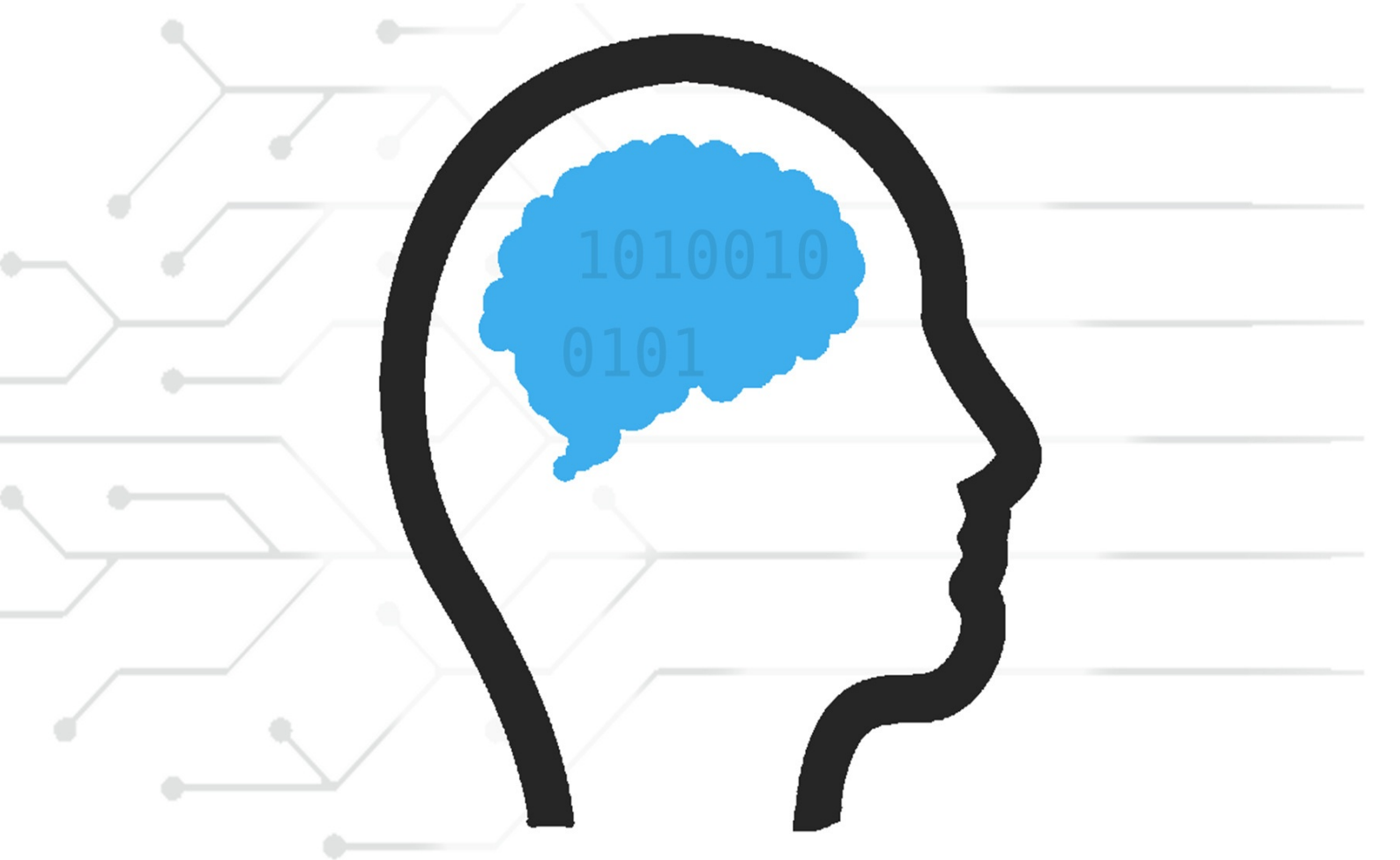


Algorithm

http://algorithm.yuanbin.me

1. Preface
2. Part I - Basics
3. Basics Data Structure
    i. Linked List
    ii. Binary Tree
    iii. Binary Search Tree
    iv. Huffman Compression
    v. Priority Queue
4. Basics Sorting
    i. Bubble Sort
    ii. Selection Sort
    iii. Insertion Sort
    iv. Merge Sort
    v. Quick Sort
    vi. Heap Sort
    vii. Bucket Sort
    viii. Counting Sort
    ix. Radix Sort
5. Basics Misc
    i. Bit Manipulation
    ii. Knapsack
6. Part II - Coding
7. String
    i. strStr
    ii. Two Strings Are Anagrams
    iii. Compare Strings
    iv. Anagrams
    v. Longest Common Substring
    vi. Rotate String
    vii. Reverse Words in a String
    viii. Valid Palindrome
    ix. Longest Palindromic Substring
    x. Space Replacement
8. Integer Array
    i. Remove Element
    ii. Zero Sum Subarray
    iii. Subarray Sum K
    iv. Subarray Sum Closest
    v. Recover Rotated Sorted Array
    vi. Product of Array Exclude Itself
    vii. Partition Array
    viii. First Missing Positive

    ix. 2 Sum
    x. 3 Sum
    xi. 3 Sum Closest
    xii. Remove Duplicates from Sorted Array
    xiii. Remove Duplicates from Sorted Array II
    xiv. Merge Sorted Array
    xv. Merge Sorted Array II
    xvi. Median
    xvii. Partition Array by Odd and Even
    xviii. Kth Largest Element
9. Binary Search
    i. Binary Search
    ii. Search Insert Position
    iii. Search for a Range
    iv. First Bad Version
    v. Search a 2D Matrix
    vi. Find Peak Element
    vii. Search in Rotated Sorted Array
    viii. Find Minimum in Rotated Sorted Array
    ix. Search a 2D Matrix II
    x. Median of two Sorted Arrays
    xi. Sqrt x
    xii. Wood Cut
10. Math and Bit Manipulation
    i. Single Number
    ii. Single Number II
    iii. Single Number III
    iv. O1 Check Power of 2
    v. Convert Integer A to Integer B
    vi. Factorial Trailing Zeroes
    vii. Unique Binary Search Trees
    viii. Update Bits
    ix. Fast Power
    x. Count 1 in Binary
    xi. Fibonacci
    xii. A plus B Problem
    xiii. Print Numbers by Recursion
    xiv. Majority Number
    xv. Majority Number II
    xvi. Majority Number III
    xvii. Digit Counts
    xviii. Ugly Number
11. Linked List
    i. Remove Duplicates from Sorted List
    ii. Remove Duplicates from Sorted List II
    iii. Remove Duplicates from Unsorted List

iv. Partition List
v. Two Lists Sum
vi. Two Lists Sum Advanced
vii. Remove Nth Node From End of List
viii. Linked List Cycle
ix. Linked List Cycle II
x. Reverse Linked List
xi. Reverse Linked List II
xii. Merge Two Sorted Lists
xiii. Merge k Sorted Lists
xiv. Reorder List
xv. Copy List with Random Pointer
xvi. Sort List
xvii. Insertion Sort List
xviii. Check if a singly linked list is palindrome
xix. Delete Node in the Middle of Singly Linked List

12. Binary Tree
    i. Binary Tree Preorder Traversal
    ii. Binary Tree Inorder Traversal
    iii. Binary Tree Postorder Traversal
    iv. Binary Tree Level Order Traversal
    v. Binary Tree Level Order Traversal II
    vi. Maximum Depth of Binary Tree
    vii. Balanced Binary Tree
    viii. Binary Tree Maximum Path Sum
    ix. Lowest Common Ancestor
    x. Invert Binary Tree
    xi. Diameter of a Binary Tree
    xii. Construct Binary Tree from Preorder and Inorder Traversal
    xiii. Construct Binary Tree from Inorder and Postorder Traversal
    xiv. Subtree
13. Binary Search Tree
    i. Insert Node in a Binary Search Tree
    ii. Validate Binary Search Tree
    iii. Search Range in Binary Search Tree
    iv. Convert Sorted Array to Binary Search Tree
    v. Convert Sorted List to Binary Search Tree
    vi. Binary Search Tree Iterator
14. Exhaustive Search
    i. Subsets
    ii. Unique Subsets
    iii. Permutation
    iv. Unique Permutations
    v. Next Permutation
    vi. Previous Permuation
    vii. Unique Binary Search Trees II

    viii. Permutation Index
    ix. Permutation Index II
    x. Permutation Sequence
    xi. Palindrome Partitioning
15. Dynamic Programming
    i. Triangle
    ii. Backpack
    iii. Minimum Path Sum
    iv. Unique Paths
    v. Unique Paths II
    vi. Climbing Stairs
    vii. Jump Game
    viii. Word Break
    ix. Longest Increasing Subsequence
    x. Palindrome Partitioning II
    xi. Longest Common Subsequence
    xii. Edit Distance
    xiii. Jump Game II
    xiv. Best Time to Buy and Sell Stock
    xv. Best Time to Buy and Sell Stock II
    xvi. Best Time to Buy and Sell Stock III
    xvii. Best Time to Buy and Sell Stock IV
    xviii. Distinct Subsequences
    xix. Interleaving String
    xx. Maximum Subarray
    xxi. Maximum Subarray II
16. Graph
    i. Topological Sorting
17. Data Structure
    i. Implement Queue by Two Stacks
    ii. Min Stack
18. Problem Misc
    i. Nuts and Bolts Problem
    ii. Heapify
    iii. String to Integer
19. Appendix I Interview and Resume
    i. Interview
    ii. Resume

/leetcode/lintcode

1. Part I//
2. Part II OJ https://leetcode.com/ http://www.lintcode.com/.
3. Part III

issue :)

https://github.com/billryan/algorithm-exercise Gitbook HTML GitHub star RSS

/~

- (Gitbook) http://algorithm.yuanbin.me
- : GitHub travis-ci
  1. EPUB. Gitbook - iPhone/iPad/MAC
  2. PDF. Gitbook, , - - Gitbook
  3. MOBI. Gitbook - Kindle . Kindle Kindle ...
- Google : `keywords site:algorithm.yuanbin.me`
- Swiftype : `Search this site`

/ .

-
- (**Google**)
  1. ()

2. ()
3. ()
4. (if)

- Code Complete

1.
2.
3.

