

Two pointers and/or Hash Map	6
√ 283. Move Zeros ++++++	6
Sub Optimal	6
Optimal:	7
One line solution:	7
1. Two Sum +	7
Hash Map, $O(n)$	7
Sort and Two pointers, $O(n \lg n)$	7
15. 3 Sum +++++	7
X 76. Minimum Window Substring ++	8
√ 205. Isomorphic Strings ++	9
Using map[256] trick as hash map:	9
√ 125. Valid Palindrome +++++	9
523. Continuous Subarray Sum +	10
X 80. Remove Duplicates from Sorted Array II +	10
Very short	10
Generic way of allowing K duplicates:	10
√ 383. Ransom Note	11
Explicit Unordered Map	11
Using array size 128	11
√ 554. Brick Wall	11
X 380. Insert Delete GetRandom $O(1)$	12
√ 242. Valid Anagram	13
√ 525. Contiguous Array	13
√ 349. Intersection of Two Arrays	13
√ 350. Intersection of Two Arrays II +	14
Using normal hash map, time $O(n)$, space $O(n)$	14
If already sorted, two pointer? Time $O(n \lg n)$ to sort, $O(n)$ after sort. Space $O(1)$	14
If one is much shorter than the other, and sorted, use binary search? Time: $O(k \log n)$	15
√ 325. Maximum Size Subarray Sum Equals k +	15
√ 340. Longest Substring with At Most K Distinct Characters	15
√ 159. Longest Substring with At Most Two Distinct Characters	15
209. Minimum Size Subarray Sum	16
X 560. Subarray Sum Equals K ++	16
Extra space store cumulative sum! N^2 time, N space!	16
No Extra Space, but similar idea: N^2 time, $O(1)$ space!	16
Hash Map, one loop! Time $O(n)$, Space $O(n)$	17
49. Group Anagrams	17
Array/String	18
X 189. Rotate Array +	18

Extra array	18
Cyclic in-place rotate	18
Reverse	18
157. Read N Characters Given Read4 +	19
158. Read N Characters Given Read4 II - Call multiple times +	19
√ 13. Roman to Integer	20
X 65. Valid Number ++	20
√ 246. Strobogrammatic Number +	21
简单粗暴	21
Hash Map LUT:	22
√ 277. Find the Celebrity ++	22
31. Next Permutation +	23
56. Merge Intervals +	23
X 57. Insert Interval	24
163: Missing Ranges	24
X New: Desired time for meeting	25
√ 128. Longest Consecutive Sequence	25
X 239. Sliding Window Maximum /Minimum +	26
√ 121. Best Time to Buy and Sell Stock	26
√ 122. Best Time to Buy and Sell Stock II	27
√ 27. Remove Element	27
Keep order	27
Do not keep order: avoid unnecessary moves...	27
√ 26. Remove Duplicates from Sorted Array	27
√ 28. Implement strStr() +	28
√ 161. One Edit Distance	28
X 621. Task Scheduler +++++	28
Priority Queue	28
√ 88. Merge Two Sorted Array ++	29
X. New 1:	30
Matrix	30
√ 311. Sparse Matrix Multiplication +++++++	30
Rank 1 update!	30
X. Variant: Sparse Vector Multiplication	30
X 363. Max Sum of Rectangle No Larger Than K	30
200. Number of Islands +++	31
新题: 找最早出现1的行	32
Sort + Binary Search	32
X 410. Split Array Largest Sum	32
X 215. K-th largest element in an array +	33
Sort, not good enough!	33

Quick Select	33
√ 347. Top K Frequent Elements	33
√ 252. Meeting Rooms +	34
√ 253. Meeting Rooms II ++	34
X 4. Median of Two Sorted Arrays	35
Simply Merge? 复杂度不过关 $O(m+n)$	35
Binary Search! $O(\log(m+n))!!$	35
X 295. Find Median from Data Stream	35
Two Heaps	35
X Two pointer	35
X 33. Search in Rotated Sorted Array ++	35
√ 280. Wiggle Sort	36
Sort and swap:	36
One path	36
√ 69. SQRT(X) +	37
√ 278. First Bad Version ++	37
DP	38
91. Decode Ways +++	38
DP	38
一亩三分地大神 :	38
Follow up: print out all possible decodes:	38
DFS!	39
38. Count and Say	40
139. Word Break ++	40
Tianhao's solution	41
X 140. Word Break II ++	41
10. Regular Expression Matching +	42
√ Recursive:	42
DP	43
√ 44. Wildcard Matching	43
300. Longest Increasing Subsequence	44
72. Edit Distance	44
322. Coin Change	44
304. Range Sum Query 2D - Immutable +++	44
√ Caching each row!	44
X Caching smarter!	44
? Length of Longest Arithmetic Progression in a sorted array	44
Trees	45
√ 109. Convert Sorted List to Binary Search Tree	45
297. Serialize and Deserialize Binary Tree +++++	45
√ 98. Validate Binary Search Tree +	46

