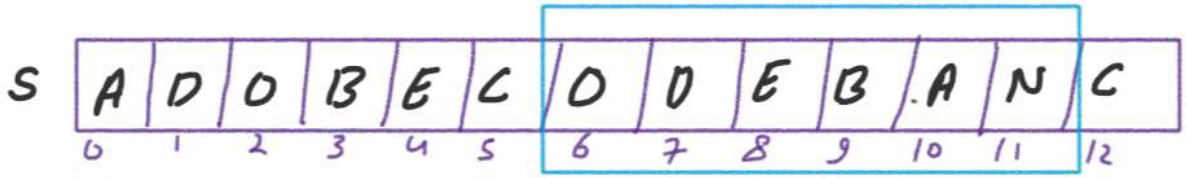


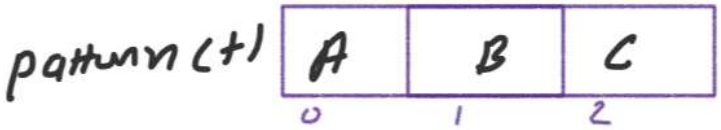
window = 6

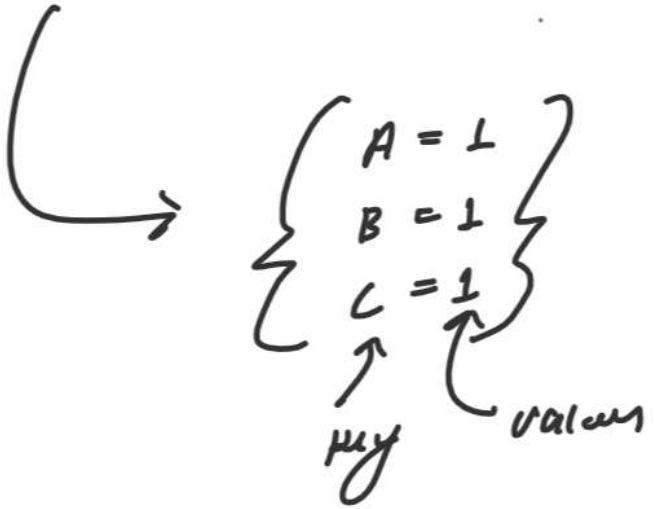
start = 6
end = 11

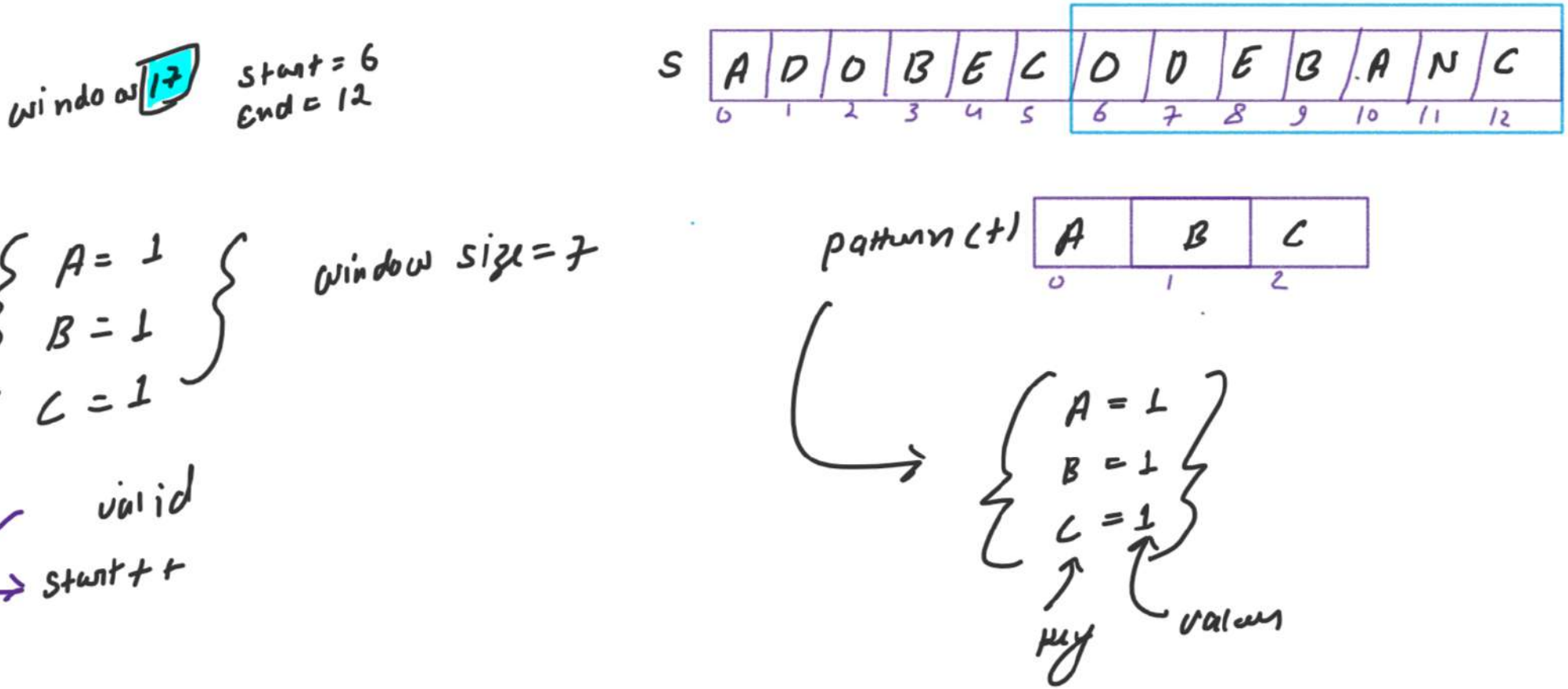






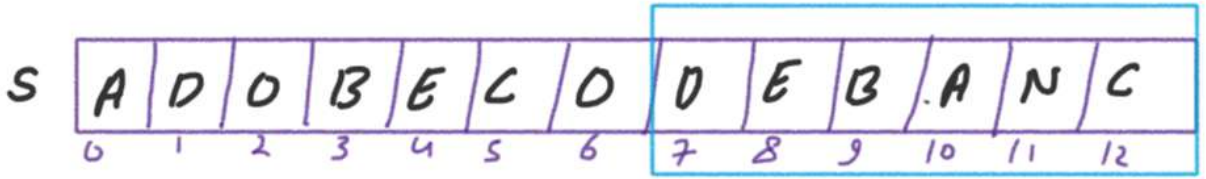


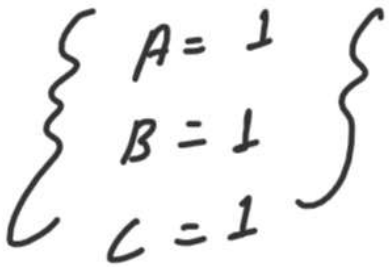



window
start = 6
end = 12
S
A D O B E C O D E B A N C
0 1 2 3 4 5 6 7 8 9 10 11 12
A = 1
B = 1
C = 1
window size = 7
valid
start++
pattern (t)
A B C
0 1 2
A = 1
B = 1
C = 1
key
values

