7-26，7-27，7-29，8-1，8-5，8-10，8-16，8-23

▼ 两数之和

```
var twoSum = function(nums, target) {
    const idx = new Map(); // 创建一个空哈希表
    for (let j = 0; j < nums.length; j++) { // 枚举 j
        const x = nums[j];
        // 在左边找 nums[i]，满足 nums[i]+x=target
        if (idx.has(target - x)) { // 找到了
            return [idx.get(target - x), j]; // 返回两个数的下标
        }
        idx.set(x, j); // 保存 nums[j] 和 j
    }
};

console.log(twoSum([2, 7, 11, 15], 9));

const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

const lines = [];
rl.on('line', (line) => {
    lines.push(line);
});

rl.on('close', () => {
    // 如果输入是多组的，可以在这里用 for 或 while 循环处理 lines 数组
    // 这里我们假设只有一组输入
    if (lines.length < 2) {
        return;
```

```
    }

    // 解析输入
    const target = parseInt(lines[0]);
    const nums = lines[1].split(' ').map(Number);

    const twoSum = function(nums, target) {
        const idx = new Map(); // 创建一个空哈希表
        for (let j = 0; j < nums.length; j++) { // 枚举 j
            const x = nums[j];
            const complement = target - x;

            // 在左边找 nums[i]，满足 nums[i]+x=target
            if (idx.has(complement)) { // 找到了
                return [idx.get(complement), j]; // 返回两个数的下标
            }

            idx.set(x, j); // 保存 nums[j] 和 j
        }
        // 根据题目要求，如果没有找到可以返回一个空数组或者特定的值
        return [];
    };

    // 调用函数并获取结果
    const result = twoSum(nums, target);

    // 格式化并打印输出
    // ACM 模式通常要求输出以空格分隔
    if (result && result.length === 2) {
        console.log(result.join(' '));
    } else {
        // 如果题目保证一定有解，则不需要这个分支
        // console.log("No two sum solution");
    }
});
```

▼ **字母异位词分组**

```
var groupAnagrams = function(strs) {
    const m = new Map();
    for (const s of strs) {
        const key = s.split('').sort().join('');
        if (!m.has(key)) {
            m.set(key, []);
        }
        m.get(key).push(s);
    }
    return Array.from(m.values());
};

const input = ["eat", "tea", "tan", "ate", "nat", "bat"];
const result = groupAnagrams(input);
console.log(result); // 示例输出：[["eat","tea","ate"],["tan","nat"],["bat"]]

const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

let input = [];

rl.on('line', line => {
    input.push(line.trim());
});

rl.on('close', () => {
    // 假设单组输入格式为一行：以空格分隔的字符串数组，例如：["eat","tea","
    let strs = JSON.parse(input[0]);

    let m = new Map();
    for (let s of strs) {
        let sorted = s.split('').sort().join('');
        if (!m.has(sorted)) {
            m.set(sorted, []);
```

```
        }
        m.get(sorted).push(s);
    }

    let res = Array.from(m.values());
    // 输出每组异位词，每组一行
    for (let group of res) {
        console.log(group.join(' '));
    }
});
```

▼ **最长连续序列**

```
var longestConsecutive = function(nums) {
    let ans = 0;
    let st = new Set(nums);
    for(let n of st){
        if(st.has(n - 1)){
            continue;
        }
        let nxt = n + 1;
        while(st.has(nxt)){
            nxt++;
        }
        ans = Math.max(ans, nxt - n);
    }
    return ans;
};

// 示例用例
console.log(longestConsecutive([100, 4, 200, 1, 3, 2])); // 输出：4，对应序
console.log(longestConsecutive([0,3,7,2,5,8,4,6,0,1])); // 输出：9，对应序列

const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
```

```
});

let inputLines = [];
rl.on('line', (line) => {
    inputLines.push(line.trim());
    if (inputLines.length === 2) {
        const n = parseInt(inputLines[0]);
        const nums = inputLines[1].split(' ').map(Number);
        console.log(longestConsecutive(nums));
        rl.close();
    }
});

function longestConsecutive(nums) {
    let ans = 0;
    let st = new Set(nums);
    for (let n of st) {
        if (st.has(n - 1)) {
            continue; // 不是序列开头
        }
        let nxt = n + 1;
        while (st.has(nxt)) {
            nxt++;
        }
        ans = Math.max(ans, nxt - n);
    }
    return ans;
}
```