√ 173. Binary Search Tree Iterator ++	46
285. Inorder Successor in BST	47
√ 111. Minimum Depth of Binary Tree	47
116. Populating Next Right Pointers in Each Node +	47
Recursive:	47
Iterative	48
X 117. Populating Next Right Pointers in Each Node II	48
298. Binary Tree Longest Consecutive Sequence	49
Top down:	49
Bottom Up:	50
110. Balanced Binary Tree	50
O(n)	50
208. Implement Trie (Prefix Tree) ++	51
212. Word Search II +	52
X 211. Add and Search Word - Data structure design +++	53
X 236. Lowest Common Ancestor of a Binary Tree +	54
X 282. Expression Add Operators	55
Linked List	55
2. Add Two Numbers	55
114. Flatten Binary Tree to Linked List ++	56
X Convert Binary Tree to circular doubly-linked List +	56
√144. Binary Tree Preorder Traversal	56
Recursive	56
Iterative	57
X 206. Reverse Linked List	57
234. Palindrome Linked List	57
X 143. Reorder List ++	58
341. Flatten Nested List Iterator +	59
160. Intersection of Two Linked Lists +	59
X Interleave list of lists	60
X 23. Merge k Sorted Lists +	60
Priority Queue:Space O(k), Time O(nk logk)	60
Divide Conquer :Space O(1), Time O(nk logk)	61
Math	61
168. Excel Sheet Column Title +	61
X 29. Divide Two Integers +	62
273. Integer to English Words +	62
67. Add Binary ++	64
415. Add Strings +	64
X 224. Basic Calculator	65
X 227. Basic Calculator II	65

X 50. POW	65
9. Palindrome Number	66
461. Hamming Distance +	67
Fast and tricky	
int hammingDistance(int x, int y) {	
int z = x ^ y;	
int count = 0;	
while (z){	
z &= (z - 1);	
count ++;	
}	
return count;	
}	67
Slow, iterate all digits	67
477. Total Hamming Distance +	67
DFS/Backtracking/BFS	68
? 新题: 几度好友	68
257. Binary Tree Paths ++	68
X 536. Construct Binary Tree from String ++	68
My first trial, not very clean	68
X 247. Strobogrammatic Number II +	69
First trial: works but slow...	69
79 Word Search +++	70
210. Course Schedule II +	71
BFS	71
46. Permutations (no duplicates!)	72
47. Permutations II (with duplicates!)	72
543. Diameter of Binary Tree +	73
124. Binary Tree Maximum Path Sum	73
102. Binary Tree Level Order Traversal +	74
314. Binary Tree Vertical Order Traversal ++	74
X 301. Remove Invalid Parentheses ++++	75
X 17. Letter Combinations of a Phone Number ++++	76
Iterative	76
Recursive/Backtracking	77
133. Clone Graph ++	77
BFS:	77
DFS	78
X 77. Combinations	78
Backtracking	78
X Iterative	79
78. Subsets ++	79
Recursive	79

Iterative	79
90. Subsets II +	80
39. Combination Sum	80
X 127. Word Ladder	81
51. N-Queens	82
新题	82
parse HTML and build a DOM tree	82
Josephus ring	82
Stack	82
X 71. Simplify Path	82
Data Structure Design	82
348. Design Tic-Tac-Toe +	82
X 146. LRU Cache	83
Give weights, return characters as	84
System Design	85
设计instagram	85
534. Design TinyURL +	85
design messengers's online/offline status.	85
memcache + hashtable	85
POI+	85
网页event售票系统	85
设计脸书search	85
Status更新系统	85
Design a simple message system.	85

Two pointers and/or Hash Map

√ 283. Move Zeros ++++++

Sub Optimal

```
void moveZeroes(vector<int>& nums) {
    int NZcnt = 0;
    for (int i=0; i<nums.size(); i++) if (nums[i]!=0) nums[NZcnt++] = nums[i];
    for (NZcnt; NZcnt<nums.size(); NZcnt++) nums[NZcnt]=0;
    return;
}
```

Space: O(1), Time O(n), 但是永远需要n次operation, 不够optimal

Optimal:

```
void moveZeroes(vector<int>& nums) {
    for (int last_NZ = 0, i = 0; i < nums.size(); i++) {
        if(nums[i] != 0) {
            swap(nums[last_NZ], nums[i]);
            last_NZ ++;
        }
    }
}
```

一遍过，连续出现0的时候直接跳过去。

One line solution:

```
fill(remove(nums.begin(), nums.end(), 0), nums.end(), 0)
```

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=281636>

Follow up: 不在意顺序，移动次数最小!

1. Two Sum +

Hash Map, O(n)

```
vector<int> twoSum(vector<int>& nums, int target) {
    unordered_map<int, int> my_map;
    for ( int i = 0; i < nums.size(); i++ ){
        int complement = target - nums[i];
        if (my_map.find(complement) == my_map.end()) {
            my_map[nums[i]] = i;
        }
        else {
            return vector<int>{my_map[complement], i};
        }
    }
}
```

Sort and Two pointers, O(nlgn)

已经进化到了BST Two sum:

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=291239&ctid=519>

先用InOrder加HashSet解决。O(n) time and O(n) space. 面试官问能否保持O(n) time 同时优化空间。提示可以想象把BST想象成sorted array。回答可以用two pointer。实现两个BST Iterator，一个从小到大，一个从大到小，然后就和sorted array解法一样了。

[173. Binary Search Tree Iterator +](#)

15. 3 Sum +++++

```
vector<vector<int>> threeSum(vector<int>& nums) {
    if(nums.size() <= 2) return{};
    vector<vector<int>> ret;
    sort(nums.begin(), nums.end());
```

```

for (int i=0; i<nums.size()-2; i++){
    int left = i+1;
    int right = nums.size()-1;
    while(left < right){
        int a = nums[left], b = nums[right], target = nums[i];
        int sum = a + b + target;
        if (sum == 0){
            ret.push_back({nums[i], nums[left], nums[right]});
        }
        if (sum >= 0) while(nums[right] == b && right > left) right--;
        if (sum <= 0) while(nums[left] == a && right > left) left++;
    }
    while (nums[i] == target) i++; //tricky to handle duplicates!!!
    //e.g. 1 2 2 2 5, i will stay at first 2, then jump to 5
}
return ret;
}