OJ

- LeetCode Online Judge - OJ discuss lintcode OJ
- LintCode | Coding interview questions online training system - leetcodeOJ `source` CC150 locale OJ leetcode
- leetcode/lintcode/ | billryan - CS
- LeetCode - GitBook -
- FreeTymeKiyan/LeetCode-Sol-Res - Clean, Understandable Solutions and Resources on LeetCode Online Judge Algorithms Problems.
- soulmachine/leetcode - C++Java
- Woodstock Blog - IT
- ITint5 | IT -
- Acm,ACM -
- -IT,,IT - IT :)

- | -
- - julyedu.com - july
- | -
- VisuAlgo - visualising data structures and algorithms through animation -
- Data Structure Visualization - //
- -
- julycoding/The-Art-Of-Programming-By-July -
- 5
- —— - CSDN
- - Lucida - Google

- Algorithm Design ()
- The Algorithm Design Manual, slides - Skiena's Audio LecturesThe Algorithm Design Manual ()
- *Introduction to Algorithm* TAOCP
- *Cracking The Coding Interview*. CTCI(CC150)Google, Mircosoft, LinkedIn HROO Design, Database, System Design, Brain Teaser
- *Offer*Coding Interviews. (Harry He)Amazon.cn50show
- -- 150

()

- Data Structures and Algorithms in C++ -by Michael T. Goodrich, Roberto Tamassia and David M. Mount

  C++

- Data Structures • (MOOC)

  MOOCC++STLvectorlistsetOJ

Part I - Basics

Reference

- VisuAlgo - visualising data structures and algorithms through animation -
- Data Structure Visualization - //

Data Structure -

Linked List -

(linear list)(array)

O(1) O(n)

(call-by-index)O(1)O(1)

(singly linked list)

`1 -> 2 -> 3 -> null` `3 -> 2 -> 1 -> null`

- curt.next curt null
- null

```
public ListNode reverse(ListNode head) {
    ListNode prev = null;
    while (head != null) {
        ListNode next = head.next;
        head.next = prev;
        prev = head;
        head = next;
    }
    return prev;
}
```

`prev -> next = prev -> next -> next` Dummy Node

(robustness)

- curt.next curt null
- null

Dummy Node

Dummy node

Dummy node Dummy node head head dummy -> headDummy node head
dummy node head Remove Duplicates From Sort
head dummy nodehead

head dummy node dummy.next

(fast/slow pointer)

21

- `*fast` `*slow` `*fast` `*slow` 2 `*fast` `slow`
- `*fast` `*slow` `*fast` `*slow` 2 `*fast = NU`
  `*slow`

Binary Tree -

(binary search tree)(binary heap)

i(1) 2^{i-1} k $2^k - 1$ T $n_0, 2$ $n_2,$ $n_0 = n_2 + 1$

$k,$ $2^k - 1$ k n k 1 n

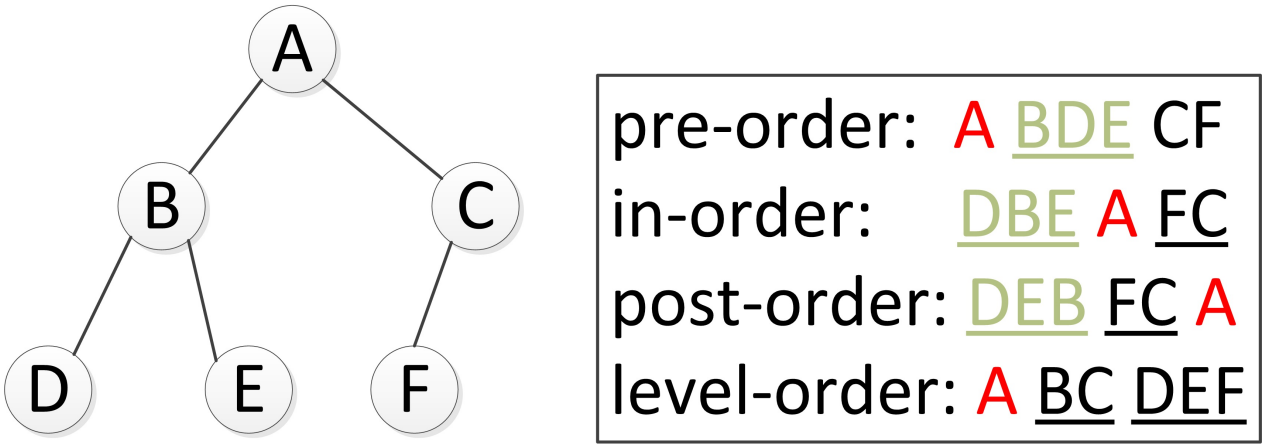
Tree traversal

()

- (depth-first)
  1. (pre-order)
  2. (in-order)
  3. (post-order)
- (breadth-first)(level-order)

A

////(stack)(queue)



LeetCode

```
struct TreeNode {
    int val;
    TreeNode *left;
    TreeNode *right;
    TreeNode(int x) : val(x), left(NULL), right(NULL) {}
};
```

——

Chapter 11

——(Divide & Conquer)

……

-
  1.
  2.
  3.
  4.
-
  1. Divide
  2. Conquer
  3. Combine
-
  1.
  2.
  3. Strassen
  4.
  5. (merge sort)
  6.
  7.
  8. (tower of Hanoi)

Binary Search Tree -

(BST)(key)(value)

()

-
-
-
-

```
template<typename Key, typename Value>
struct BSTNode{
    Key key;
    Value val;
    BSTNode* left;
    BSTNode* right;
    BSTNode(Key k, Value v, BSTNode* l = NULL, BSTNode* r = NULL)
        :key(k), val(v), left(l), right(r){}
};
```

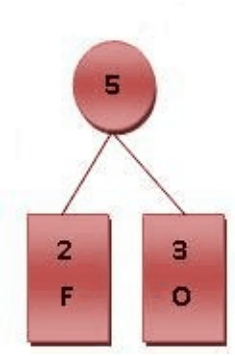
Huffman Compression -

78

(variable-length code)(prefix code)Huffman

F, O, R, G, E, T

Symbol	F	O	R	G	E	T
Frequence	2	3	4	4	5	7
Code	000	001	100	101	01	10



Huffman | - CoolShell.cn -

Priority Queue -

(operating system)

(Abstract Data TypeADT)(interface)(method)

- (insert_with_priority)
- (pull_highest_priority_element)
- (peak) C++STL

```
template <typename T> class priority_queue{
    void push (const T& val);
    void pop ();
    const T& top() const;
};
```

(heap)

Reference

- -
- STL: priority_queue

Basics Sorting -

———

1. -()
2. -
    -
    -
    -

OJ(rule of thumb)(back-of-the-envelop calculation)

```
10^9 operations per second
```

1s 10^3 $O(n^2)$ 10^5 $O(n^2)$ $O(n \log n)$

()

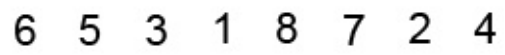
- Comparison Sorting
    1. Bubble Sort
    2. Selection Sort
    3. Insertion Sort
    4. Shell Sort
    5. Merge Sort
    6. Quck Sort
    7. Heap Sort
- Bucket Sort
- Counting Sort
- Radix Sort

-
-

Reference

- Sorting algorithm - Wikipedia, the free encyclopedia -
- Big-O cheatsheet -
- | Jark's Blog - Python
- Startup-_ -
- slide

Bubble Sort -



Implementation

Python

```
#!/usr/bin/env python


def bubbleSort(alist):
    for i in xrange(len(alist)):
        print(alist)
        for j in xrange(1, len(alist) - i):
            if alist[j - 1] > alist[j]:
                alist[j - 1], alist[j] = alist[j], alist[j - 1]

    return alist

unsorted_list = [6, 5, 3, 1, 8, 7, 2, 4]
print(bubbleSort(unsorted_list))
```

Java

```
public class Sort {
    public static void main(String[] args) {
        int unsortedArray[] = new int[]{6, 5, 3, 1, 8, 7, 2, 4};
        bubbleSort(unsortedArray);
        System.out.println("After sort: ");
        for (int item : unsortedArray) {
            System.out.print(item + " ");
        }
    }

    public static void bubbleSort(int[] array) {
        int len = array.length;
        for (int i = 0; i < len; i++) {
            for (int item : array) {
```

```
                System.out.print(item + " ");
            }
            System.out.println();
            for (int j = 1; j < len - i; j++) {
                if (array[j - 1] > array[j]) {
                    int temp = array[j - 1];
                    array[j - 1] = array[j];
                    array[j] = temp;
                }
            }
        }
    }
}
```

C++

```
void bubbleSort(vector<int> & arr){
    for(int i = 0; i < arr.size(); i++){
        for(int j = 1; j < arr.size() - i; j++){
            if(arr[j - 1] > arr[j])){
                std::swap(arr[j-1], arr[j]);
            }
        }
    }
    return arr;
}
```

$O(n^2)$, temp $O(1)$. $O(n^2)$

```
void bubbleSort(vector<int> & arr){
    bool unsorted = true;
    for(int i = 0; i < arr.size() && unsorted; i++){
        unsorted = false;
        for(int j = 1; j < arr.size() - i; j++){
            if(arr[j - 1] > arr[j])){
                std::swap(arr[j-1], arr[j]);
                unsorted = true;
            }
        }
    }
    return arr;
}
```

Reference

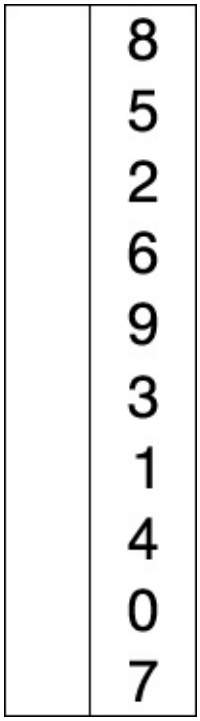
- -

Selection Sort -

1.
2.

- =(N-1)+(N-2)+(N-3)+...+2+1~N^2/2
- =N
-
-

File:Selection-Sort-Animation.gif - IB Computer Science



Implementation

Python

```
#!/usr/bin/env python


def selectionSort(alist):
    for i in xrange(len(alist)):
        print(alist)
        min_index = i
        for j in xrange(i + 1, len(alist)):
            if alist[j] < alist[min_index]:
```

```
                min_index = j
        alist[min_index], alist[i] = alist[i], alist[min_index]
    return alist

unsorted_list = [8, 5, 2, 6, 9, 3, 1, 4, 0, 7]
print(selectionSort(unsorted_list))
```

Java

```
public class Sort {
    public static void main(String[] args) {
        int unsortedArray[] = new int[]{8, 5, 2, 6, 9, 3, 1, 4, 0, 7};
        selectionSort(unsortedArray);
        System.out.println("After sort: ");
        for (int item : unsortedArray) {
            System.out.print(item + " ");
        }
    }

    public static void selectionSort(int[] array) {
        int len = array.length;
        for (int i = 0; i < len; i++) {
            for (int item : array) {
                System.out.print(item + " ");
            }
            System.out.println();
            int min_index = i;
            for (int j = i + 1; j < len; j++) {
                if (array[j] < array[min_index]) {
                    min_index = j;
                }
            }
            int temp = array[min_index];
            array[min_index] = array[i];
            array[i] = temp;
        }
    }
}
```

C++

```
void selectionSort(vector<int> & arr){
    int min_idx = 0;
    for(int i = 0; i < arr.size(); i++){
        min_idx = i;
        for(int j = i + 1; j < arr.size(); j++){
            if (arr[j] < arr[min_idx])
                min_idx = j;
        }
        std::swap(arr[i], arr[min_idx]);
    }
}
```

Reference

- -
- The Selection Sort — Problem Solving with Algorithms and Data Structures

Insertion Sort -

()in-place(O(1))

1.
2.
3.
4. 3
5.
6. 2~5

-
- >=<=
- 0
- 1N-1i(a[i])
- ~$N^2/2$ $N^2/2$ $N-10$
- ~$N^2/4$~$N^2/4$

Implementation

Python

```
#!/usr/bin/env python


def insertionSort(alist):
    for i, item_i in enumerate(alist):
        print alist
        index = i
```

```
        while index > 0 and alist[index - 1] > item_i:
            alist[index] = alist[index - 1]
            index -= 1

        alist[index] = item_i

    return alist

unsorted_list = [6, 5, 3, 1, 8, 7, 2, 4]
print(insertionSort(unsorted_list))
```

Java

```
public class Sort {
    public static void main(String[] args) {
        int unsortedArray[] = new int[]{6, 5, 3, 1, 8, 7, 2, 4};
        insertionSort(unsortedArray);
        System.out.println("After sort: ");
        for (int item : unsortedArray) {
            System.out.print(item + " ");
        }
    }

    public static void insertionSort(int[] array) {
        int len = array.length;
        for (int i = 0; i < len; i++) {
            int index = i, array_i = array[i];
            while (index > 0 && array[index - 1] > array_i) {
                array[index] = array[index - 1];
                index -= 1;
            }
            array[index] = array_i;

            /* print sort process */
            for (int item : array) {
                System.out.print(item + " ");
            }
            System.out.println();
        }
    }
}
```

(C++)

```
template<typename T>
void insertion_sort(T arr[], int len) {
    int i, j;
    T temp;
    for (int i = 1; i < len; i++) {
        temp = arr[i];
        for (int j = i - 1; j >= 0 && arr[j] > temp; j--) {
            a[j + 1] = a[j];
        }
        arr[j + 1] = temp;
    }
```

```
}
```