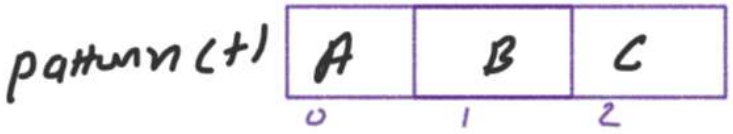
window 18

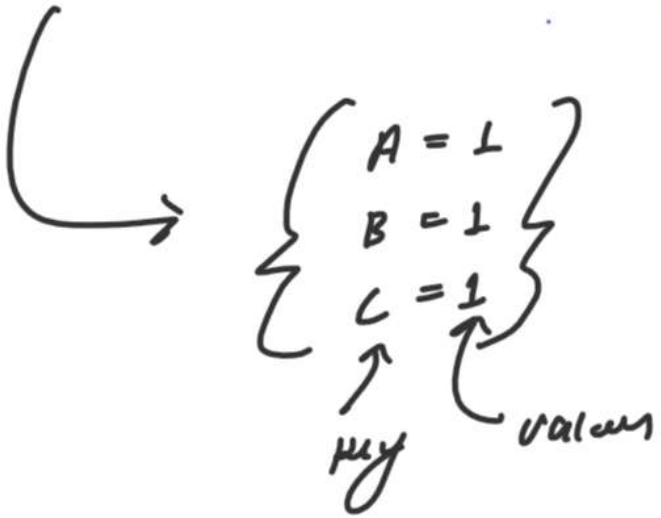
start = 7
end = 12



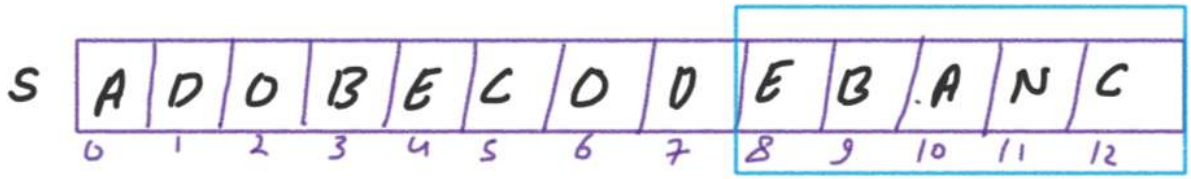


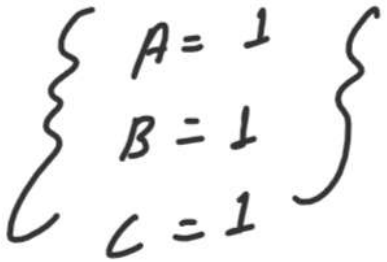
window size = 6



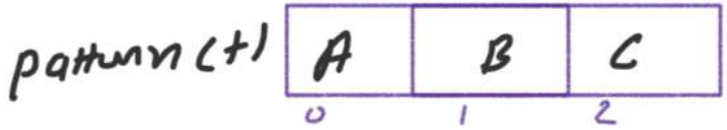


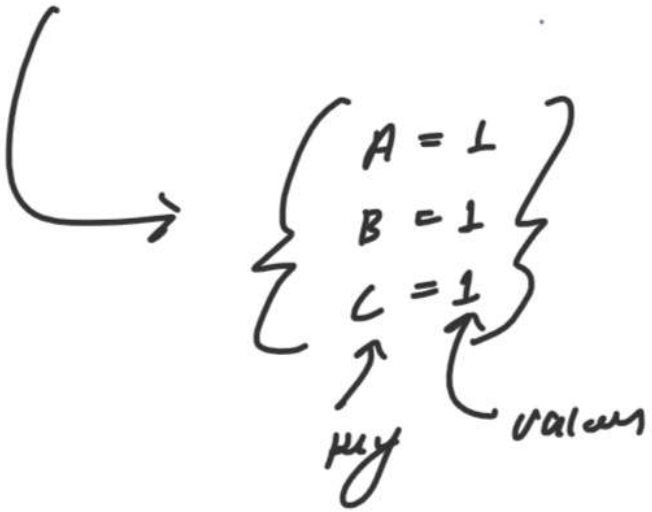
window [19] start = 8
end = 12



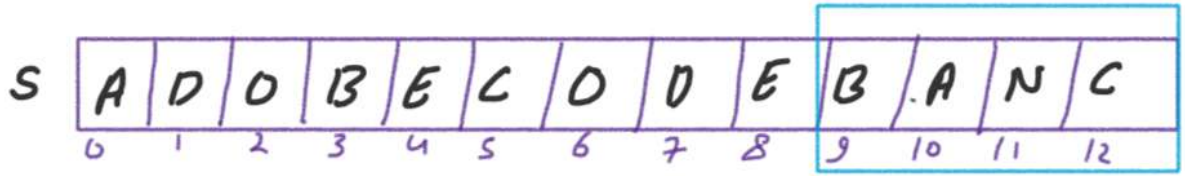


window size = 5





window 20 start = 9
end = 12



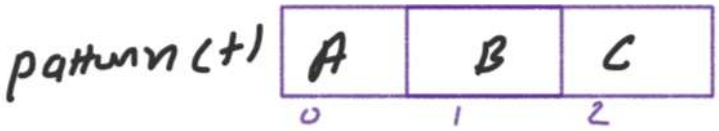
A = 1
B = 1
C = 1

window size = 4

BANC

valid

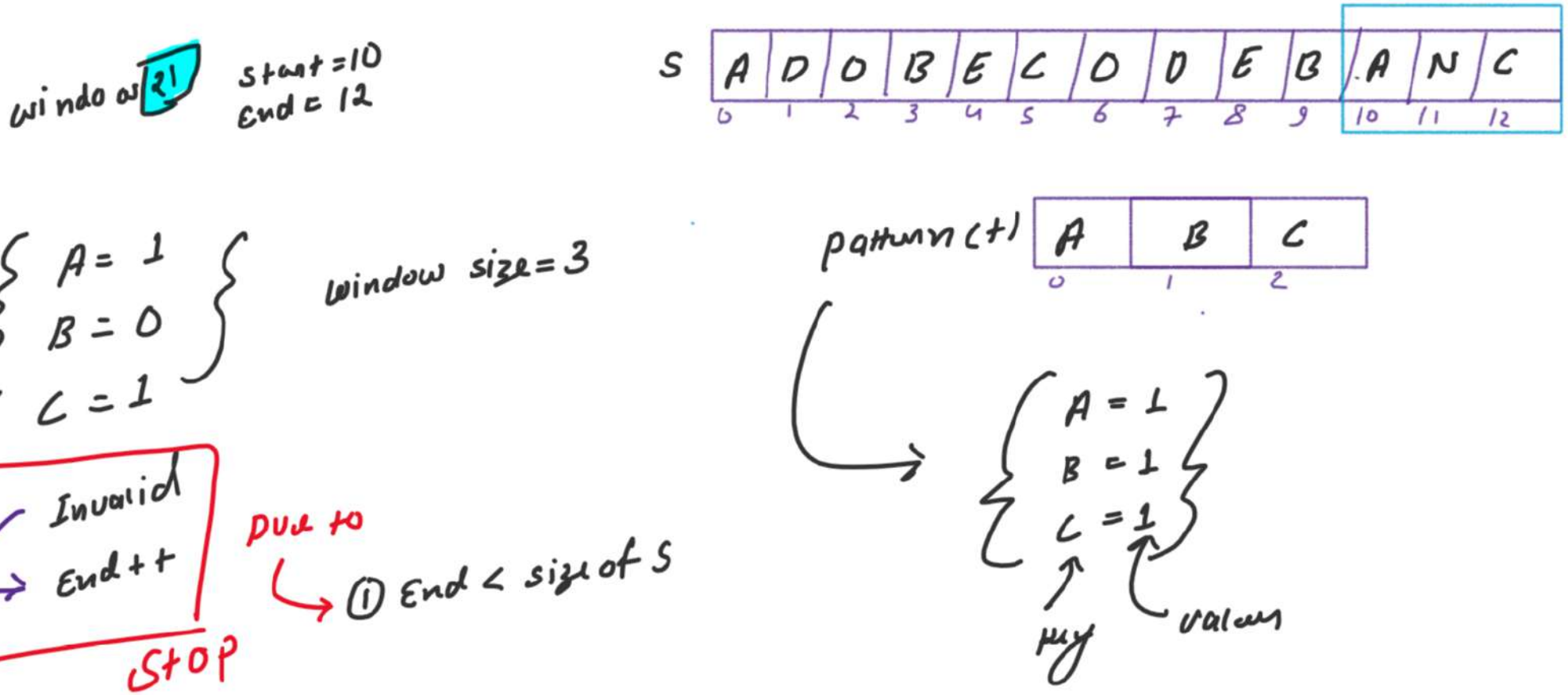
start++



A = 1
B = 1
C = 1

key

value


window 21
start = 10
End = 12
S
A D O B E C O D E B A N C
0 1 2 3 4 5 6 7 8 9 10 11 12
A = 1
B = 0
C = 1
Window size = 3
Invalid
End++
STOP
Due to
① End < size of S
pattern (+)
A B C
0 1 2
A = 1
B = 1
C = 1
key
value

Final Answer is **BANC**

↳ This is minimum window substring

```cpp
// Problem 01: Minimum Window Substring (Leetcode-76)

class Solution {
public:
    string minWindow(string s, string t) {
        // Invalid case
        int len1 = s.length();
        int len2 = t.length();
        int start = 0;
        if(len1 < len2) {
            return "";
        }

        // Valid case
        int ansIndex = -1;
        int ansLen = INT_MAX;

        unordered_map<char,int> sMap;
        unordered_map<char,int> tMap;