```

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=279819>

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=282097>

X 76. Minimum Window Substring ++

```

string minWindow(string s, string t) {
    int count = t.size();
    int begin = 0, end = 0;
    int d = INT_MAX, head = 0;
    vector<int> m_map(128, 0);
    for (char c : t) m_map[c]++;

    while(end < s.size()){
        if (m_map[s[end]] > 0) count--;
        m_map[s[end]]--; //decrease no matter if was larger than 0 or not!!!
        end++;

        while(count == 0){
            if (d > end - begin){
                d = end - begin;
                head = begin; //this get assigned only when d<end-begin!!!
            }
            if (m_map[s[begin]] == 0) count++;
            m_map[s[begin]]++;
            begin++;
        }
    }
    return d == INT_MAX ? "" : s.substr(head, d);
}

```


√ 205. Isomorphic Strings ++

Using map[256] trick as hash map:

```
bool isIsomorphic(string s, string t) {
    int ms[256] = {0};
    int mt[256] = {0};
    for (int i = 0; i < s.length(); i++){
        if (ms[s[i]] != mt[t[i]]) {
            return false;
        }
        ms[s[i]] = i + 1;
        mt[t[i]] = i + 1;
    }
    return true;
}
```

坑：ms[s[i]] ++ 这样过不了，只能看前面用了几次，但是“aab” “aba”这两者情况区分不出来！
用i+1的话可以记录上一次在哪个位置被用的
一定要 + 1，或者加几都可以！不然会跟初始值“0”混掉。

√ 125. Valid Palindrome +++++

```
bool isPalindrome(string s) {
    int left = 0;
    int right = s.length() - 1;
    while (left < right) {
        while (!isalnum(s[left]) && left < right) {
            left++;
        }
        while (!isalnum(s[right]) && left < right) {
            right--;
        }
        if (toupper(s[left]) != toupper(s[right])) {
            return false;
        }
        left++;
        right--;
    }
    return true;
}
```

四个坑全踩上了：

坑1：第5，8行while loop，要有条件left < right。不然就容易循环跳出直接返回true了

坑2：要用isalnum而不是isalpha。因为数字的出现不能忽略

坑3：大小写当成一样的，要用toupper

坑4：left++ 和 right--怎么能忘！

523. Continuous Subarray Sum +

```
bool checkSubarraySum(vector<int>& nums, int k) {
    int sum = 0;
    unordered_map<int, int> m_map;
    m_map[0] = -1;
    for (int i = 0; i < nums.size(); i++){
        sum += nums[i];
        if (k != 0)    sum = sum % k;
        if (map.find(sum) != map.end()){
            if (i - map[sum]>1)    return true;
        }
        else    map[sum] = i;
    }
    return false;
}
```

Must be else in red! Otherwise, map[sum] value updates, and can't stay on the first. As a result, $i - \text{map}[\text{sum}] > 1$ never becomes true!!!

X 80. Remove Duplicates from Sorted Array II +

```
int removeDuplicates(vector<int>& nums) {
    int nz_cnt= 0;
    for (int i = 0; i < nums.size(); i++){
        nums[nz_cnt] = nums[i];
        while(i < nums.size() - 2 && nums[i] == nums[i + 1] && nums[i] == nums[i + 2]){
            i++;
        }
        nz_cnt++;
    }
    return ret;
}
```

Very short

```
int i = 0;
for (int n : nums){
    if (i < 2 || n > nums[i-2]) {
        nums[i++] = n;
    }
}
return i;
```

Generic way of allowing K duplicates:

```
int removeDuplicates(vector<int>& nums) {
    if (nums.empty()) return 0;
    int k = 2;
    int i = 1, j = 1;
    int cnt = 1;
```

```

while (j < nums.size()){
    if (nums[j] != nums[j-1]) {
        cnt = 1;
        nums[i++] = nums[j];
    }
    else {
        if (cnt < k) {
            nums[i++] = nums[j];
            cnt++;
        }
    }
    j++;
}
return i;
}

```

√ 383. Ransom Note

Explicit Unordered Map

```

bool canConstruct(string ransomNote, string magazine) {
    unordered_map<char, int> lut;
    for (char c : magazine) {
        lut[c] ++;
    }
    for (char c : ransomNote) {
        if (lut.find(c) == lut.end() || lut[c] == 0) {
            return false;
        }
        else{
            lut[c] --;
        }
    }
    return true;
}

```

Using array size 128

√ 554. Brick Wall

```

int leastBricks(vector<vector<int>>& wall) {
    int tot = wall.size();
    unordered_map<int, int> edges;
    for (int i = 0; i < tot; i++){
        int pos = 0;
        for (int j= 0; j < wall[i].size() - 1; j++){
            pos += wall[i][j];
            edges[pos] += 1;
        }
    }
}

```

```

    int max = 0;
    for (auto it = edges.begin(); it != edges.end(); it++){
        max = it->second > max ? it->second : max;
    }
    return tot - max;
}