Shell sort

hh

(C++):

```
template<typename T>
void shell_sort(T arr[], int len) {
    int gap, i, j;
    T temp;
    for (gap = len >> 1; gap > 0; gap >>= 1)
        for (i = gap; i < len; i++) {
            temp = arr[i];
            for (j = i - gap; j >= 0 && arr[j] > temp; j -= gap)
                arr[j + gap] = arr[j];
            arr[j + gap] = temp;
        }
}
```

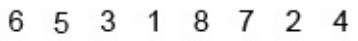
hhhwiki hSedgewick $\Theta(N^{4/3})$

Reference

- -
- -
- The Insertion Sort — Problem Solving with Algorithms and Data Structures

Merge Sort -

(divide and conquer)



Python

```python
#!/usr/bin/env python


class Sort:
    def mergeSort(self, alist):
        if len(alist) <= 1:
            return alist

        mid = len(alist) / 2
        left = self.mergeSort(alist[:mid])
        print("left = " + str(left))
        right = self.mergeSort(alist[mid:])
        print("right = " + str(right))
        return self.mergeSortedArray(left, right)

    #@param A and B: sorted integer array A and B.
    #@return: A new sorted integer array
    def mergeSortedArray(self, A, B):
        sortedArray = []
        l = 0
        r = 0
        while l < len(A) and r < len(B):
            if A[l] < B[r]:
                sortedArray.append(A[l])
                l += 1
            else:
                sortedArray.append(B[r])
                r += 1
        sortedArray += A[l:]
        sortedArray += B[r:]

        return sortedArray

unsortedArray = [6, 5, 3, 1, 8, 7, 2, 4]
merge_sort = Sort()
```

```
print(merge_sort.mergeSort(unsortedArray))
```

(in-place)

Java

```
public class MergeSort {
    public static void main(String[] args) {
        int unsortedArray[] = new int[]{6, 5, 3, 1, 8, 7, 2, 4};
        mergeSort(unsortedArray);
        System.out.println("After sort: ");
        for (int item : unsortedArray) {
            System.out.print(item + " ");
        }
    }

    private static void merge(int[] array, int low, int mid, int high) {
        int[] helper = new int[array.length];
        // copy array to helper
        for (int k = low; k <= high; k++) {
            helper[k] = array[k];
        }
        // merge array[low...mid] and array[mid + 1...high]
        int i = low, j = mid + 1;
        for (int k = low; k <= high; k++) {
            // k means current location
            if (i > mid) {
            // no item in left part
                array[k] = helper[j];
                j++;
            } else if (j > high) {
            // no item in right part
                array[k] = helper[i];
                i++;
            } else if (helper[i] > helper[j]) {
            // get smaller item in the right side
                array[k] = helper[j];
                j++;
            } else {
            // get smaller item in the left side
                array[k] = helper[i];
                i++;
            }
        }
    }

    public static void sort(int[] array, int low, int high) {
        if (high <= low) return;
        int mid = low + (high - low) / 2;
        sort(array, low, mid);
        sort(array, mid + 1, high);
        merge(array, low, mid, high);
        for (int item : array) {
            System.out.print(item + " ");
        }
```

```
        System.out.println();
    }

    public static void mergeSort(int[] array) {
        sort(array, 0, array.length - 1);
    }
}
```

C++

```
void merge (vector<int>& arr, int low, int mid, int high){
    vector<int> helper(arr.size());
    for(int k = low; k <= high; k++){
        helper[k] = arr[k];
    }
    int i = low, j = mid+1;
    for(int k = low; k <= high; k++){
        if(i > mid){
            arr[k] = helper[j];
            j++;
        }
        else if(j > high){
            arr[k] = helper[i];
            i++;
        }
        else if(helper[j] > helper[i]){
            arr[k] = helper[j];
            j++;
        }
        else{
            arr[k] = helper[i];
            i++;
        }
    }
}

void mergeSort(vector<int>& arr, int low, int high){
    int mid = low + (high - low)/2;
    mergeSort(arr, low, mid);
    mergeSort(arr, mid + 1, high);
    merge(arr, low, mid, high);
}
```

$O(N \log N), \qquad O(N)$

Reference

- Mergesort - Robert Sedgewick

Bucket Sort

1.
2. Divide -
3.
4. Conquer -

Reference

- Bucket Sort Visualization -
-

Basics Miscellaneous

Bit Manipulation

(bitwise and)(bitwise or)(bitwise not)nn

XOR - (exclusive or)

> 01

```
x ^ 0 = x
x ^ 1s = ~x // 1s = ~0
x ^ (~x) = 1s
x ^ x = 0 // interesting and important!
a ^ b = c => a ^ c = b, b ^ c = a // swap
a ^ b ^ c = a ^ (b ^ c) = (a ^ b) ^ c // associative
```

(shift operation)

/2 `0b0010 * 0b0110` `0b0110 << 2`.

1. `x` `n` - `x & (~0 << n)`
2. `x` `n` (01) - `x & (1 << n)`
3. `x` `n` - `(x >> n) & 1`
4. `n` `1` - `x | (1 << n)`
5. `n` `0` - `x & (~(1 << n))`
6. `x` `n` () - `x & ((1 << n) - 1)`
7. `n` 0() - `x & (~((1 << (n + 1)) - 1))`
8. `n` `v` ; `v` 110 - `mask = ~(1 << n); x = (x & mask) | (v << i)`

(Bitmap)

flag array
int $1/32$.(int32)
100setbittestbit

```
#define N 1000000 // 1 million
#define WORD_LENGTH sizeof(int) * 8 //sizeof8int

//bitsii0~1000000
void setbit(unsigned int* bits, unsigned int i){
    bits[i / WORD_LENGTH] |= 1<<(i % WORD_LENGTH);
}

int testbit(unsigned int* bits, unsigned int i){
```

```
    return bits[i/WORD_LENGTH] & (1<<(i % WORD_LENGTH));
}

unsigned int bits[N/WORD_LENGTH + 1];
```

Reference

- 1 » NoAlGo
- 2 » NoAlGo
- | Matrix67: The Aha Moments
- *cc150* chapter 8.5 and chapter 9.5
- 2
- Elementary Algorithms Larry LIU Xinyu

String -

strStr

Source

- leetcode: Implement strStr() | LeetCode OJ
- lintcode: lintcode - (13) strstr

```
strstr (a.k.a find sub string), is a useful function in string operation.
You task is to implement this function.

For a given source string and a target string,
you should output the "first" index(from 0) of target string in source string.

If target is not exist in source, just return -1.

Example
If source="source" and target="target", return -1.

If source="abcdabcdefg" and target="bcd", return 1.

Challenge
O(n) time.

Clarification
Do I need to implement KMP Algorithm in an interview?

    - Not necessary. When this problem occurs in an interview,
    the interviewer just want to test your basic implementation ability.
```

forKMP

Java

```java
/**
 * http://www.jiuzhang.com//solutions/implement-strstr
 */
class Solution {
    /**
     * Returns a index to the first occurrence of target in source,
     * or -1  if target is not part of source.
     * @param source string to be scanned.
     * @param target string containing the sequence of characters to match.
     */
    public int strStr(String source, String target) {
        if (source == null || target == null) {
            return -1;
        }
```

```
        int i, j;
        for (i = 0; i < source.length() - target.length() + 1; i++) {
            for (j = 0; j < target.length(); j++) {
                if (source.charAt(i + j) != target.charAt(j)) {
                    break;
                } //if
            } //for j
            if (j == target.length()) {
                return i;
            }
        } //for i

        // did not find the target
        return -1;
    }
}
```

1. `source` `target`
2. `i` `i < source.length()` `source.charAt(i + j)`
3. 1 `==` 2 `s1``s2` `target``source` 3if `{return -1;}` 4Java C++ 5 `int i, j;`
4. for `i` , `j`

Another Similar Question

```
/**
 * http://www.jiuzhang.com//solutions/implement-strstr
 */
public class Solution {
    public String strStr(String haystack, String needle) {
        if(haystack == null || needle == null) {
            return null;
        }
        int i, j;
        for(i = 0; i < haystack.length() - needle.length() + 1; i++) {
            for(j = 0; j < needle.length(); j++) {
                if(haystack.charAt(i + j) != needle.charAt(j)) {
                    break;
                }
            }
            if(j == needle.length()) {
                return haystack.substring(i);
            }
        }
        return null;
    }
}
```

Two Strings Are Anagrams

Source

- CC150: (158) Two Strings Are Anagrams
- leetcode: Valid Anagram | LeetCode OJ

```
Write a method anagram(s,t) to decide if two strings are anagrams or not.

Example
Given s="abcd", t="dcab", return true.

Challenge
O(n) time, O(1) extra space
```

1 - hashmap

`false`.

C++

```cpp
class Solution {
public:
    /**
     * @param s: The first string
     * @param t: The second string
     * @return true or false
     */
    bool anagram(string s, string t) {
        if (s.empty() || t.empty()) {
            return false;
        }
        if (s.size() != t.size()) {
            return false;
        }

        int letterCount[256] = {0};

        for (int i = 0; i != s.size(); ++i) {
            ++letterCount[s[i]];
            --letterCount[t[i]];
        }
        for (int i = 0; i != t.size(); ++i) {
            if (letterCount[t[i]] != 0) {
                return false;
            }
        }

        return true;
    }
```

```
};
```

1. ()
2. 256
3. s t `letterCount` 0 `false`.

() `for` `t.size() > 256` 256 `t.size()`, `i` `t[i]`.

$O(2n),$ $O(256).$

2 -

1 hashmap hashmap

C++

```
class Solution {
public:
    /**
     * @param s: The first string
     * @param b: The second string
     * @return true or false
     */
    bool anagram(string s, string t) {
        if (s.empty() || t.empty()) {
            return false;
        }
        if (s.size() != t.size()) {
            return false;
        }

        sort(s.begin(), s.end());
        sort(t.begin(), t.end());

        if (s == t) {
            return true;
        } else {
            return false;
        }
    }
};
```

s t

C++ STL sort $O(n)$ $O(n^2)$ `s == t` $O(n)$.

Reference

- *CC150 Chapter 9.1* p109

Compare Strings

Source

- lintcode: (55) Compare Strings

```
Compare two strings A and B, determine whether A contains all of the characters in B.

The characters in string A and B are all Upper Case letters.

Example
For A = "ABCD", B = "ABC", return true.

For A = "ABCD" B = "AABC", return false.
```

Two Strings Are Anagrams | Data Structure and Algorithm BAB="AABC"A A="ABCD"Afalse.

strstr AB A B

C++

```cpp
class Solution {
public:
    /**
     * @param A: A string includes Upper Case letters
     * @param B: A string includes Upper Case letter
     * @return:  if string A contains all of the characters in B return true
     *           else return false
     */
    bool compareStrings(string A, string B) {
        if (A.size() < B.size()) {
            return false;
        }

        const int AlphabetNum = 26;
        int letterCount[AlphabetNum] = {0};
        for (int i = 0; i != A.size(); ++i) {
            ++letterCount[A[i] - 'A'];
        }
        for (int i = 0; i != B.size(); ++i) {
            --letterCount[B[i] - 'A'];
            if (letterCount[B[i] - 'A'] < 0) {
                return false;
            }
        }

        return true;
```

```
    }
};
```

1. B A `false`,
2.

A B $O(2n), \quad O(26).$

Rotate String

Source

- lintcode: (8) Rotate String

```
Given a string and an offset, rotate string by offset. (rotate from left to right)

Example
Given "abcdefg"

for offset=0, return "abcdefg"

for offset=1, return "gabcdef"

for offset=2, return "fgabcde"

for offset=3, return "efgabcd"

...
```

---

offset

Python

```python
class Solution:
    """
    param A: A string
    param offset: Rotate string with offset.
    return: Rotated string.
    """
    def rotateString(self, A, offset):
        if A is None or len(A) == 0:
            return A

        offset %= len(A)
        before = A[:len(A) - offset]
        after = A[len(A) - offset:]
        # [::-1] means reverse in Python
        A = before[::-1] + after[::-1]
        A = A[::-1]

        return A
```

C++

```
class Solution {
public:
  /**
     * param A: A string
     * param offset: Rotate string with offset.
     * return: Rotated string.
     */
    string rotateString(string A, int offset) {
        if (A.empty() || A.size() == 0) {
            return A;
        }

        int len = A.size();
        offset %= len;
        reverse(A, 0, len - offset - 1);
        reverse(A, len - offset, len - 1);
        reverse(A, 0, len - 1);
        return A;
    }

private:
    void reverse(string &str, int start, int end) {
        while (start < end) {
            char temp = str[start];
            str[start] = str[end];
            str[end] = temp;
            start++;
            end--;
        }
    }
};
```

Java

```
public class Solution {
    /*
     * param A: A string
     * param offset: Rotate string with offset.
     * return: Rotated string.
     */
    public char[] rotateString(char[] A, int offset) {
        if (A == null || A.length == 0) {
            return A;
        }

        int len = A.length;
        offset %= len;
        reverse(A, 0, len - offset - 1);
        reverse(A, len - offset, len - 1);
        reverse(A, 0, len - 1);

        return A;
    }

    private void reverse(char[] str, int start, int end) {
        while (start < end) {
            char temp = str[start];
```

```
            str[start] = str[end];
            str[end] = temp;
            start++;
            end--;
        }
    }
};
```

1. A0
2. `offset` A `len`
3.

Python slice Pythonic!

