1. Minimum Window Substring

Given a string S and a string T, find the minimum window in S which will contain all the characters in T in complexity O(n).

**Example:**

Input: S = "ADOBECODEBANC", T = "ABC"  
Output: "BANC"

**Note:**

* If there is no such window in S that covers all characters in T, return the empty string "".
* If there is such window, you are guaranteed that there will always be only one unique minimum window in S.

**解**

解法1 暴力枚举

解法2 滑动窗口，设置两个指针 l, r表示一个窗口

step1 移动指针r直到窗口[l, r]包含了子串中的所有字符

step2 移动指针l直到窗口[l, r]不能再收缩

step3 计算窗口长度，保留最小的窗口

step4 重复step1

class Solution {  
public:  
 string minWindow(string s, string t) {  
 if(s.size() == 0 || t.size() == 0)return "";  
 int cnt[256] = {0}, tmpCnt[256] = {0};  
 for(char ch : t){  
 cnt[ch]++;  
 tmpCnt[ch]++;  
 }  
 int l = 0, r = 0;  
 int minL = 0, minR = 0, minLen = 1 << 31 - 1;  
 int len = 0;  
 while(r < s.size()){  
 //找右边界  
 while(r < s.size() && len < t.size()){  
 char ch = s[r];  
 if(cnt[ch] != 0)tmpCnt[ch]--;  
 if(cnt[ch] != 0 && tmpCnt[ch] >= 0)len++;  
 r++;  
 }  
 if(len < t.size())break;  
 //收缩左边界  
 while(l < r){  
 char ch = s[l];  
 if(cnt[ch] != 0){  
 tmpCnt[ch]++;  
 if(tmpCnt[ch] > 0)break;  
 }  
 l++;  
 }  
 if(r - l < minLen){  
 minLen = r - l;  
 minL = l;  
 }  
 l++;  
 len--;  
 }  
 if(minLen > s.size())return "";  
 return s.substr(minL, minLen);  
 }  
   
};