```

X 380. Insert Delete GetRandom O(1)

```

class RandomizedSet {
public:
    /** Initialize your data structure here. */
    RandomizedSet(){}
    /** Inserts a value to the set. Returns true if the set did not already contain the
specified element. */
    bool insert(int val) {
        if (m_map.find(val) != m_map.end()) return false;
        nums.push_back(val);
        m_map[val] = nums.size() - 1;
        return true;
    }
    /**Removes a value from the set. Returns true if the set contained the specified element */
    bool remove(int val) {
        if (m_map.find(val) == m_map.end()) return false;
        int idx = m_map[val];
        int end_val = nums[nums.size() - 1];
        nums[idx] = end_val;
        nums.pop_back(); //Don't forget!!!
        m_map[end_val] = idx;
        m_map.erase(val); //also need to erase val from map!
        return true;
    }
    /** Get a random element from the set. */
    int getRandom() {
        return nums[rand()%nums.size()];
    }
private:
    vector<int> nums; //store the numbers, can return a random one in O(1) time
    unordered_map<int, int> m_map; //key is the value of the integer, value is their index
in the nums vector!!!
};
/**
 * Your RandomizedSet object will be instantiated and called as such:
 * RandomizedSet obj = new RandomizedSet();
 * bool param_1 = obj.insert(val);
 * bool param_2 = obj.remove(val);
 * int param_3 = obj.getRandom();
 */

```

√ 242. Valid Anagram

```
class Solution {
private:
    vector<int> count(string input){
        vector<int> table(26, 0);
        int i;
        for (i = 0 ; i < input.size(); i++)    table[input[i] - 'a']++;
        return table;
    }
public:
    bool isAnagram(string s, string t) {
        if (s.size()!=t.size())    return false;
        vector<int> sTable = count(s);
        vector<int> tTable = count(t);

        if (sTable==tTable)    return true;
        else return false;
    }
};
```

√ 525. Contiguous Array

```
int findMaxLength(vector<int>& nums) {
    int count = 0;
    int ret = 0;
    unordered_map<int, int> m_map;
    m_map[0] = -1;
    for (int i = 0; i < nums.size(); i++) {
        count += nums[i] == 0 ? 1 : -1;
        if (m_map.find(count) != map.end()){
            ret = max(ret, i - map[count]);
        }
        else    m_map[count] = i;
    }
    return ret;
}
```

√ 349. Intersection of Two Arrays

```
vector<int> intersection(vector<int>& nums1, vector<int>& nums2) {
    unordered_set<int> nums1_set;
    for (int i : nums1){
        if (nums1_set.find(i) == nums1_set.end()) nums1_set.insert(i);
    }
    vector<int> ret;
    for (int j : nums2){
        if (nums1_set.find(j) != nums1_set.end()){
            ret.push_back(j);
        }
    }
    return ret;
}
```

```

        nums1_set.erase(j);
    }
}
return ret;

```

✓ 350. Intersection of Two Arrays II +

Follow up: what if **already sorted**! What if there is duplicates... how to optimize

Using normal hash map, time $O(n)$, space $O(n)$

```

vector<int> intersect(vector<int>& nums1, vector<int>& nums2) {
    vector<int> ret;
    unordered_map<int, int> m_map;
    for (int num : nums1) {
        m_map[num]++;
    }
    for (int num : nums2) {
        if (m_map.find(num) != m_map.end()) {
            if (m_map[num] > 0) {
                ret.push_back(num);
                m_map[num]--;
            }
            else{
                m_map.erase(num);
            }
        }
    }
    return ret;
}

```

If already sorted, two pointer? Time $O(n \log n)$ to sort, $O(n)$ after sort. Space $O(1)$

```

vector<int> intersect(vector<int>& nums1, vector<int>& nums2) {
    vector<int> ret;
    sort(nums1.begin(), nums1.end());
    sort(nums2.begin(), nums2.end());
    int it1 = 0, it2 = 0;
    while (it1 < nums1.size() && it2 < nums2.size()) {
        if (nums1[it1] < nums2[it2]) {
            it1++;
        }
        else if (nums1[it1] > nums2[it2]) {
            it2++;
        }
        else {
            ret.push_back(nums1[it1]);
            it1++;
            it2++;
        }
    }
}

```

```

        return ret;
    }

```

If one is much shorter than the other, and sorted, use binary search? Time: $O(k \log n)$

√ 325. Maximum Size Subarray Sum Equals k +

```

int maxSubArrayLen(vector<int>& nums, int k) {
    int ret=0;
    unordered_map<int, int> cum_map;
    int cum=0;
    cum_map[0] = -1; //小trick, 当cum sum 等于k时, 需要返回i+1!
    for (int i = 0; i < nums.size(); i++) {
        cum += nums[i];
        if (cum_map.find(cum) == cum_map.end()) cum_map[cum] = i;
        If (cum_map.find(cum-k) != cum_map.end())
            ret = i - cum_map[cum - k] > ret ? i - cum_map[cum - k] : ret;
    }
    return ret;
}

```

√ 340. Longest Substring with At Most K Distinct Characters

```

int lengthOfLongestSubstringKDistinct(string s, int k) {
    int table[128] = {0};
    int j = -1, ret = 0, distinct = 0;
    for(int i = 0; i < s.size(); i++) {
        distinct += table[s[i]] == 0;
        table[s[i]]++;
        while(distinct > k){
            j++;
            table[s[j]]--;
            distinct -= table[s[j]] == 0;
        }
        ret = max(ret, i - j);
    }
    return ret;
}

```

√ 159. Longest Substring with At Most Two Distinct Characters

```

int lengthOfLongestSubstringTwoDistinct(string s) {
    int count = 0;
    int dict[128] = {0};
    int j = -1, ret = 0;
    for (int i = 0; i < s.size(); i++){
        if (dict[s[i]] == 0) count++;
        dict[s[i]]++;
        while (count > 2){
            j++;

```

```

        dict[s[j]]--;
        if (dict[s[j]] == 0) count--;
    }
    ret = max(ret, i - j);
}
return ret;
}