▼ 和为 K 的子数组

```
const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});
```

```
let inputLines = [];

rl.on('line', (line) => {
    inputLines.push(line.trim());
    if (inputLines.length === 2) {
        const n = parseInt(inputLines[0]);
        const nums = inputLines[1].split(' ').map(Number);
        const k = parseInt(inputLines[2]);

        console.log(subarraySum(nums, k));
        rl.close();
    }
});

function subarraySum(nums, k) {
    let ans = 0, s = 0;
    const cnt = new Map();
    cnt.set(0, 1);

    for (const x of nums) {
        s += x;
        ans += cnt.get(s - k) ?? 0;
        cnt.set(s, (cnt.get(s) ?? 0) + 1);
    }
    return ans;
}

console.log(subarraySum([1,1,1,2,3], 2));  // 输出 3
console.log(subarraySum([1,2,3], 3));      // 输出 2 ([1,2]和[3])
console.log(subarraySum([1,-1,0], 0));     // 输出 3 ([1,-1], [-1,0], [0])
```

▼ **移动零**

```
var moveZeroes = function(nums) {
    let j = 0;
    for (let i = 0; i < nums.length; i++) {
        if (nums[i] !== 0) {
            [nums[i], nums[j]] = [nums[j], nums[i]];
```

```
            j++;
        }
    }
};

let nums = [0, 1, 0, 3, 12];
moveZeroes(nums);
console.log(nums);

const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

rl.on('line', function (line) {
    let nums = line.trim().split(' ').map(Number);
    moveZeroes(nums);
    console.log(nums.join(' '));
});

function moveZeroes(nums) {
    let j = 0;
    for (let i = 0; i < nums.length; i++) {
        if (nums[i] !== 0) {
            [nums[i], nums[j]] = [nums[j], nums[i]];
            j++;
        }
    }
}
```

▼ **盛最多水的容器**

```
var maxArea = function(height) {
    let ans = 0, left = 0, right = height.length - 1;
    while(left < right){
        let maxAns = (right - left) * Math.min(height[left], height[right]);
```

```
        ans = Math.max(ans, maxAns);
        if(height[left] < height[right]){
            left++;
        }
        else{
            right--;
        }
    }
    return ans;
};

// 示例测试
let test1 = [1,8,6,2,5,4,8,3,7];
let test2 = [1,1];
let test3 = [4,3,2,1,4];
let test4 = [1,2,1];

console.log("Test1:", maxArea(test1)); // 输出: 49
console.log("Test2:", maxArea(test2)); // 输出: 1
console.log("Test3:", maxArea(test3)); // 输出: 16
console.log("Test4:", maxArea(test4)); // 输出: 2

const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

let inputLines = [];
rl.on("line", function (line) {
  inputLines.push(line.trim());
  if (inputLines.length === 3) {
    const n = parseInt(inputLines[0]); // 第一行：数组长度
    const height = inputLines[1].split(" ").map(Number); // 第二行：数组内容
    // 第三行留空或保留接口，可用于其他用途
    console.log(maxArea(height));
    rl.close();
```

```
  }
});

function maxArea(height) {
  let ans = 0; // 初始化最大面积
  let left = 0; // 左指针
  let right = height.length - 1; // 右指针

  while (left < right) {
    const width = right - left; // 当前左右指针之间的宽度
    const minHeight = Math.min(height[left], height[right]); // 当前可盛水高度
    const area = width * minHeight; // 当前面积
    ans = Math.max(ans, area); // 更新最大值

    // 移动高度较小的指针
    if (height[left] < height[right]) {
      left++;
    } else {
      right--;
    }
  }

  return ans; // 返回最大面积
}
```