        // Store freq of pattern string → T.C. O(N)
        for(char ch: t) {
            tMap[ch]++;
        }

        // Initialise count variable

        if(ansIndex == -1) {
            return "";
        }
        else {
            return s.substr(ansIndex, ansLen);
        }
    }
};
```

```cpp
// Initialise count variable
// count variable means: yeh ek aisa character hai jo pattern and current window me present hai
int count = 0;
int end = 0;
while(end < s.length()) {  → T.C. O(M)

    // Update freq in sMap
    char ch = s[end];
    sMap[ch]++;
    if(sMap[ch] <= tMap[ch]) {
        // Valid character
        count++;
    }

    // Valid window
    if(count == len2) {
        // window found, jisme poora pattern existb krta h
        //-> minimise
        // Minimise sirf usi case me karunga, jis case me
        // 1. start index par jo character present h,
        // 2. ya toh extra hai ya
        // 3. aisa character jo pattern k andar hi nahi h, use bhi remove karo
        while(sMap[s[start]] > tMap[s[start]]) {
            sMap[s[start]]--;
            start++;
        }

        // Ab answer nikalo
        int windowKaSize = end - start + 1;
        if(windowKaSize < ansLen) {
            ansLen = windowKaSize;
            ansIndex = start;
        }
    }
    // Invalid window
    end++;
```

T.C. => O(N+M)