```

跟上一题一个套路！

209. Minimum Size Subarray Sum

```

int minSubArrayLen(int s, vector<int>& nums) {
    int left = 0, sum = 0, ans = INT_MAX;
    for (int i = 0; i < nums.size(); i++){
        sum += nums[i];
        while (sum >= s){
            ans = min(ans, i + 1 - left);
            sum -= nums[left++];
        }
    }
    return ans==INT_MAX? 0: ans;
}

```

依然类似

X 560. Subarray Sum Equals K ++

Extra space store cummulative sum! N^2 time, N space!

```

int subarraySum(vector<int>& nums, int k) {
    int count = 0;
    vector<int> sum(nums.size() + 1);
    sum[0] = 0;
    for(int i = 1; i <= nums.size(); i++) {
        sum[i] = sum[i - 1] + nums[i - 1];
    }
    for (int i = 0; i < nums.size(); i++) {
        for (int j = i + 1; j < nums.size() + 1; j++) {
            if (sum[j] - sum[i] == k) {
                count++;
            }
        }
    }
    return count;
}

```

No Extra Space, but similar idea: N^2 time, $O(1)$ space!

```

int subarraySum(vector<int>& nums, int k) {
    int count = 0;
    for (int i = 0; i < nums.size(); i++) {
        int sum = 0;

```



```

        for (int j = i; j < nums.size(); j++) {
            sum += nums[j];
            if (sum == k) {
                count++;
            }
        }
    }
    return count;
}

```

Hash Map, one loop! Time $O(n)$, Space $O(n)$

```

int subarraySum(vector<int>& nums, int k) {
    int count = 0, sum = 0;
    unordered_map<int, int> my_map;
    my_map[0] = 1;
    for(int i = 0; i < nums.size() ;i++) {
        sum += nums[i];
        if (my_map.find(sum - k) != my_map.end()) {
            count += my_map[sum - k];
        }
        if (my_map.find(sum) == my_map.end()){
            my_map[sum] = 1;
        }
        else{
            my_map[sum]++;
        }
    }
    return count;
}

```

Tianhao's

```

int subarraySum(vector<int>& nums, int k) {
    unordered_map<int, int> count;
    int sum = 0, ret = 0;
    count[0] = 1;
    for(int i = 0; i < nums.size(); i++) {
        sum += nums[i];
        ret += count[sum - k];
        count[sum]++;
    }
    return ret;
}

```

49. Group Anagrams

```

vector<vector<string>> groupAnagrams(vector<string>& strs) {
    unordered_map<string, multiset<string>> mp;
    for (string s: strs){
        string t = s;
        std::sort(t.begin(), t.end());
    }
}

```

```

        mp[t].insert(s);
    }
    vector<vector<string>> res;
    for (auto m : mp){
        vector<string> anagram (m.second.begin(), m.second.end());
        res.push_back(anagram);
    }
    return res;
}

```

Array/String

X 189. Rotate Array +

Extra array

```

void rotate(vector<int>& nums, int k) {
    int len = nums.size();
    k = k % len;
    vector<int> cp = nums;
    for (int i = 0; i < nums.size(); i++) {
        nums[(i + k) % len] = cp[i];
    }
}

```

Cyclic in-place rotate

```

void rotate(vector<int>& nums, int k) {
    int len = nums.size();
    while (k < 0) {
        k += len; //handle what if K is negative, won't be used here
    }
    k = k % len;
    int count = 0;
    for (int i = 0; count < len; i++) {
        int prev = nums[i];
        int start = i;
        int curr = start;
        do{
            curr = (curr + k) % len;
            int temp = nums[curr];
            nums[curr] = prev;
            prev = temp;
            count++;
        } while (curr != start);
    }
}

```

Reverse

```
void rotate(vector<int>& nums, int k) {
    int len = nums.size();
    k = k % len;
    reverse(nums, 0, len - 1);
    reverse(nums, 0, k - 1);
    reverse(nums, k, len - 1);
}

void reverse(vector<int>& nums, int begin, int end) {
    while (begin < end) {
        swap(nums[begin++], nums[end--]);
    }
}
```

157. Read N Characters Given Read4 +

// Forward declaration of the read4 API.

```
int read4(char *buf);
```

```
class Solution {
public:
    /**
     * @param buf Destination buffer
     * @param n    Maximum number of characters to read
     * @return    The number of characters read
     */
    int read(char *buf, int n) {
        int count = 0;
        int m;
        while (count < n && (m = read4(buf + count)) > 0) {
            count += m;
        }
        return count > n ? n : count;
    }
};
```

158. Read N Characters Given Read4 II - Call multiple times +

// Forward declaration of the read4 API.

```
int read4(char *buf);
```

```
class Solution {
public:
    /**
     * @param buf Destination buffer
     * @param n    Maximum number of characters to read
     * @return    The number of characters read
     */
    int read(char *buf, int n) {
```

```

int count = 0;
while (count < n){
    if (buf4_begin < buf4_end) { //buffer里有存货 从buffer里面读
        buf[count++] = buf4[buf4_begin++];
    }
    else if ((buf4_end = read4(buf4)) > 0) { //buffer没有存货 调用read4放进buffer, 并且读第一个
        buf[count++] = buf4[0];
        buf4_begin = 1;
    }
    else break; //如果read4也没返回 说明没了
}
return count;
}
char buf4[4];
int buf4_begin=0;
int buf4_end=0;
};

```

√ 13. Roman to Integer

```

int romanToInt(string s) {
    int len = s.length();
    if (len == 0) {
        return 0;
    }
    unordered_map <char, int> T = { {'I', 1},
                                     {'V', 5},
                                     {'X', 10},
                                     {'L', 50},
                                     {'C', 100},
                                     {'D', 500},
                                     {'M', 1000}, };

    int ret = T[s[len - 1]];
    for (int i = len - 2; i >= 0; i--) {
        if (T[s[i]] >= T[s[i + 1]]) {
            ret += T[s[i]];
        }
        else{
            ret -= T[s[i]];
        }
    }
    return ret;
}