$O(n),$ $O(1).$ 3 $O(n),$ $O(1).$

Reference

- Reverse a string in Python - Stack Overflow

Valid Palindrome

- tags: [palindrome]

Source

- leetcode: Valid Palindrome | LeetCode OJ
- lintcode: (415) Valid Palindrome

```
Given a string, determine if it is a palindrome,
considering only alphanumeric characters and ignoring cases.

Example
"A man, a plan, a canal: Panama" is a palindrome.

"race a car" is not a palindrome.
Note
Have you consider that the string might be empty?
This is a good question to ask during an interview.
For the purpose of this problem,
we define empty string as valid palindrome.

Challenge
O(n) time without extra memory.
```

Check if a singly linked list is palindrome.

Python

```python
class Solution:
    # @param {string} s A string
    # @return {boolean} Whether the string is a valid palindrome
    def isPalindrome(self, s):
        if not s:
            return True

        l, r = 0, len(s) - 1

        while l < r:
            # find left alphanumeric character
            if not s[l].isalnum():
                l += 1
                continue
            # find right alphanumeric character
            if not s[r].isalnum():
                r -= 1
                continue
            # case insensitive compare
```

```
            if s[l].lower() == s[r].lower():
                l += 1
                r -= 1
            else:
                return False
        #
        return True
```

C++

```
class Solution {
public:
    /**
     * @param s A string
     * @return Whether the string is a valid palindrome
     */
    bool isPalindrome(string& s) {
        if (s.empty()) return true;

        int l = 0, r = s.size() - 1;
        while (l < r) {
            // find left alphanumeric character
            if (!isalnum(s[l])) {
                ++l;
                continue;
            }
            // find right alphanumeric character
            if (!isalnum(s[r])) {
                --r;
                continue;
            }
            // case insensitive compare
            if (tolower(s[l]) == tolower(s[r])) {
                ++l;
                --r;
            } else {
                return false;
            }
        }

        return true;
    }
};
```

Java

```
public class Solution {
    /**
     * @param s A string
     * @return Whether the string is a valid palindrome
     */
    public boolean isPalindrome(String s) {
        if (s == null || s.isEmpty()) return true;
```

```
        int l = 0, r = s.length() - 1;
        while (l < r) {
            // find left alphanumeric character
            if (!Character.isLetterOrDigit(s.charAt(l))) {
                l++;
                continue;
            }
            // find right alphanumeric character
            if (!Character.isLetterOrDigit(s.charAt(r))) {
                r--;
                continue;
            }
            // case insensitive compare
            if (Character.toLowerCase(s.charAt(l)) == Character.toLowerCase(s.charAt(r)))
                l++;
                r--;
            } else {
                return false;
            }
        }

        return true;
    }
}
```

1. ()
2.

API

$O(n), \quad O(1).$

Longest Palindromic Substring

- tags: [palindrome]

Source

- leetcode: Longest Palindromic Substring | LeetCode OJ
- lintcode: (200) Longest Palindromic Substring

```
Given a string S, find the longest palindromic substring in S.
You may assume that the maximum length of S is 1000,
and there exists one unique longest palindromic substring.

Example
Given the string = "abcdzdcab", return "cdzdc".
Challenge
O(n2) time is acceptable. Can you do it in O(n) time.
```

1 - (brute force)

Python

```python
class Solution:
    # @param {string} s input string
    # @return {string} the longest palindromic substring
    def longestPalindrome(self, s):
        if not s:
            return ""

        n = len(s)
        longest, left, right = 0, 0, 0
        for i in xrange(0, n):
            for j in xrange(i + 1, n + 1):
                substr = s[i:j]
                if self.isPalindrome(substr) and len(substr) > longest:
                    longest = len(substr)
                    left, right = i, j
        # construct longest substr
        result = s[left:right]
        return result

    def isPalindrome(self, s):
        if not s:
            return False
        # reverse compare
        return s == s[::-1]
```

C++

```
class Solution {
public:
    /**
     * @param s input string
     * @return the longest palindromic substring
     */
    string longestPalindrome(string& s) {
        string result;
        if (s.empty()) return s;

        int n = s.size();
        int longest = 0, left = 0, right = 0;
        for (int i = 0; i < n; ++i) {
            for (int j = i + 1; j <= n; ++j) {
                string substr = s.substr(i, j - i);
                if (isPalindrome(substr) && substr.size() > longest) {
                    longest = j - i;
                    left = i;
                    right = j;
                }
            }
        }

        result = s.substr(left, right - left);
        return result;
    }

private:
    bool isPalindrome(string &s) {
        int n = s.size();
        for (int i = 0; i < n; ++i) {
            if (s[i] != s[n - i - 1]) return false;
        }
        return true;
    }
};
```

Java

```
public class Solution {
    /**
     * @param s input string
     * @return the longest palindromic substring
     */
    public String longestPalindrome(String s) {
        String result = new String();
        if (s == null || s.isEmpty()) return result;

        int n = s.length();
        int longest = 0, left = 0, right = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j <= n; j++) {
                String substr = s.substring(i, j);
```

```
                if (isPalindrome(substr) && substr.length() > longest) {
                    longest = substr.length();
                    left = i;
                    right = j;
                }
            }
        }

        result = s.substring(left, right);
        return result;
    }

    private boolean isPalindrome(String s) {
        if (s == null || s.isEmpty()) return false;

        int n = s.length();
        for (int i = 0; i < n; i++) {
            if (s.charAt(i) != s.charAt(n - i - 1)) return false;
        }

        return true;
    }
}
```

`left` , `right`

$O(C_n^2) = O(n^2)$, $O(n)$, $O(n^3)$. TLE. `substr` $O(n)$.

2 - (dynamic programming)

"bab""cbabc" `s` `n` $(n \times n)$bool `P` `P[i, j], i<= j` $[s_i, ..., s_j]$
`P[i, j] = P[i+1, j-1] AND s[i] == s[j]`

`P[i, i] = true`

`P[i, i+1] = (s[i] == s[i+1])`

```
string longestPalindrome(string s) {
    int n = s.length();
    int maxBegin = 0;
    int maxLen = 1;
    bool table[1000][1000] = {false};

    for (int i = 0; i < n; i++) {
        table[i][i] = true;
```

```
    }

    for (int i = 0; i < n-1; i++) {
            if (s[i] == s[i+1]) {
            table[i][i+1] = true;
            maxBegin = i;
            maxLen = 2;
        }
    }
    for (int len = 3; len <= n; len++) {
        for (int i = 0; i < n-len+1; i++) {
            int j = i+len-1;
            if (s[i] == s[j] && table[i+1][j-1]) {
                table[i][j] = true;
                maxBegin = i;
                maxLen = len;
            }
        }
    }
    return s.substr(maxBegin, maxLen);
}
```

$O(n^2)O(n^2)$

3 - Manacher's Algorithm

Reference

- Longest Palindromic Substring Part I | LeetCode
- Longest Palindromic Substring Part II | LeetCode

Space Replacement

Source

- lintcode: (212) Space Replacement

```
Write a method to replace all spaces in a string with %20.
The string is given in a characters array, you can assume it has enough space
for replacement and you are given the true length of the string.

Example
Given "Mr John Smith", length = 13.

The string after replacement should be "Mr%20John%20Smith".

Note
If you are using Java or Pythonplease use characters array instead of string.

Challenge
Do it in-place.
```

`%20` overflow `%20` ——

Java

```java
public class Solution {
    /**
     * @param string: An array of Char
     * @param length: The true length of the string
     * @return: The true length of new string
     */
    public int replaceBlank(char[] string, int length) {
        if (string == null) return 0;

        int space = 0;
        for (char c : string) {
            if (c == ' ') space++;
        }

        int r = length + 2 * space - 1;
        for (int i = length - 1; i >= 0; i--) {
            if (string[i] != ' ') {
                string[r] = string[i];
                r--;
            } else {
                string[r--] = '0';
```

```
                string[r--] = '2';
                string[r--] = '%';
            }
        }

        return length + 2 * space;
    }
}
```

```
class Solution {
public:
    /**
     * @param string: An array of Char
     * @param length: The true length of the string
     * @return: The true length of new string
     */
    int replaceBlank(char string[], int length) {
        int space = 0;
        for (int i = 0; i < length; i++) {
            if (string[i] == ' ') space++;
        }

        int r = length + 2 * space - 1;
        for (int i = length - 1; i >= 0; i--) {
            if (string[i] != ' ') {
                string[r] = string[i];
                r--;
            } else {
                string[r--] = '0';
                string[r--] = '2';
                string[r--] = '%';
            }
        }

        return length + 2 * space;
    }
};
```

$O(n)$, `r` $O(1)$.

Integer Array -

Remove Element

Source

- leetcode: Remove Element | LeetCode OJ
- lintcode: (172) Remove Element

```
Given an array and a value, remove all occurrences of that value in place and return the

The order of elements can be changed, and the elements after the new length don't matter.

Example
Given an array [0,4,4,0,0,2,4,4], value=4

return 4 and front four elements of the array is [0,0,0,2]
```

1 -

lintcode C++vector

C++

```
class Solution {
public:
    /**
     *@param A: A list of integers
     *@param elem: An integer
     *@return: The new length after remove
     */
    int removeElement(vector<int> &A, int elem) {
        for (vector<int>::iterator iter = A.begin(); iter < A.end(); ++iter) {
            if (*iter == elem) {
                iter = A.erase(iter);
                --iter;
            }
        }

        return A.size();
    }
};
```

`--iter` , `for` `iter` `A.erase()` `while`

vectorerase$O(n)$ $O(n^2)$ 2

2 -

——

C++

```
class Solution {
public:
    int removeElement(int A[], int n, int elem) {
        for (int i = 0; i < n; ++i) {
            if (A[i] == elem) {
                A[i] = A[n - 1];
                --i;
                --n;
            }
        }

        return n;
    }
};
```

`A[i] == elem`(n) `i` `n`1 `n`

$O(n)$

Reference

- Remove Element |

First Missing Positive

Source

- leetcode: First Missing Positive | LeetCode OJ
- lintcode: (189) First Missing Positive

```
Given an unsorted integer array, find the first missing positive integer.

Example
Given [1,2,0] return 3, and [3,4,-1,1] return 2.

Challenge
Your algorithm should run in O(n) time and uses constant space.
```

$O(n \log n)$,

$O(n)$ $O(n)$

12()

$A[i] = x$, x - $A[x-1] = x$, $A[x-1]$ $A[i]$.