▼ 三数之和

```
var threeSum = function(nums) {
    nums.sort((a,b) => a - b);
    let ans = [];
    let i = 0;
    let len = nums.length;
    for(i = 0; i < len - 2; i++){
        if(i > 0 && nums[i] === nums[i - 1]) continue;
        if(nums[i] + nums[i + 1] + nums[i + 2] > 0) break;
        if(nums[i] + nums[len - 1] + nums[len - 2] < 0) continue;
        let j = i + 1, k = len - 1;
        while(j < k){
```

```
            let sm = nums[i] + nums[j] + nums[k];
            if(sm > 0){
                k--;
            }
            else if(sm < 0){
                j++;
            }
            else{
                ans.push([nums[i], nums[j], nums[k]]);
                for(j++; j < k && nums[j] === nums[j - 1]; j++);
                for(k--; j < k && nums[k] === nums[k + 1]; k--);
            }
        }
    }
    return ans;
};

// 示例数组
const nums = [-1, 0, 1, 2, -1, -4];

// 打印结果
console.log(threeSum(nums)); // 期望输出：[[ -1, -1, 2 ], [ -1, 0, 1 ]]

const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

let inputLines = [];

rl.on('line', (line) => {
    inputLines.push(line.trim());
    if (inputLines.length === 2) {
        const n = parseInt(inputLines[0]);
        const nums = inputLines[1].split(' ').map(Number);
```

```
        const result = threeSum(nums);
        for (const triplet of result) {
            console.log(triplet.join(' '));
        }
        rl.close();
    }
});

function threeSum(nums) {
    nums.sort((a, b) => a - b);
    let ans = [];
    let len = nums.length;
    for (let i = 0; i < len - 2; i++) {
        if (i > 0 && nums[i] === nums[i - 1]) continue;
        if (nums[i] + nums[i + 1] + nums[i + 2] > 0) break;
        if (nums[i] + nums[len - 1] + nums[len - 2] < 0) continue;

        let j = i + 1, k = len - 1;
        while (j < k) {
            let sm = nums[i] + nums[j] + nums[k];
            if (sm > 0) {
                k--;
            } else if (sm < 0) {
                j++;
            } else {
                ans.push([nums[i], nums[j], nums[k]]);
                for (j++; j < k && nums[j] === nums[j - 1]; j++);
                for (k--; j < k && nums[k] === nums[k + 1]; k--);
            }
        }
    }
    return ans;
}
```

▼ **接雨水**

```
var trap = function(height) {
    let ans = 0, left = 0, right = height.length - 1, preMax = 0, sufMax = 0;
```

```
    while (left < right) {
        preMax = Math.max(preMax, height[left]);
        sufMax = Math.max(sufMax, height[right]);
        if (preMax < sufMax) {
            ans += preMax - height[left++];
        } else {
            ans += sufMax - height[right--];
        }
    }
    return ans;
};

console.log(trap([0,1,0,2,1,0,1,3,2,1,2,1]));  // 输出: 6
console.log(trap([4,2,0,3,2,5]));              // 输出: 9
console.log(trap([2,0,2]));                    // 输出: 2
console.log(trap([5,4,1,2]));                  // 输出: 1

const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

let inputLines = [];

rl.on('line', (line) => {
    inputLines.push(line.trim());
    if (inputLines.length === 2) {
        // 第一行是数组长度，第二行是数组元素（空格分隔）
        const n = parseInt(inputLines[0]);
        const height = inputLines[1].split(' ').map(Number);

        console.log(trap(height));
        rl.close();
    }
});
```

```
function trap(height) {
    let ans = 0, left = 0, right = height.length - 1, preMax = 0, sufMax = 0;
    while (left < right) {
        preMax = Math.max(preMax, height[left]);
        sufMax = Math.max(sufMax, height[right]);
        if (preMax < sufMax) {
            ans += preMax - height[left];
            left++;
        } else {
            ans += sufMax - height[right];
            right--;
        }
    }
    return ans;
}
```

▼ **无重复字符的最长子串**

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

let inputLines = [];

rl.on('line', (line) => {
  inputLines.push(line.trim());
  if (inputLines.length === 1) { // 单行字符串输入
    const s = inputLines[0];
    console.log(lengthOfLongestSubstring(s));
    rl.close();
  }
});

function lengthOfLongestSubstring(s) {
  let ans = 0, left = 0;
```

```
  const window = new Set();

  for (let right = 0; right < s.length; right++) {
    const c = s[right];
    while (window.has(c)) {
      window.delete(s[left]);
      left++;
    }
    window.add(c);
    ans = Math.max(ans, right - left + 1);
  }