```

X 65. Valid Number ++

```

bool isNumber(string s) {
    int i = 0;
    //skip leading space

```

```

for (; s[i] == ' '; i++) {}
// check signs
if (s[i] == '+' || s[i] == '-') i++;
// check signs before e
int n_num = 0, n_ptr = 0;
for (; (s[i] <= '9' && s[i] >= '0' ) || s[i] == '.'; i++) {
    if (s[i] == '.') { // .1 and 1. are both correct...
        n_ptr++;
    }
    else{
        n_num++;
    }
}
if (n_ptr > 1 || n_num < 1) {
    return false;
}
//check e
if (s[i] == 'e') {
    i++;
    if (s[i] == '+' || s[i] == '-') {
        i++;
    }
    int n_num = 0;
    for (; s[i] >= '0' && s[i] <= '9'; n_num++, i++) {}
    if (n_num < 1) {
        return false;
    }
}
//ending spaces
for (; s[i] == ' '; i++) {}
return i == s.size();
}

```

√ 246. Strobogrammatic Number +

简单粗暴

```

class Solution {
public:
    bool isStrobogrammatic(string num) {
        int left = 0;
        int right = num.length() - 1;
        while (left <= right) {
            if (!isValid(num[left], num[right])){
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
}

```

```
private:
    bool isValid(char left, char right){
        if (left == '0' && right == '0' )        return true;
        else if (left == '1' && right == '1')      return true;
        else if (left == '8' && right == '8')      return true;
        else if (left == '9' && right == '6')      return true;
        else if (left == '6' && right == '9')      return true;
        else    return false;
    }
};
```

Hash Map LUT:

```
class Solution {
public:
    bool isStrobogrammatic(string num) {
        unordered_map<char, char> lut{{'0','0'}, {'1','1'}, {'6','9'}, {'8','8'}, {'9','6'}};
        int left = 0, right = num.length() - 1;
        while (left <= right) {
            if (lut[num[left]] != num[right]) return false;
            left ++;
            right --;
        }
        return true;
    }
};
```

√ 277. Find the Celebrity ++

// Forward declaration of the knows API.

```
bool knows(int a, int b);
```

```
int findCelebrity(int n) {
    if (n <= 1) {
        return n;
    }
    int candidate = 0;
    for (int i = 0; i < n; i++) {
        if (!knows(i, candidate)) {
            candidate = i; //for any i know candidate, i is not qualified as the new
candidate. Only some i not know the candidate can be the new candidate!
        }
    }
    for (int i = 0; i < n; i++) {
        if (i == candidate) continue;
        if (!knows(i, candidate)) {
            return -1;
        }
        if (knows(candidate, i)) {
            return -1;
        }
    }
}
```

```

    }
    return candidate;
}

```

31. Next Permutation +

1. Find the first pair of two successive numbers $a[i]a[i]$ and $a[i-1]a[i-1]$, from the right, which satisfy $a[i] > a[i-1]a[i-1]$.
2. Replace the number $a[i-1]a[i-1]$ with the number which is just larger than itself among the numbers lying to its right section, say $a[j]a[j]$.
3. Swap the numbers $a[i-1]a[i-1]$ and $a[j]a[j]$.
4. Reverse the numbers following $a[i-1]a[i-1]$ to get the next smallest lexicographic permutation.

```

public:
    void nextPermutation(vector<int>& nums) {
        int i = nums.size()-2;
        while(i >= 0 && nums[i] >=nums [i+1]) i--;
        if (i >= 0){
            int j = nums.size() - 1;
            while (nums[j] <= nums[i] && j >= 0) j--;
            swap (nums[i], nums[j]);
        }
        reverse(nums, i+1);
    }
private:
    void reverse(vector<int>& nums, int start){
        int i = start;
        int j = nums.size()-1;
        while (i < j){
            swap(nums[i], nums[j]);
            i++;
            j--;
        }
    }
}

```

Follow up: Previous permutation

56. Merge Intervals +

```

/**
 * Definition for an interval.
 * struct Interval {
 *     int start;
 *     int end;
 *     Interval() : start(0), end(0) {}
 *     Interval(int s, int e) : start(s), end(e) {}
 * };
 */
vector<Interval> merge(vector<Interval>& intervals) {
    vector<Interval> ret;
    if (intervals.empty()) { //vector also has empty() method!
        return ret;
    }
}

```

```

    }
    sort(intervals.begin(), intervals.end(), [](Interval a, Interval b){return a.start
< b.start;}); //this is a lambda
    ret.push_back(intervals[0]);
    for (int i = 1; i < intervals.size(); i++){
        if (ret.back().end < intervals[i].start) { //vector风骚的用法, back()!!
            ret.push_back(intervals[i]);
        }
        else {
            ret.back().end = max(ret.back().end, intervals[i].end);
        }
    }
    return ret;
}

```

Follow up: 二维Interval

X 57. Insert Interval

```

vector<Interval> insert(vector<Interval>& intervals, Interval newInterval) {
    vector<Interval> ret;
    auto it = intervals.begin();
    for(; it!=intervals.end(); ++it){
        if(newInterval.end < (*it).start) //all intervals after will not overlap with
the newInterval
            break;
        else if(newInterval.start > (*it).end) //*it will not overlap with the
newInterval
            ret.push_back(*it);
        else{ //update newInterval because *it overlap with the newInterval
            newInterval.start = min(newInterval.start, (*it).start);
            newInterval.end = max(newInterval.end, (*it).end);
        }
    }
    // don't forget the rest of the intervals and the newInterval
    ret.push_back(newInterval);
    for(; it!=intervals.end(); ++it)
        ret.push_back(*it);
    return ret;
}