$f[i] = i + 1$, 1

C++

```
class Solution {
public:
    /**
     * @param A: a vector of integers
     * @return: an integer
     */
    int firstMissingPositive(vector<int> A) {
        const int size = A.size();

        for (int i = 0; i < size; ++i) {
            while (0 < A[i]  && A[i] <= size &&
                   (A[i] != i + 1) && (A[i] != A[A[i] - 1])) {
                int temp = A[A[i] - 1];
                A[A[i] - 1] = A[i];
                A[i] = temp;
            }
        }

        for (int i = 0; i < size; ++i) {
            if (A[i] != i + 1) {
```

```
                return i + 1;
            }
        }

        return size + 1;
    }
};
```

1. `A[i]` ...
2. `A[i] \leq size`
3. `A[i] != i + 1`,
4. `A[i] != A[A[i] - 1]`,

`while` `i`

```
int temp = A[i];
A[i] = A[A[i] - 1];
A[A[i] - 1] = temp;
```

bug :(`A[i]`

11

`while` $O(n)$. $O(n)$, `temp` $O(1)$.

Reference

- Find First Missing Positive | N00tc0d3r
- LeetCode: First Missing Positive - Yu's Garden -
- First Missing Positive |

2 Sum

Source

- leetcode: Two Sum | LeetCode OJ
- lintcode: (56) 2 Sum

```
Given an array of integers, find two numbers such that they add up to a specific target n

The function twoSum should return indices of the two numbers
such that they add up to the target, where index1 must be less than index2.
Please note that your returned answers (both index1 and index2) are not zero-based.

You may assume that each input would have exactly one solution.

Input: numbers={2, 7, 11, 15}, target=9
Output: index1=1, index2=2
```

1 -

`target` , `target` too naive... `target` $O(n^2)$, `target` ——

$x_i + x_j = target$, $x_i = target - x_j$, i j ...

() x_j, `target`

C++

```
class Solution {
public:
    /*
     * @param numbers : An array of Integer
     * @param target : target = numbers[index1] + numbers[index2]
     * @return : [index1+1, index2+1] (index1 < index2)
     */
    vector<int> twoSum(vector<int> &nums, int target) {
        vector<int> result;
        const int length = nums.size();
        if (0 == length) {
            return result;
        }

        // first value, second index
        unordered_map<int, int> hash(length);
        for (int i = 0; i != length; ++i) {
            if (hash.find(target - nums[i]) != hash.end()) {
                result.push_back(hash[target - nums[i]]);
                result.push_back(i + 1);
```

```
                return result;
            } else {
                hash[nums[i]] = i + 1;
            }
        }

        return result;
    }
};
```

1.
2. C++ 11 `unordered_map`
3.

$O(n),$ $O(n).$

Python

```
class Solution:
    """
    @param numbers : An array of Integer
    @param target : target = numbers[index1] + numbers[index2]
    @return : [index1 + 1, index2 + 1] (index1 < index2)
    """
    def twoSum(self, numbers, target):
        hashdict = {}
        for i, item in enumerate(numbers):
            if (target - item) in hashdict:
                return (hashdict[target - item] + 1, i + 1)
            hashdict[item] = i

        return (-1, -1)
```

Python `dict` enumerate list, tuple `(-1, -1)`.

2 -

C++

```
class Solution {
public:
```

```
    /*
     * @param numbers : An array of Integer
     * @param target : target = numbers[index1] + numbers[index2]
     * @return : [index1+1, index2+1] (index1 < index2)
     */
    vector<int> twoSum(vector<int> &nums, int target) {
        vector<int> result;
        const int length = nums.size();
        if (0 == length) {
            return result;
        }

        // first num, second is index
        vector<pair<int, int> > num_index(length);
        // map num value and index
        for (int i = 0; i != length; ++i) {
            num_index[i].first = nums[i];
            num_index[i].second = i + 1;
        }

        sort(num_index.begin(), num_index.end());
        int start = 0, end = length - 1;
        while (start < end) {
            if (num_index[start].first + num_index[end].first > target) {
                --end;
            } else if(num_index[start].first + num_index[end].first == target) {
                int min_index = min(num_index[start].second, num_index[end].second);
                int max_index = max(num_index[start].second, num_index[end].second);
                result.push_back(min_index);
                result.push_back(max_index);
                return result;
            } else {
                ++start;
            }
        }

        return result;
    }
};
```

1.
2. `length` `nums.size()`
3. `pair`
4.
5.

`pair` $O(n)$, $O(n)$. $O(n \log n)$, $O(n)$.

> lintcode $O(n \log n)$ $O(1)$, $O(n)$ $O(1)$ $O(n)$ $O(n$

3 Sum

Source

- leetcode: 3Sum | LeetCode OJ
- lintcode: (57) 3 Sum

```
Given an array S of n integers, are there elements a, b, c in S such that a + b + c = 0?
Find all unique triplets in the array which gives the sum of zero.

Example
For example, given array S = {-1 0 1 2 -1 -4}, A solution set is:

(-1, 0, 1)
(-1, -1, 2)
Note
Elements in a triplet (a,b,c) must be in non-descending order. (ie, a ≤ b ≤ c)

The solution set must not contain duplicate triplets.
```

1 - + + 2 Sum

2 Sum, 3 Sum 2 Sum 1 Sum + 1 Sum 3 Sum 1 Sum + 2 Sum 2 Sum

Python

```python
class Solution:
    """
    @param numbersbers : Give an array numbersbers of n integer
    @return : Find all unique triplets in the array which gives the sum of zero.
    """
    def threeSum(self, numbers):
        triplets = []
        length = len(numbers)
        if length < 3:
            return triplets

        numbers.sort()
        for i in xrange(length):
            target = 0 - numbers[i]
            # 2 Sum
            hashmap = {}
            for j in xrange(i + 1, length):
                item_j = numbers[j]
                if (target - item_j) in hashmap:
                    triplet = [numbers[i], target - item_j, item_j]
                    if triplet not in triplets:
                        triplets.append(triplet)
```

```
                else:
                    hashmap[item_j] = j

        return triplets
```

1. 3
2.
3. 2 Sum
4.

2 Sum

$O(n \log n)$, `for` $O(n^2)()$ $O(n)$.

leetcode 500 + ms,

C++

```
class Solution {
public:
    vector<vector<int> > threeSum(vector<int> &num)
    {
        vector<vector<int> > result;
        if (num.size() < 3) return result;

        int ans = 0;

        sort(num.begin(), num.end());

        for (int i = 0;i < num.size() - 2; ++i)
        {
            if (i > 0 && num[i] == num[i - 1])
                continue;
            int j = i + 1;
            int k = num.size() - 1;

            while (j < k)
            {
                ans = num[i] + num[j] + num[k];

                if (ans == 0)
                {
                    result.push_back({num[i], num[j], num[k]});
                    ++j;
                    while (j < num.size() && num[j] == num[j - 1])
                        ++j;
                    --k;
                    while (k >= 0 && num[k] == num[k + 1])
                        --k;
```

```
                }
                else if (ans > 0)
                    --k;
                else
                    ++j;
            }
        }

        return result;
    }
};
```

pythonhash map

```
S = {-1 0 1 2 -1 -4}

S = {-4 -1 -1 0 1 2}
      ↑  ↑        ↑
      i  j        k
         →        ←
ijkS[i]+S[j]+S[k]=ans0jS[j]kS[k]
ans>0S[k]kans<0S[j]jans==0
```

in,jkn-i $O(n^2)$ 52ms

Reference

- 3Sum |
- A simply Python version based on 2sum - O(n^2) - Leetcode Discuss

Merge Sorted Array

Source

- leetcode: Merge Sorted Array | LeetCode OJ
- lintcode: (6) Merge Sorted Array

```
Given two sorted integer arrays A and B, merge B into A as one sorted array.

Example
A = [1, 2, 3, empty, empty], B = [4, 5]

After merge, A will be filled as [1, 2, 3, 4, 5]

Note
You may assume that A has enough space (size that is greater or equal to m + n)
to hold additional elements from B.
The number of elements initialized in A and B are m and n respectively.
```

in-place $O(n^2)$ A A B `m == 0` `n == 0` B A A

Python

```python
class Solution:
    """
    @param A: sorted integer array A which has m elements,
              but size of A is m+n
    @param B: sorted integer array B which has n elements
    @return: void
    """
    def mergeSortedArray(self, A, m, B, n):
        if B is None:
            return A

        index = m + n - 1
        while m > 0 and n > 0:
            if A[m - 1] > B[n - 1]:
                A[index] = A[m - 1]
                m -= 1
            else:
                A[index] = B[n - 1]
                n -= 1
            index -= 1

        # B has elements left
        while n > 0:
```

```
            A[index] = B[n - 1]
            n -= 1
            index -= 1
```

C++

```cpp
class Solution {
public:
    /**
     * @param A: sorted integer array A which has m elements,
     *           but size of A is m+n
     * @param B: sorted integer array B which has n elements
     * @return: void
     */
    void mergeSortedArray(int A[], int m, int B[], int n) {
        int index = m + n - 1;
        while (m > 0 && n > 0) {
            if (A[m - 1] > B[n - 1]) {
                A[index] = A[m - 1];
                --m;
            } else {
                A[index] = B[n - 1];
                --n;
            }
            --index;
        }

        // B has elements left
        while (n > 0) {
            A[index] = B[n - 1];
            --n;
            --index;
        }
    }
};
```

Java

```java
class Solution {
    /**
     * @param A: sorted integer array A which has m elements,
     *           but size of A is m+n
     * @param B: sorted integer array B which has n elements
     * @return: void
     */
    public void mergeSortedArray(int[] A, int m, int[] B, int n) {
        if (A == null || B == null) return;

        int index = m + n - 1;
        while (m > 0 && n > 0) {
            if (A[m - 1] > B[n - 1]) {
                A[index] = A[m - 1];
                m--;
            } else {
```

```
                A[index] = B[n - 1];
                n--;
            }
            index--;
        }

        // B has elements left
        while (n > 0) {
            A[index] = B[n - 1];
            n--;
            index--;
        }
    }
}
```

while (conditional AND)

$O(n)$. $O(1)$.

Merge Sorted Array II

Source

- lintcode: (64) Merge Sorted Array II

```
Merge two given sorted integer array A and B into a new sorted integer array.

Example
A=[1,2,3,4]

B=[2,4,5,6]

return [1,2,2,3,4,4,5,6]

Challenge
How can you optimize your algorithm
if one array is very large and the other is very small?
```

in-place,

Python

```python
class Solution:
    #@param A and B: sorted integer array A and B.
    #@return: A new sorted integer array
    def mergeSortedArray(self, A, B):
        if A is None or len(A) == 0:
            return B
        if B is None or len(B) == 0:
            return A

        C = []
        aLen, bLen = len(A), len(B)
        i, j = 0, 0
        while i < aLen and j < bLen:
            if A[i] < B[j]:
                C.append(A[i])
                i += 1
            else:
                C.append(B[j])
                j += 1

        # A has elements left
        while i < aLen:
            C.append(A[i])
            i += 1
```

```
        # B has elements left
        while j < bLen:
            C.append(B[j])
            j += 1

        return C
```

C++

```
class Solution {
public:
    /**
     * @param A and B: sorted integer array A and B.
     * @return: A new sorted integer array
     */
    vector<int> mergeSortedArray(vector<int> &A, vector<int> &B) {
        if (A.empty()) return B;
        if (B.empty()) return A;

        int aLen = A.size(), bLen = B.size();
        vector<int> C;
        int i = 0, j = 0;
        while (i < aLen && j < bLen) {
            if (A[i] < B[j]) {
                C.push_back(A[i]);
                ++i;
            } else {
                C.push_back(B[j]);
                ++j;
            }
        }

        // A has elements left
        while (i < aLen) {
            C.push_back(A[i]);
            ++i;
        }

        // B has elements left
        while (j < bLen) {
            C.push_back(B[j]);
            ++j;
        }

        return C;
    }
};
```

Java

```
class Solution {
    /**
     * @param A and B: sorted integer array A and B.
     * @return: A new sorted integer array
```

```
     */
    public ArrayList<Integer> mergeSortedArray(ArrayList<Integer> A, ArrayList<Integer> B
        if (A == null || A.isEmpty()) return B;
        if (B == null || B.isEmpty()) return A;

        ArrayList<Integer> C = new ArrayList<Integer>();
        int aLen = A.size(), bLen = B.size();
        int i = 0, j = 0;
        while (i < aLen && j < bLen) {
            if (A.get(i) < B.get(j)) {
                C.add(A.get(i));
                i++;
            } else {
                C.add(B.get(j));
                j++;
            }
        }

        // A has elements left
        while (i < aLen) {
            C.add(A.get(i));
            i++;
        }

        // B has elements left
        while (j < bLen) {
            C.add(B.get(j));
            j++;
        }

        return C;
    }
}
```

A, B $O(n)$, $O(1)$.

