Array, linked list:

Dummy head: in linked list questions, if the new head can either be or not be the original head, then create a dummy head. If it always be the old head or new head, then no need to create the dummy head.

# Two pointers:

## 左右指针: 中间向两端扩散的双指针、两端向中间收缩的指针

多用于数组, especially sorted

while (left < right) -> [left, right)

while (left <= right) -> [left, right]

表示有效区间

## Binary search: one sorted array

[left, right), mid = left + (right-left) / 2

## 快慢指针

多用于链表，数组

## 滑动窗口

多用于substring matching（子数组）问题

# PreSum: for frequently calculate sum in a range

前缀和主要适⽤的场景是原始数组不会被修改的情况下，频繁查询某个区间的累加和。

class PrefixSum {

// 前缀和数组

private int[] prefix;

/\* 输⼊⼀个数组，构造前缀和 \*/

public PrefixSum(int[] nums) {

prefix = new int[nums.length + 1];

// prefix[0] = 0; auto

// 计算 nums 的累加和

for (int i = 0; i < nums.length; i++) {

prefix[i+1] = prefix[i] + nums[i];

}

}

/\* 查询闭区间 [i, j] 的累加和 \*/

public int query(int i, int j) {

return prefix[j + 1] - prefix[i];

}

}

Prefix[i] means the sum before nums[i]

# Diff array:

差分数组的主要适⽤场景是频繁对原始数组的某个区间的元素进⾏增减.

int[] diff = new int[nums.length];

// 构造差分数组

diff[0] = nums[0];

for (int i = 1; i < nums.length; i++) {

diff[i] = nums[i] - nums[i - 1];

}

Diff[i] means nums[i] – nums[i-1]

Binary Tree:

# Traversal:

## DFS:

Recursion(NLNRN)

PreOrder: NLR

InOrder: LNR

PostOrder: LRN

Stack

## BFS:

Queue

# divide and conquer

Recursion based algorithm:

Base case

Transition function

Divide and conquer:

Enumeration: 就是Traverse

Recursion based:

Backtracking:

Tree traversal, record route, make choice among choice list on each level, results are got on each leaf node.

results = []

def backtrack(路径, [选择列表参数], result):

if 满⾜结束条件 (reach leaf node):

results.add(路径)

return

for 选择 in 选择列表:

filter out invalid choice

if (choice is invalid) {

continue;

}

做选择

backtrack(路径, [选择列表参数], result) -> one level deeper

撤销选择 (if no more following choices, then this is not needed)

每一层都有已经走过的路径和选择列表，有时候选择列表要依靠选择列表参数来生成

最优解问题：

DP:

Greedy: 局部最优即全局最优

Notes:

1. 2 sum

sorted -> two pointers

Unsorted -> hashmap