  return ans;
}

console.log(lengthOfLongestSubstring("abcabcbb")); // 3
console.log(lengthOfLongestSubstring("bbbbb"));    // 1
console.log(lengthOfLongestSubstring("pwwkew"));   // 3
console.log(lengthOfLongestSubstring(""));         // 0
```

▼ **找到字符串中所有字母异位词**

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

let inputLines = [];

rl.on('line', (line) => {
  inputLines.push(line.trim());
  if (inputLines.length === 2) {
    const s = inputLines[0];
    const p = inputLines[1];
    const res = findAnagrams(s, p);
    console.log(res.join(' '));
```

```
    rl.close();
  }
});

function findAnagrams(s, p) {
  let ans = [];
  let cnt = new Array(26).fill(0);
  for (let c of p) {
    cnt[c.charCodeAt() - 'a'.charCodeAt()]++;
  }
  let left = 0;
  for (let right = 0; right < s.length; right++) {
    let c = s[right].charCodeAt() - 'a'.charCodeAt();
    cnt[c]--;
    while (cnt[c] < 0) {
      cnt[s[left].charCodeAt() - 'a'.charCodeAt()]++;
      left++;
    }
    if (right - left + 1 === p.length) {
      ans.push(left);
    }
  }
  return ans;
}


console.log(findAnagrams("cbaebabacd", "abc"));  // 输出: [0, 6]
console.log(findAnagrams("abab", "ab"));        // 输出: [0, 1, 2]
console.log(findAnagrams("af", "be"));          // 输出: []
```

▼ **滑动窗口最大值**

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
```

```
let inputLines = [];

rl.on('line', (line) => {
  inputLines.push(line.trim());
  if (inputLines.length === 3) {
    const n = parseInt(inputLines[0]);
    const nums = inputLines[1].split(' ').map(Number);
    const k = parseInt(inputLines[2]);

    const res = maxSlidingWindow(nums, k);
    console.log(res.join(' '));
    rl.close();
  }
});

function maxSlidingWindow(nums, k) {
  let ans = [];
  let qu = [];
  for (let i = 0; i < nums.length; i++) {
    while (qu.length && nums[qu[qu.length - 1]] <= nums[i]) {
      qu.pop();
    }
    qu.push(i);

    if (i - qu[0] >= k) {
      qu.shift();
    }

    if (i >= k - 1) {
      ans.push(nums[qu[0]]);
    }
  }
  return ans;
}

console.log(maxSlidingWindow([1,3,-1,-3,5,3,6,7], 3));  // 输出 [3,3,5,5,6,7]
```

```
console.log(maxSlidingWindow([9,11], 2));        // 输出 [11]
console.log(maxSlidingWindow([4,-2], 2));        // 输出 [4]
```

▼ 最小覆盖子串

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

let lines = [];

rl.on('line', (line) => {
  lines.push(line.trim());
  if (lines.length === 2) {
    const s = lines[0];
    const t = lines[1];

    const result = minWindow(s, t);
    console.log(result);
    rl.close();
  }
});

function minWindow(s, t) {
  const cnt = Array(128).fill(0);
  let missingTypes = 0;
  for (let c of t) {
    const code = c.codePointAt(0);
    if (cnt[code] === 0) missingTypes++;
    cnt[code]++;
  }

  const m = s.length;
  let ansLeft = -1, ansRight = m;
  let left = 0;
```

```
  for (let right = 0; right < m; right++) {
    const c = s[right].codePointAt(0);
    cnt[c]--;
    if (cnt[c] === 0) missingTypes--;

    while (missingTypes === 0) {
      if (right - left < ansRight - ansLeft) {
        ansLeft = left;
        ansRight = right;
      }
      const x = s[left].codePointAt(0);
      if (cnt[x] === 0) missingTypes++;
      cnt[x]++;
      left++;
    }
  }

  return ansLeft < 0 ? "" : s.substring(ansLeft, ansRight + 1);
}

console.log(minWindow("ADOBECODEBANC", "ABC"));  // 输出 "BANC"
console.log(minWindow("a", "a"));               // 输出 "a"
console.log(minWindow("a", "aa"));              // 输出 ""
console.log(minWindow("aaflslflsldkalskaaa", "aaa")); // 输出 "aaa"
```