Challenge

Merge

Search -

-
- 1find the first/last position of...2 $O(\log n)$
-

Binary Search -

Source

- lintcode: lintcode - (14) Binary Search

```
Binary search is a famous question in algorithm.

For a given sorted array (ascending order) and a target number, find the first index of t

If the target number does not exist in the array, return -1.

Example
If the array is [1, 2, 3, 3, 4, 5, 10], for given target 3, return 2.

Challenge
If the count of numbers is bigger than MAXINT, can your code work properly?
```

Java

```java
/**
 * fork
 * http://www.jiuzhang.com//solutions/binary-search/
 */
class Solution {
    /**
     * @param nums: The integer array.
     * @param target: Target to find.
     * @return: The first position of target. Position starts from 0.
     */
    public int binarySearch(int[] nums, int target) {
        if (nums == null || nums.length == 0) {
            return -1;
        }

        int start = 0;
        int end = nums.length - 1;
        int mid;
        while (start + 1 < end) {
            mid = start + (end - start) / 2; // avoid overflow when (end + start)
            if (target < nums[mid]) {
                end = mid;
            } else if (target > nums[mid]) {
                start = mid;
            } else {
```

```
                end = mid;
            }
        }

        if (nums[start] == target) {
            return start;
        }
        if (nums[end] == target) {
            return end;
        }

        return -1;
    }
}
```

1. 0
2. `start, end, mid` mid
3.
4. while `start + 1 < end` `start <= end` `start == end`
5. targetstartend—— first position or last position
6. `end = mid`
7. while `start + 1 < end` mid `+1` `-1`

C++

```
class Solution {
public:
    /**
     * @param nums: The integer array.
     * @param target: Target number to find.
     * @return: The first position of target. Position starts from 0.
     */
    int binarySearch(vector<int> &nums, int target) {
        if( nums.size() == 0 ) return -1;

        int lo = 0, hi = nums.size();
        while(lo < hi){
            int mi = lo + (hi - lo)/2;
            if(nums[mi] < target)
                lo = mi + 1;
            else
                hi = mi;
        }

        if(nums[lo] == target) return lo;
        return -1;
    }
};
```

C/C++index[lo, hi)lohi `for(i = lo; i < hi; i++)` `int length = hi - lo;`
STLiteratorend()iteratorC++

1. `lo < hi` `lo` `hi` `lo == hi`
2. `lo` `nums[mi]` `lo` `mi + 1` `lo` `target` `hi` `lo` **`target`** `target`-1

Search Insert Position

Source

- lintcode: (60) Search Insert Position

```
Given a sorted array and a target value, return the index if the target is found. If not,

You may assume no duplicates in the array.

Example
[1,3,5,6], 5 → 2
[1,3,5,6], 2 → 1
[1,3,5,6], 7 → 4
[1,3,5,6], 0 → 0
```

`find the first/last position of...`

Java

```java
public class Solution {
    /**
     * param A : an integer sorted array
     * param target :  an integer to be inserted
     * return : an integer
     */
    public int searchInsert(int[] A, int target) {
        if (A == null) {
            return -1;
        }
        if (A.length == 0) {
            return 0;
        }

        int start = 0, end = A.length - 1;
        int mid;

        while (start + 1 < end) {
            mid = start + (end - start) / 2;
            if (A[mid] == target) {
                return mid; // no duplicates, if not `end = target;`
            } else if (A[mid] < target) {
                start = mid;
            } else {
                end = mid;
            }
        }
```

```
        if (A[start] >= target) {
            return start;
        } else if (A[end] >= target) {
            return end; // in most cases
        } else {
            return end + 1; // A[end] < target;
        }
    }
}
```

, [1,3,5,6], 7 → 4else `return end + 1;`

C++

```
class Solution {
    /**
     * param A : an integer sorted array
     * param target :  an integer to be inserted
     * return : an integer
     */
public:
    int searchInsert(vector<int> &A, int target) {
        int N = A.size();
        if (N == 0) return 0;
        if (A[N-1] < target) return N;
        int lo = 0, hi = N;
        while (lo < hi) {
            int mi = lo + (hi - lo)/2;
            if (A[mi] < target)
                lo = mi + 1;
            else
                hi = mi;
        }
        return lo;
    }
};
```

lintcode - (14) Binary SearchC++[lo, hi) `target` `A[m]` `target` `lo` `hi` `target` `ho` `ho` `target`

Search for a Range

Source

- lintcode: (61) Search for a Range

```
Given a sorted array of integers, find the starting and ending position of a given target

Your algorithm's runtime complexity must be in the order of O(log n).

If the target is not found in the array, return [-1, -1].

Example
Given [5, 7, 7, 8, 8, 10] and target value 8,
return [3, 4].
```

Search for a range first & last position `(target == nums[mid] end = mid start = mid`

Java

```java
/**
 * fork
 * http://www.jiuzhang.com/solutions/search-for-a-range/
 */
public class Solution {
    /**
     *@param A : an integer sorted array
     *@param target :  an integer to be inserted
     *return : a list of length 2, [index1, index2]
     */
    public ArrayList<Integer> searchRange(ArrayList<Integer> A, int target) {
        ArrayList<Integer> result = new ArrayList<Integer>();
        int start, end, mid;
        result.add(-1);
        result.add(-1);

        if (A == null || A.size() == 0) {
            return result;
        }

        // search for left bound
        start = 0;
        end = A.size() - 1;
        while (start + 1 < end) {
            mid = start + (end - start) / 2;
            if (A.get(mid) == target) {
```

```
                end = mid; // set end = mid to find the minimum mid
            } else if (A.get(mid) > target) {
                end = mid;
            } else {
                start = mid;
            }
        }
        if (A.get(start) == target) {
            result.set(0, start);
        } else if (A.get(end) == target) {
            result.set(0, end);
        } else {
            return result;
        }

        // search for right bound
        start = 0;
        end = A.size() - 1;
        while (start + 1 < end) {
            mid = start + (end - start) / 2;
            if (A.get(mid) == target) {
                start = mid; // set start = mid to find the maximum mid
            } else if (A.get(mid) > target) {
                end = mid;
            } else {
                start = mid;
            }
        }
        if (A.get(end) == target) {
            result.set(1, end);
        } else if (A.get(start) == target) {
            result.set(1, start);
        } else {
            return result;
        }

        return result;
        // write your code here
    }
}
```

1. 0
2. `start, end, mid` mid
3.
4. while `start + 1 < end` `start <= end` `start == end`
5. `A.get(start) == target` `A.get(end) == target` targetstartendendstartstart.
6. `A.get(end) == target` `A.get(start) == target`
7. `A.get(mid) == target` first postion `end = mid` last position `start = mid`
8. `start, end`

C++

```
class Solution {
    /**
     *@param A : an integer sorted array
     *@param target :  an integer to be inserted
     *return : a list of length 2, [index1, index2]
     */
public:
    vector<int> searchRange(vector<int> &A, int target) {
        // good, fail are the result
        // When found, returns good, otherwise returns fail
        int N = A.size();
        vector<int> fail = {-1, -1};
        if(N == 0)
            return fail;
        vector<int> good;

        // search for starting position
        int lo = 0, hi = N;
        while(lo < hi){
            int m = lo + (hi- lo)/2;
            if(A[m] < target)
                lo = m + 1;
            else
                hi = m;
        }

        if(A[lo] != target)
            return fail;

        good.push_back(lo);

        // search for ending position
        lo = 0; hi = N;
        while(lo < hi){
            int m = lo + (hi - lo)/2;
            if(target < A[m])
                hi = m;
            else
                lo = m + 1;
        }
        good.push_back(lo - 1);

        return good;
    }
};
```

"target""target"[lo, hi)"target" `lo` 11

Sqrt x

Source

- leetcode: Sqrt(x) | LeetCode OJ
- lintcode: (141) Sqrt(x)

-

x, k, $1 \leq k \leq x$, k

Python

```
class Solution:
    # @param {integer} x
    # @return {integer}
    def mySqrt(self, x):
        if x < 0:
            return -1
        elif x == 0:
            return 0

        start, end = 1, x
        while start + 1 < end:
            mid = start + (end - start) / 2
            if mid**2 == x:
                return mid
            elif mid**2 > x:
                end = mid
            else:
                start = mid

        return start
```

1. 0
2. `start < end` , 1
3. `start` .

start, end, mid? `while start + 1 < end` `start` `end` `end == 1 || end ==2`

1 `start` `end` x k, $start \leq k \leq end$ $mid^2 == x$ st

`start`

C++

```
class Solution{
public:
    int mySqrt(int x) {
        if(x <= 1) return x;
        int lo = 2, hi = x;
        while(lo < hi){
            int m = lo + (hi - lo)/2;
            int q = x/m;
            if(q == m and x % m == 0)
                return m;
            else if(q < m)
                hi = m;
            else
                lo = m + 1;
        }

        return lo - 1;
    }
};
```

"target x^2"[lo, hi)[Search for a range] `m * m == x` `x / m == m` `x * x` `INT_MAX`

$O(\log n)$, `start` , `end` , `mid` $O(1)$.

sqrt() --

Linked List -

Remove Duplicates from Sorted List

Source

- leetcode: Remove Duplicates from Sorted List | LeetCode OJ
- lintcode: (112) Remove Duplicates from Sorted List

```
Given a sorted linked list,
delete all duplicates such that each element appear only once.

Example
Given 1->1->2, return 1->2.
Given 1->1->2->3->3, return 1->2->3.
```

`next` `next`,

Python

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    # @param {ListNode} head
    # @return {ListNode}
    def deleteDuplicates(self, head):
        if head is None:
            return None

        node = head
        while node.next is not None:
            if node.val == node.next.val:
                node.next = node.next.next
            else:
                node = node.next

        return head
```

C++

```cpp
/**
 * Definition of ListNode
 * class ListNode {
```

```
 * public:
 *     int val;
 *     ListNode *next;
 *     ListNode(int val) {
 *         this->val = val;
 *         this->next = NULL;
 *     }
 * }
 */
class Solution {
public:
    /**
     * @param head: The first node of linked list.
     * @return: head node
     */
    ListNode *deleteDuplicates(ListNode *head) {
        if (head == NULL) {
            return NULL;
        }

        ListNode *node = head;
        while (node->next != NULL) {
            if (node->val == node->next->val) {
                ListNode *temp = node->next;
                node->next = node->next->next;
                delete temp;
            } else {
                node = node->next;
            }
        }

        return head;
    }
};
```

Java

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
public class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if (head == null) return null;

        ListNode node = head;
        while (node.next != null) {
            if (node.val == node.next.val) {
                node.next = node.next.next;
            } else {
                node = node.next;
            }
        }
```

```
            return head;
        }
    }
```

1. headNULL
2. `node->val == node->next->val` `node->next` (C/C++)
3. `else`

`while` `node != null && node->next != null`, `head`

$O(n),$ $O(1).$

Reference

- Remove Duplicates from Sorted List |

Remove Duplicates from Sorted List II

Source

- leetcode: Remove Duplicates from Sorted List II | LeetCode OJ
- lintcode: (113) Remove Duplicates from Sorted List II

```
Given a sorted linked list, delete all nodes that have duplicate numbers,
leaving only distinct numbers from the original list.

Example
Given 1->2->3->3->4->4->5, return 1->2->5.
Given 1->1->1->2->3, return 2->3.
```

