**Interval**

<https://www.lintcode.com/problem/insert-interval>

**Merge (lists/arrays)**

<https://www.lintcode.com/problem/median-of-two-sorted-arrays>

**String (substring)**

<https://lintcode.com/problem/longest-palindrome>

[https://lintcode.com/problem/valid-palindrome](https://lintcode.com/problem/valid-palindrome/)

**Two pointers**

**前向--窗口类**

**总结：**

* 前向窗口类指针移动模版

<https://lintcode.com/problem/minimum-size-subarray-sum>

<https://lintcode.com/problem/longest-substring-without-repeating-characters>

<https://lintcode.com/problem/minimum-window-substring>

<https://www.lintcode.com/problem/longest-substring-with-at-most-k-distinct-characters>

**第k大问题（堆类型问题）-- 矩阵或多个数组**

**总结：**

* 堆型问题
* 见到集合求Min/Max就要想到堆
* 见到数组先排序

#############################

Kth Largest in N Arrays

• 用什么数据结构?

--- Answer:堆

• 方法:

--- 把N个数组先排序

--- 排序过后把每个数组最大的数字放入堆里面

--- 然后堆里面只维护前k个元素

--- 堆pop k次得到答案。

--- 时间复杂度N\*Len\*Log(len)+K\*logN (len 是**平均每个数组的长度**)

#############################

• 三道题相似点:

--- 求矩阵/数组的第k大

• 可以总结的规律

--- 规律1

• 见到需要维护一个集合的最小值/最大值的时候要想到用堆

--- 规律2

• 第k小的元素，Heap用来维护当前**候选集合**。

--- 规律3

• 见到数组要想到先排序

<https://www.lintcode.com/problem/kth-smallest-number-in-sorted-matrix>

<https://www.lintcode.com/problem/kth-largest-in-n-arrays>

<https://www.lintcode.com/problem/kth-smallest-sum-in-two-sorted-arrays>

**并查集 Union Find**

<https://www.lintcode.com/problem/connecting-graph>

<https://www.lintcode.com/problem/connecting-graph-ii>

<https://www.lintcode.com/problem/connecting-graph-iii>

<https://www.lintcode.com/problem/number-of-islands>

<https://www.lintcode.com/problem/number-of-islands-ii>

<https://www.lintcode.com/problem/graph-valid-tree>

**Trie Tree**

<https://www.lintcode.com/problem/implement-trie-prefix-tree> /[https://www.jiuzhang.com/solution/implement-trie](https://www.jiuzhang.com/solution/implement-trie/)/[https://www.jiuzhang.com/solution/trie/](https://www.jiuzhang.com/solution/trie/#tag-highlight)

<https://www.lintcode.com/problem/add-and-search-word/>

<https://www.lintcode.com/problem/word-search-ii>

<https://www.lintcode.com/problem/word-squares/>

**灌水问题 (trapping water)**

<https://www.lintcode.com/problem/trapping-rain-water>

<https://www.lintcode.com/problem/trapping-rain-water-ii>

**Heap**

<https://www.lintcode.com/problem/trapping-rain-water-ii>

**-- 中位数问题**

<https://www.lintcode.com/problem/data-stream-median>

**-- 中位数问题 TreeSet 解法(self-balanced binary search tree)**

中位数，那么就可以想到最大最小堆结构来表示中位数

首先建立最大最小堆，遍历数组分别往两个堆里插入数组元素

**这里为止最大堆的数字数目最多比最小堆多1，或者等于。每次的中位数就是最大堆的最大值**

当窗口容积满了后，找到应该出窗口的数值，将其删除，并补入新数据

<https://www.lintcode.com/problem/sliding-window-median> (Lin同学的PriorityQueue和E同学的TreeSet比较好)

**Stack**

1. 利用栈暂且保存有效信息
2. 翻转栈的运用
3. 栈优化dfs，变成非递归

<https://www.lintcode.com/problem/min-stack>

<https://www.lintcode.com/problem/implement-queue-by-two-stacks>

<https://www.lintcode.com/problem/expression-expand>

**单调栈**

<https://www.lintcode.com/problem/largest-rectangle-in-histogram>

<https://www.lintcode.com/problem/maximal-rectangle>

<https://www.lintcode.com/problem/max-tree>

**二分法**

NOTE: 能用二分法的都是可以用枚举法来做的。二分优于枚举的地方在于每次减少一半的检验量。从枚举出发，发现二分性(在某种条件下可以摈弃一部分的待查元素)