```

163: Missing Ranges

```

class Solution {
private:
    string convertString(uint64_t start, uint64_t end){
        if (start == end) return to_string(start);
        else return to_string(start) + "->" + to_string(end);
    }
public:
    vector<string> findMissingRanges(vector<int>& nums, int lower, int upper) {

```



```

vector<string> ret;
int pre = lower - 1;
uint64_t up = (uint64_t) upper+1;
for (int i=0; i<=nums.size(); i++){
    uint64_t curr = (i==nums.size()) ? up : nums[i];
    if (curr - pre >= 2)
        ret.push_back(convertString(pre+1, curr-1));
    pre = curr;
}
return ret;
}
};

```

X New: Desired time for meeting

两个input :

1) desired time range to arrange a meeting -- TimeRange desired_time

2) a list of busy intervals -- List<TimeRange> busy_intervals

[1point3acres.com](http://www.1point3acres.com)

output:

list of time ranges where a meeting can be scheduled, 会议没有时长限制 -- List<TimeRange>

```

class TimeRange {
    double start,
    double end
}

```

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=286594>

```

vector<TimeRange> scheduleMeet(TimeRange desired_time, vector<TimeRange> busy_intervals){
    vector<TimeRange> ret;
    double s = desired_time.start;
    double e = desired_time.end;
    sort(busy_intervals.begin(), busy_intervals.end())[(TimeRange a, TimeRange
b){return a.start<b.start});
    for (auto I : busy_intervals){
        if (I.start > e) break;
        if (I.end < s) continue;
        If (I.start > s) {
            TimeRange tmp{s, I.start};
            ret.push_back(tmp);
        }
        s = I.end();
    }
    if (s < e) ret.push_back(TimeRange {s,e});
}

```

√ 128. Longest Consecutive Sequence

```

int longestConsecutive(vector<int>& nums) {
    unordered_set<int> nums_set;

```

```

for(int num:nums) nums_set.insert(num);
int max = 0;
int tot;
for(int i : nums_set){
    int pre = 0, post = 0;
    int n = i, m = i;
    while (nums_set.find(n - 1) != nums_set.end()){
        n--;
        pre++;
        nums_set.erase(n);
    }
    while (nums_set.find(m + 1) != nums_set.end()){
        m++;
        post++;
        nums_set.erase(m);
    }
    int tot = pre + post + 1;
    max = tot > max ? tot : max;
}
return max;
}

```

X 239. Sliding Window Maximum /Minimum +

```

vector<int> maxSlidingWindow(vector<int>& nums, int k) {
    deque<int> dq;
    vector<int> ans;
    for (int i = 0; i < nums.size(); i++) {
        if (!dq.empty() && dq.front() == i - k) dq.pop_front();
        while (!dq.empty() && nums[dq.back()] < nums[i])
            dq.pop_back();
        dq.push_back(i); //无论如何都要push back i !!!
        if (i >= k - 1) ans.push_back(nums[dq.front()]);
    }
    return ans;
}

```

√ 121. Best Time to Buy and Sell Stock

```

int maxProfit(vector<int>& prices) {
    int max_itvl = 0;
    int min = INT_MAX;
    for (int p : prices){
        min = p < min ? p : min;
        max_itvl = (p - min) > max_itvl ? p - min: max_itvl;
    }
    return max_itvl;
}

```

√ 122. Best Time to Buy and Sell Stock II

```
int maxProfit(vector<int>& prices) {
    if (prices.size()==0) return 0;
    int ret=0;
    for (int i=1; i<prices.size(); i++){
        if (prices[i]>prices[i-1]) ret+=prices[i]-prices[i-1];
    }
    return ret;
}
```

如果考虑transaction cost就不能这么粗暴了...

√ 27. Remove Element

Keep order

```
int removeElement(vector<int>& nums, int val) {
    int count = 0;
    for(int i = 0; i < nums.size(); i++) {
        if (nums[i] == val) count++;
        else{
            nums[i - count] = nums[i]; // or swap();
        }
    }
    return nums.size() - count;
}
```

<http://www.1point3acres.com/bbs/thread-285255-1-1.html>

Do not keep order: avoid unnecessary moves...

```
int removeElement(vector<int>& nums, int val) {
    int n = nums.size(), i = 0;
    while (i < n) {
        if (nums[i] == val) {
            nums[i] = nums[n - 1];
            n--;
        }
        else {
            i++;
        }
    }
    return n;
}
```

√ 26. Remove Duplicates from Sorted Array

Actually also **two pointers**! Very similar to 27

```
int removeDuplicates(vector<int>& nums) {
    int ret = 0;
```

```

    if (nums.size() == 0) return 0;
    for(int i = 0; i < nums.size(); i++) {
        nums[ret] = nums[i];
        ret++;
        while (i < nums.size() - 1 && nums[i] == nums[i + 1]) i++;
    }
    return ret;
}

```

√ 28. Implement strStr() +

```

int strStr(string haystack, string needle) {
    int m = haystack.length();
    int n = needle.length();
    if (n == 0) return 0;
    for (int i = 0; i < m - n + 1; i++) {
        int j = 0;
        for (; j < n; j++){
            if (haystack[i+j] != needle[j]) break;
        }
        if (n == j) return i;
    }
    return -1;
}

```

√ 161. One Edit Distance

```

bool isOneEditDistance(string s, string t) {
    int m = s.length();
    int n = t.length();
    if (m < n) return isOneEditDistance(t, s);
    if (m - n > 1) return false;
    bool mismatch = false;
    for (int i = 0; i < n; i++){
        if (s[i] != t[i]){
            if (m == n) s[i] = t[i];
            else t.insert(i, 1, s[i]);
            mismatch = true;
            break;
        }
    }
    return (!mismatch && m - n == 1) || (mismatch && s == t);
}

```

X 621. Task Scheduler +++++

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=279020>

Priority Queue

```

int leastInterval(vector<char>& tasks, int n) {

```

}

各种神奇的变种：

follow up: allow reordering of tasks. (priority queue, greedy)

complexity analysis, etc.