()if **——dummy node.**

```
ListNode *dummy = new ListNode(0);
dummy->next = head;
ListNode *node = dummy;
```

`dummy` nextheaddummyhead `A->B->C` BACAnextCheadnodenode

`dummy` `node->val == node->next->val`

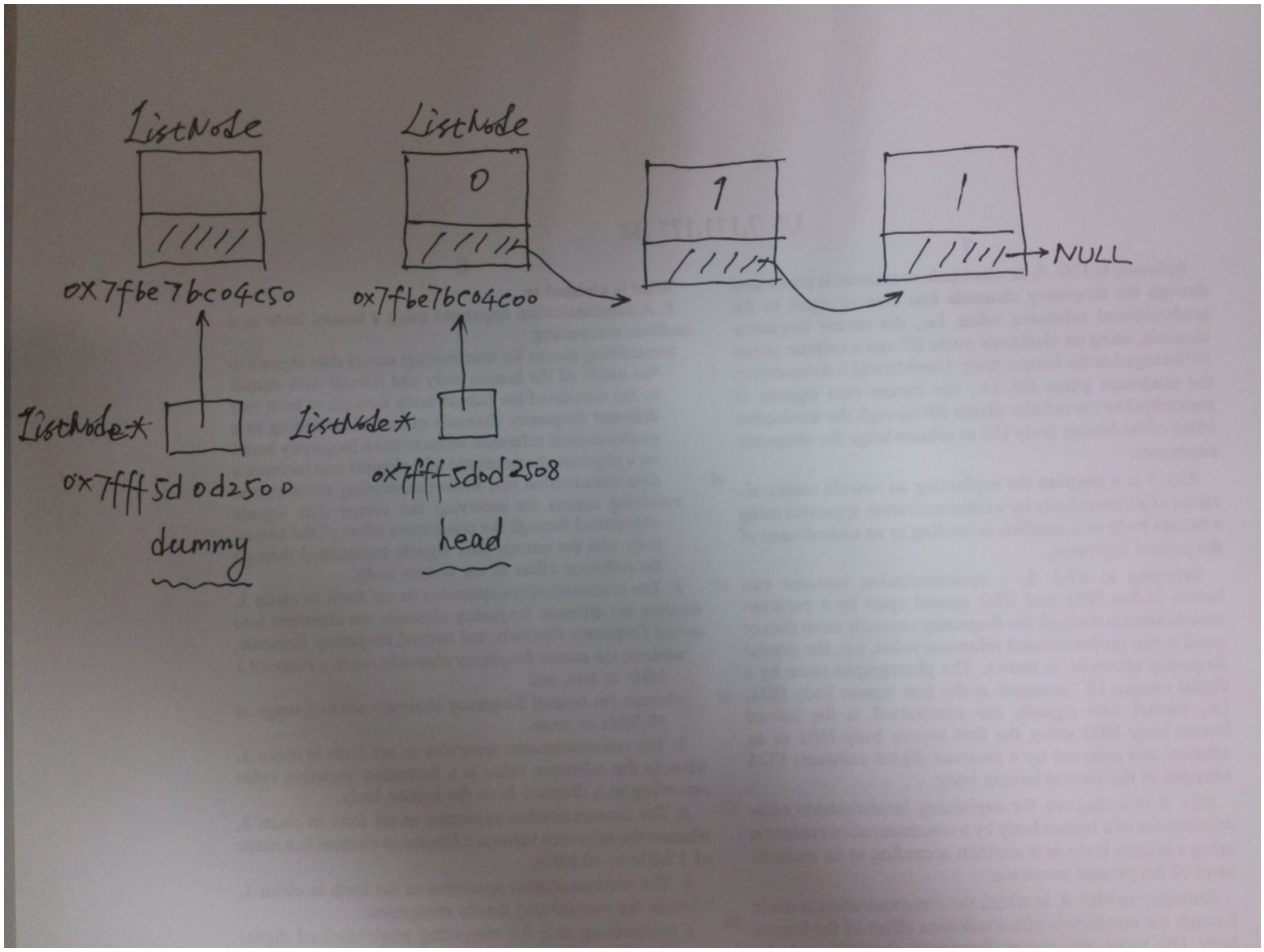
1. `next`
2. val `node->next` `node->next->next`

C++ - Wrong

```
/**
 * Definition of ListNode
 * class ListNode {
 * public:
 *     int val;
 *     ListNode *next;
 *     ListNode(int val) {
 *         this->val = val;
 *         this->next = NULL;
 *     }
 * }
 */
class Solution{
public:
```

```
    /**
     * @param head: The first node of linked list.
     * @return: head node
     */
    ListNode * deleteDuplicates(ListNode *head) {
        if (head == NULL || head->next == NULL) {
            return NULL;
        }

        ListNode *dummy;
        dummy->next = head;
        ListNode *node = dummy;

        while (node->next != NULL && node->next->next != NULL) {
            if (node->next->val == node->next->next->val) {
                int val = node->next->val;
                while (node->next != NULL && val == node->next->val) {
                    ListNode *temp = node->next;
                    node->next = node->next->next;
                    delete temp;
                }
            } else {
                node->next = node->next->next;
            }
        }

        return dummy->next;
    }
};
```

1. dummyclass `new`
2. else `node->next = node->next->next;` `dummy-next` `dummy-next` `node = node->next;` nodenode->next



ListNode `dummy` ox7fff5d0d25000x7fbe7bc04c50. `head` ox7fff5d0d25080x7fbe7bc04c00.

Python

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    # @param {ListNode} head
    # @return {ListNode}
    def deleteDuplicates(self, head):
        if head is None:
            return None

        dummy = ListNode(0)
        dummy.next = head
        node = dummy
        while node.next is not None and node.next.next is not None:
            if node.next.val == node.next.next.val:
                val_prev = node.next.val
                while node.next is not None and node.next.val == val_prev:
                    node.next = node.next.next
```

```
            else:
                node = node.next

        return dummy.next
```

C++

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        if (head == NULL) return NULL;

        ListNode *dummy = new ListNode(0);
        dummy->next = head;
        ListNode *node = dummy;
        while (node->next != NULL && node->next->next != NULL) {
            if (node->next->val == node->next->next->val) {
                int val_prev = node->next->val;
                // remove ListNode node->next
                while (node->next != NULL && val_prev == node->next->val) {
                    ListNode *temp = node->next;
                    node->next = node->next->next;
                    delete temp;
                }
            } else {
                node = node->next;
            }
        }

        return dummy->next;
    }
};
```

Java

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
public class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if (head == null) return null;
```

```
        ListNode dummy = new ListNode(0);
        dummy.next = head;
        ListNode node = dummy;
        while(node.next != null && node.next.next != null) {
            if (node.next.val == node.next.next.val) {
                int val_prev = node.next.val;
                while (node.next != null && node.next.val == val_prev) {
                    node.next = node.next.next;
                }
            } else {
                node = node.next;
            }
        }

        return dummy.next;
    }
}
```

1. head NULL NULL
2. newdummy `dummy->next`
3. nodedummy
4. valwhile
5. `dummy->next`

Python `is not None`

(node.next node.next.next) $O(2n)$. dummy $O(1)$.

Reference

- Remove Duplicates from Sorted List II |

Reverse Linked List

Source

- leetcode: Reverse Linked List | LeetCode OJ
- lintcode: (35) Reverse Linked List

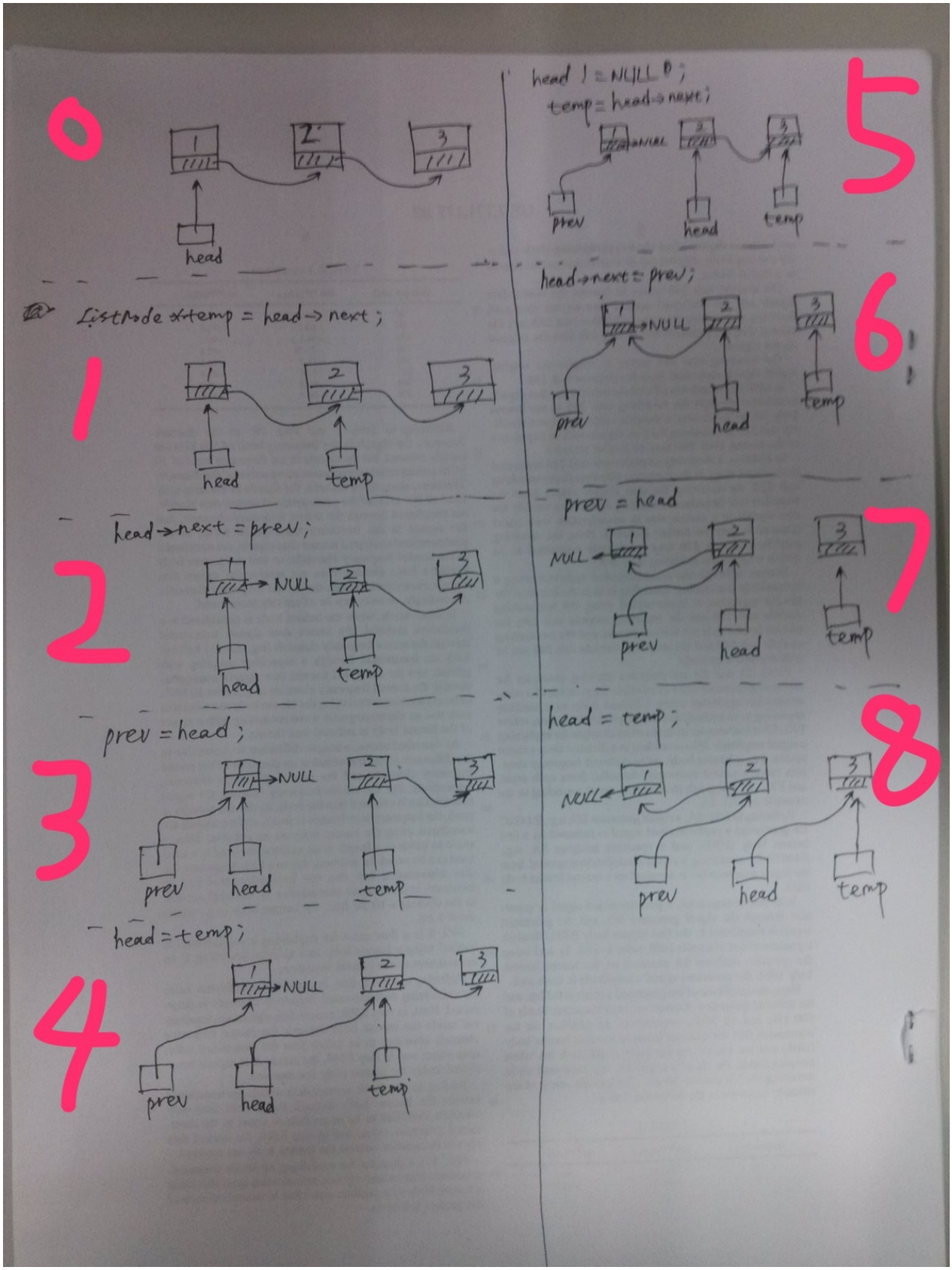
```
Reverse a linked list.

Example
For linked list 1->2->3, the reversed linked list is 3->2->1

Challenge
Reverse it in-place and in one-pass
```

1 -

`1->2->3` `3->2->1` 112 `1->2` `2->1` 12

```
temp = head->next;
head->next = prev;
prev = head;
head = temp;
```

`prev` `head`,



1. head
2. headprev
3. prevhead
4. head

Python

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    # @param {ListNode} head
    # @return {ListNode}
    def reverseList(self, head):
        prev = None
        curr = head
        while curr is not None:
            temp = curr.next
            curr.next = prev
            prev = curr
            curr = temp
        # fix head
        head = prev

        return head
```

C++

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* reverse(ListNode* head) {
        ListNode *prev = NULL;
        ListNode *curr = head;
        while (curr != NULL) {
            ListNode *temp = curr->next;
            curr->next = prev;
            prev = curr;
            curr = temp;
        }
        // fix head
        head = prev;

        return head;
    }
};
```

Java

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
public class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode prev = null;
        ListNode curr = head;
        while (curr != null) {
            ListNode temp = curr.next;
            curr.next = prev;
            prev = curr;
            curr = temp;
        }
        // fix head
        head = prev;

        return head;
    }
}
```

prev

$O(n),$ $O(1).$

2 -

1.
2.
3.

