76. Minimum Window Substring (Hard)

Given two strings `s` and `t`, return the shortest substring of `s` such that every character in `t`, including duplicates, is present in the substring. If such a substring does not exist, return an empty string `""`.

You may assume that the correct output is always unique.

Example 1:

```
Input: s = "OUZODYXAZV", t = "XYZ"

Output: "YXAZ"
```

Explanation: `"YXAZ"` is the shortest substring that includes `"X"`, `"Y"`, and `"Z"` from string `t`.

Example 2:

```
Input: s = "xyz", t = "xyz"

Output: "xyz"
```

Example 3:

```
Input: s = "x", t = "xy"

Output: ""
```

Constraints:

- `1 <= s.length <= 1000`
- `1 <= t.length <= 1000`
- `s` and `t` consist of uppercase and lowercase English letters.

▼ 思路:

作法一：hashmap 分別儲存目標字母需要次數與字串出現字母次數，對整串字串以 sliding window搜尋

T:$O(n*m)$→ 每個新增長度都會搜尋比對所有目標字母數 , S:$O(m)$ → n為s長度，m出現過字母

作法二：hashmap 分別儲存目標字母需要次數與所有字串出現字母次數，對整串字串以sliding window搜尋，差別在於直接以額外變數儲存是否滿足需要字母次數無須重新比對所有目標字母

作法一

```
class Solution {
public:
    string minWindow(string s, string t) {
        unordered_map<char, int> check, target;

        if(t.size()==0) return t;
        pair<int ,int> pos = {0,0};
        for(char c:t) check[c] = 0,target[c]++;

        int maxf=INT_MAX,l=0,flag=1;
        for(int r=0;r<s.size();r++){
            flag=1;
            if(check.find(s[r])!=check.end()){
                check[s[r]]++;
            }
            for(auto c:check){
                if(c.second < target[c.first]){
                    flag=0;
                    break;
                }
            }
            while(flag){
                pos = (maxf<r-l+1) ? pos : make_pair(l,r);
                maxf = (maxf<r-l+1)? maxf: r-l+1;
```

```
            if(check.find(s[l])!=check.end()){
                if(check[s[l]]==target[s[l]]){
                    break;
                }
                check[s[l]]--;
            }
            l++;
        }
    }
    return (flag == 1) ? s.substr(pos.first,pos.second - pos.first+1) : "";
  }
};
```

作法二

```
class Solution {
public:
    string minWindow(string s, string t) {
        unordered_map<char, int> check, target;
        if(t.size()==0 || t.size()>s.size()) return "";
        pair<int ,int> pos = {0,0};
        for(char c:t) target[c]++;

        int maxf=INT_MAX,l=0,have=0,need=target.size();
        for(int r=0;r<s.size();r++){
            check[s[r]]++;
            if(target.find(s[r])!=target.end() && check[s[r]]==target[s[r]]){
                have++;
            }
            while(have == need){
                if(maxf>=r-l+1) pos = make_pair(l,r), maxf = r-l+1;
                if(target.find(s[l])!=target.end()){
                    if(check[s[l]]==target[s[l]]){
                        break;
                    }
                    check[s[l]]--;
```

```
            }
            l++;
          }
        }
        return (have == need) ? s.substr(pos.first,pos.second - pos.first+1) : "";
    }
};
```