**二维二分**

[https://www.jiuzhang.com/solutions/find-peak-element/](https://www.jiuzhang.com/solutions/find-peak-element/#tag-other)

<https://www.lintcode.com/problem/find-peak-element-ii>

**按照值域二分**

<https://www.lintcode.com/problem/sqrtx>

<https://www.lintcode.com/problem/sqrtx-ii>

<https://www.lintcode.com/problem/wood-cut>

<https://www.lintcode.com/problem/find-the-duplicate-number>

**Sweep Line (扫描线)**

<https://www.lintcode.com/problem/number-of-airplanes-in-the-sky/>

<https://leetcode.com/problems/the-skyline-problem>

<https://www.lintcode.com/problem/building-outline>

**Deque (单调的deque)**

只会考这一题

<https://www.lintcode.com/problem/sliding-window-maximum>

**动态规划DP**

**1. 序列型**

<https://www.lintcode.com/problem/house-robber>

用滚动数组(rolling array)来优化存储空间从O(n)到O(1):

%2 模除2的技巧/性质：even % 2 == 0, odd % 2 == 1

一个数模除比自身大的数都等于它自身。

这里的滚动数组是dp数组（不是input的参数数组）。**"全部的"**dp数组的下标都要%2模除2

<https://www.lintcode.com/problem/house-robber-ii/>

**1.1 网格类的题目**

正方形用右下角作为定位角，长方形可以用左上角和右下角作为定位角（我还是用右下角）。

<https://www.lintcode.com/problem/maximal-square>

<https://www.lintcode.com/problem/maximal-square-ii/>

**2. 记忆化搜索**

* 一定用到dfs递归
* 递归函数中首先判断dp[]是否直接return。这一步代表memory
  + 判断条件可借助flag[]或dp[]中的特殊值
  + 如不能直接return，则嵌套dfs继续递归，千万别去想用for loop
* 要画递归树来推到出dp方程
* dp[i]初始化可通过看dp方程中哪些i不能求得就去初始化它
* dp方程的推进形式不是传统的dp[i]与dp[i - 1], dp[i -2] ...等的关系。而是通过进一步调用dfs递归函数推进

引子题:

[https://www.lintcode.com/problem/longest-continuous-increasing-subsequence](https://www.lintcode.com/problem/longest-continuous-increasing-subsequence/description)

<https://www.jiuzhang.com/solutions/longest-increasing-continuous-subsequence/>

**真题:**

[https://www.lintcode.com/problem/longest-increasing-continuous-subsequence-ii](https://www.lintcode.com/problem/longest-increasing-continuous-subsequence-ii/)

<https://www.lintcode.com/problem/coins-in-a-line>

<https://www.lintcode.com/problem/coins-in-a-line-ii>

**3. 区间类dp**

[https://www.lintcode.com/problem/coins-in-a-line-iii](https://www.lintcode.com/problem/coins-in-a-line-iii/description)

<https://www.lintcode.com/problem/stone-game>

<https://www.lintcode.com/problem/burst-balloons>

**4. 匹配类dp**

1. 恰当定义dp[i][j]的含义，i和j都是表示前i和前j个的意思。是序列型dp
2. 核心就是把dp[i - 1][j - 1], dp[i - 1][j], dp[i][j - 1]三种情况转化到dp[i][j]。即判断它们三者之间的大小再赋值给dp[i][j]

<https://www.lintcode.com/problem/edit-distance>

<https://www.lintcode.com/problem/longest-common-subsequence/>

**5. 背包类dp**

1. 矩阵中要添加0个物品和0体积
2. 矩阵中行是几个物品，列是几个体积，都是从0开始
3. 行列都是每次加1递增
4. 都可以用滚动数组在行上%2优化空间复杂度，要注意%2时也要定义成dp[2][m + 1]
5. 双重for循环i, j从0还是1开始选要看dp[][]方程是否有数组越界值，例如dp[i - 1][j]，i就要从1开始选

0-1背包：一个数只可以能取或不能取

无穷背包：一个数可以不取或取多次

不能取(A[i - 1] > j)能取（else即A[i - 1] <= j）是比较容量和当前数值

有上限的就是背包

<https://www.lintcode.com/problem/backpack>

<https://www.lintcode.com/problem/backpack-ii>

<https://www.lintcode.com/problem/backpack-iv>

<https://www.lintcode.com/problem/k-sum>