()

Python

```python
"""
Definition of ListNode

class ListNode(object):

    def __init__(self, val, next=None):
```

```
        self.val = val
        self.next = next
"""
class Solution:
    """
    @param head: The first node of the linked list.
    @return: You should return the head of the reversed linked list.
                 Reverse it in-place.
    """
    def reverse(self, head):
        # case1: empty list
        if head is None:
            return head
        # case2: only one element list
        if head.next is None:
            return head
        # case3: reverse from the rest after head
        newHead = self.reverse(head.next)
        # reverse between head and head->next
        head.next.next = head
        # unlink list from the rest
        head.next = None

        return newHead
```

C++

```
/**
 * Definition of ListNode
 *
 * class ListNode {
 * public:
 *     int val;
 *     ListNode *next;
 *
 *     ListNode(int val) {
 *         this->val = val;
 *         this->next = NULL;
 *     }
 * }
 */
class Solution {
public:
    /**
     * @param head: The first node of linked list.
     * @return: The new head of reversed linked list.
     */
    ListNode *reverse(ListNode *head) {
        // case1: empty list
        if (head == NULL) return head;
        // case2: only one element list
        if (head->next == NULL) return head;
        // case3: reverse from the rest after head
        ListNode *newHead = reverse(head->next);
        // reverse between head and head->next
        head->next->next = head;
        // unlink list from the rest
```

```
        head->next = NULL;

        return newHead;
    }
};
```

Java

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
public class Solution {
    public ListNode reverse(ListNode head) {
        // case1: empty list
        if (head == null) return head;
        // case2: only one element list
        if (head.next == null) return head;
        // case3: reverse from the rest after head
        ListNode newHead = reverse(head.next);
        // reverse between head and head->next
        head.next.next = head;
        // unlink list from the rest
        head.next = null;

        return newHead;
    }
}
```

case1 case2 case3

$O(n),\quad O(n),()\qquad O(1).$

Reference

- - -
- data structures - Reversing a linked list in Java, recursively - Stack Overflow
- C++ |
- iteratively and recursively Java Solution - Leetcode Discuss

Merge Two Sorted Lists

Source

- lintcode: (165) Merge Two Sorted Lists
- leetcode: Merge Two Sorted Lists | LeetCode OJ

```
Merge two sorted linked lists and return it as a new list.
The new list should be made by splicing together the nodes of the first two lists.

Example
Given 1->3->8->11->15->null, 2->null , return 1->2->3->8->11->15->null
```

`dummy` `next` `next` `next` `NULL`...

`dummy` `dummy` `lastNode` `l1` `l2` `NULL` `while` `lastNode->next`

C++

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* l1, ListNode* l2) {
        ListNode *dummy = new ListNode(0);
        ListNode *lastNode = dummy;
        while ((NULL != l1) && (NULL != l2)) {
            if (l1->val < l2->val) {
                lastNode->next = l1;
                l1 = l1->next;
            } else {
                lastNode->next = l2;
                l2 = l2->next;
            }

            lastNode = lastNode->next;
        }

        // do not forget this line!
        lastNode->next =  (NULL != l1) ? l1 : l2;
```

```
        return dummy->next;
    }
};
```

1. `dummy->next`
2. `dummy` `lastNode` `lastNode` `dummy`
3. l1,l2l1/l2 `lastNode->next` `lastNode`
4. l1/l2while `lastNode->next`
5. `dummy->next`

`lastNode` `dummy->next` `lastNode` `dummy`

> 1. 2. `la`

$O(1)$. `lastNode` $O(l1+l2)$. $O(1)$.

Reference

- Merge Two Sorted Lists |

Maximum Depth of Binary Tree# Binary Tree -

- Binary Tree | Algorithm

Binary Tree Preorder Traversal

Source

- leetcode: Binary Tree Preorder Traversal | LeetCode OJ
- lintcode: (66) Binary Tree Preorder Traversal

```
Given a binary tree, return the preorder traversal of its nodes' values.

Note
Given binary tree {1,#,2,3},

   1
    \
     2
    /
   3

return [1,2,3].

Example
Challenge
Can you do it without recursion?
```

1 -

() `null` vector

(stackoverflow)

Python - Divide and Conquer

```python
"""
Definition of TreeNode:
class TreeNode:
    def __init__(self, val):
        this.val = val
        this.left, this.right = None, None
"""


class Solution:
    """
    @param root: The root of binary tree.
    @return: Preorder in ArrayList which contains node values.
    """
    def preorderTraversal(self, root):
```

```
        if root == None:
            return []
        return [root.val] + self.preorderTraversal(root.left) \
                          + self.preorderTraversal(root.right)
```

C++ - Divide and Conquer

```
/**
 * Definition of TreeNode:
 * class TreeNode {
 * public:
 *     int val;
 *     TreeNode *left, *right;
 *     TreeNode(int val) {
 *         this->val = val;
 *         this->left = this->right = NULL;
 *     }
 * }
 */

class Solution {
public:
    /**
     * @param root: The root of binary tree.
     * @return: Preorder in vector which contains node values.
     */
    vector<int> preorderTraversal(TreeNode *root) {
        vector<int> result;
        if (root != NULL) {
            // Divide ()
            vector<int> left = preorderTraversal(root->left);
            vector<int> right = preorderTraversal(root->right);
            // Merge
            result.push_back(root->val);
            result.insert(result.end(), left.begin(), left.end());
            result.insert(result.end(), right.begin(), right.end());
        }

        return result;
    }
};
```

C++ - Traversal

```
/**
 * Definition of TreeNode:
 * class TreeNode {
 * public:
 *     int val;
 *     TreeNode *left, *right;
 *     TreeNode(int val) {
 *         this->val = val;
 *         this->left = this->right = NULL;
 *     }
```

```
 * }
 */

class Solution {
public:
    /**
     * @param root: The root of binary tree.
     * @return: Preorder in vector which contains node values.
     */
    vector<int> preorderTraversal(TreeNode *root) {
        vector<int> result;
        traverse(root, result);

        return result;
    }

private:
    void traverse(TreeNode *root, vector<int> &ret) {
        if (root != NULL) {
            ret.push_back(root->val);
            traverse(root->left, ret);
            traverse(root->right, ret);
        }
    }
};
```

Java - Divide and Conquer

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
public class Solution {
    public List<Integer> preorderTraversal(TreeNode root) {
        List<Integer> result = new ArrayList<Integer>();
        if (root != null) {
            // Divide
            List<Integer> left = preorderTraversal(root.left);
            List<Integer> right = preorderTraversal(root.right);
            // Merge
            result.add(root.val);
            result.addAll(left);
            result.addAll(right);
        }

        return result;
    }
}
```

`traverse` `vector` C++ vector, vectorvectorpush_back, insert Java `addAll`.

$O(n)$, (stack)

2 -

()(NULL)

Python

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution:
    # @param {TreeNode} root
    # @return {integer[]}
    def preorderTraversal(self, root):
        if root is None:
            return []

        result = []
        s = []
        s.append(root)
        while s:
            root = s.pop()
            result.append(root.val)
            if root.right is not None:
                s.append(root.right)
            if root.left is not None:
                s.append(root.left)

        return result
```

C++

```
/**
 * Definition of TreeNode:
 * class TreeNode {
 * public:
 *     int val;
 *     TreeNode *left, *right;
 *     TreeNode(int val) {
 *         this->val = val;
 *         this->left = this->right = NULL;
 *     }
 * }
```

```
 */

class Solution {
public:
    /**
     * @param root: The root of binary tree.
     * @return: Preorder in vector which contains node values.
     */
    vector<int> preorderTraversal(TreeNode *root) {
        vector<int> result;
        if (root == NULL) return result;

        stack<TreeNode *> s;
        s.push(root);
        while (!s.empty()) {
            TreeNode *node = s.top();
            s.pop();
            result.push_back(node->val);
            if (node->right != NULL) {
                s.push(node->right);
            }
            if (node->left != NULL) {
                s.push(node->left);
            }
        }

        return result;
    }
};
```

Java

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
public class Solution {
    public List<Integer> preorderTraversal(TreeNode root) {
        List<Integer> result = new ArrayList<Integer>();
        if (root == null) return result;

        Stack<TreeNode> s = new Stack<TreeNode>();
        s.push(root);
        while (!s.empty()) {
            TreeNode node = s.pop();
            result.add(node.val);
            if (node.right != null) s.push(node.right);
            if (node.left != null) s.push(node.left);
        }

        return result;
    }
```

```
}
```

1. root
2. root
3. s
4. (pop)
5.
6.

4,5,6

$O(n),$ $O(n).$

Problem Misc

String to Integer

Source

- leetcode: String to Integer (atoi) | LeetCode OJ
- lintcode: (54) String to Integer(atoi)

```
Implement function atoi to convert a string to an integer.

If no valid conversion could be performed, a zero value is returned.

If the correct value is out of the range of representable values,
INT_MAX (2147483647) or INT_MIN (-2147483648) is returned.

Example
"10" => 10

"-1" => -1

"123123123123123" => 2147483647

"1.0" => 1
```

()

Java

```java
public class Solution {
    /**
     * @param str: A string
     * @return An integer
     */
    public int atoi(String str) {
        if (str == null || str.length() == 0) return 0;

        // trim left and right spaces
        String strTrim = str.trim();
        int len = strTrim.length();
        // sign symbol for positive and negative
        int sign = 1;
        // index for iteration
        int i = 0;
        if (strTrim.charAt(i) == '+') {
            i++;
        } else if (strTrim.charAt(i) == '-') {
            sign = -1;
            i++;
        }
```

```
        // store the result as long to avoid overflow
        long result = 0;
        while (i < len) {
            if (strTrim.charAt(i) < '0' || strTrim.charAt(i) > '9') {
                break;
            }
            result = 10 * result + sign * (strTrim.charAt(i) - '0');
            // overflow
            if (result > Integer.MAX_VALUE) {
                return Integer.MAX_VALUE;
            } else if (result < Integer.MIN_VALUE) {
                return Integer.MIN_VALUE;
            }
            i++;
        }

        return (int)result;
    }
}
```

while while long

C++

```
class Solution {
public:
    bool overflow(string str, string help){
        if(str.size() > help.size()) return true;
        else if(str.size() < help.size()) return false;
        for(int i = 0; i < str.size(); i++){
            if(str[i] > help[i]) return true;
            else if(str[i] < help[i]) return false;
        }
        return false;
    }
    int myAtoi(string str) {
        // ans: number, sign: +1 or -1
        int ans = 0;
        int sign = 1;
        int i = 0;
        int N = str.size();

        // eliminate spaces
        while(i < N){
            if(isspace(str[i]))
                i++;
            else
                break;
        }

        // if the whole string contains only spaces, return
        if(i == N) return ans;
```

```
        if(str[i] == '+')
            i++;
        else if(str[i] == '-'){
            sign = -1;
            i++;
        }

        // "help" gets the string of valid numbers
        string help;
        while(i < N){
            if('0' <= str[i] and str[i] <= '9')
                help += str[i++];
            else
                break;
        }

        const string maxINT = "2147483647";
        const string minINT = "2147483648";

        // test whether overflow, test only number parts with both signs

        if(sign == 1){
            if(overflow(help, maxINT)) return INT_MAX;
        }
        else{
            if(overflow(help, minINT)) return INT_MIN;
        }

        for(int j=0; j<help.size(); j++){
            ans = 10 * ans + int(help[j] - '0');
        }

        return ans*sign;
    }
};
```

C++stringCstring to long machine-dependent `INT_MAX` `INT_MIN`

Reference

- String to Integer (atoi) Java/C++/Python