✓ 88. Merge Two Sorted Array ++

}

X. New 1:

Find all characters that have most continuous appearances (longest sequence). 比如：“this send meet” -> [s, e] 再比如：“this is pea” -> [t, h, i, s, i, s, p, e, a] 这个题他只给出input和output让猜程序是干什么的。没有见过的话，可能需要稍微分析一下。是个新题？我没在面经见过。 Tricky part: 注意skip空格

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=282097>

Matrix

√ 311. Sparse Matrix Multiplication ++++++

Rank 1 update!

```
vector<vector<int>> multiply(vector<vector<int>>& A, vector<vector<int>>& B) {
    vector<vector<int>> ret(A.size(), vector<int>(B[0].size(), 0));
    for (int m = 0; m < A.size(); m++){
        for (int n = 0; n < B.size(); n++){
            if (A[m][n] != 0){
                for (int j = 0; j < B[0].size(); j++){
                    ret[m][j] += A[m][n]*B[n][j];
                }
            }
        }
    }
    return ret;
}
```

X. Variant: Sparse Vector Multiplication

Note: 要自己提出结构并计算内积 一开始用了dict，被要求用别的“在某些方面更好的”结构，就用了array of tuple.

X 363. Max Sum of Rectangle No Larger Than K

```
int maxSumSubmatrix(vector<vector<int>>& matrix, int k) {
    if (matrix.empty()) return 0;
    int row = matrix.size(), col = matrix[0].size(), res = INT_MIN;
    for (int l = 0; l < col; ++l) {
        vector<int> sums(row, 0);
        for (int r = l; r < col; ++r) {
            for (int i = 0; i < row; ++i) {
                sums[i] += matrix[i][r];
            }
            // Find the max subarray no more than K
            set<int> accuSet;
```

```

        accuSet.insert(0);
        int curSum = 0, curMax = INT_MIN;
        for (int sum : sums) {
            curSum += sum;
            set<int>::iterator it = accuSet.lower_bound(curSum - k);
            if (it != accuSet.end()) curMax = std::max(curMax, curSum - *it);
            accuSet.insert(curSum);
        }
        res = std::max(res, curMax);
    }
}
return res;
}

```

200. Number of Islands ++

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=234534>

```

class Solution {
public:
    int numIslands(vector<vector<char>>& grid) {
        int m = grid.size();
        if (!m) return 0;
        int n = grid[0].size();
        if (!n) return 0;
        int res = 0;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (grid[i][j] == '1') {
                    res++;
                    dfs(grid, i, j);
                }
            }
        }
        return res;
    }
    void dfs(vector<vector<char>>& grid, int x, int y) {
        grid[x][y] = 0;
        if (x > 0 && grid[x - 1][y] == '1') {
            dfs(grid, x - 1, y);
        }
        if (x < grid.size() - 1 && grid[x + 1][y] == '1') {
            dfs(grid, x + 1, y);
        }
        if (y < grid[0].size() - 1 && grid[x][y + 1] == '1') {
            dfs(grid, x, y + 1);
        }
        if (y > 0 && grid[x][y - 1] == '1') {
            dfs(grid, x, y - 1);
        }
    }
};

```

新题: 找最早出现1的行

Given a 2D array. Each row is constructed by 0's at the beginning and 1's at the following.

ex: [
 [0,0,0,0,0,1,1]
 [0,0,0,1,1,1,1]
 [0,0,0,0,1,1,1]
]

Return the first column number that has 1 occurs. In the example it should be 3

最优解：右上开始，遇到1向左，遇到0向下。

Sort + Binary Search

X 410. Split Array Largest Sum

```
class Solution {
private:
    bool canSplit(vector<int>& nums, int m, int bound){
        int cnt = 1;
        int curSum = 0;
        for (int i = 0; i < nums.size(); i++){
            curSum += nums[i];
            if (curSum > bound){
                curSum = nums[i];
                cnt ++;
                if (cnt > m) return false;
            }
        }
        return true;
    }
public:
    int splitArray(vector<int>& nums, int m) {
        int left = 0, right = 0;
        for (int i = 0; i < nums.size(); i++){
            right += nums[i];
            left = max(left, nums[i]);
        }
        while(left < right){
            int mid = left + (right - left) / 2;
            if(canSplit(nums, m, mid)) right = mid;
            else left = mid + 1;
        }
        return left;
    }
};
```


X 215. K-th largest element in an array +

Sort, not good enough!

```
int findKthLargest(vector<int>& nums, int k) {
    sort(nums.begin(), nums.end());
    return nums[nums.size() - k];
}
```

Quick Select

With idea of quick sort, leveraging “pivot” concept: Time Complexity $O(n)$ on average

Algorithm:

Initialize left: 0, right: nums.size() - 1

Pivot is left, Position the array, if pivot at (k - 1)th location, done

If pivot right to (k - 1)th position, right = pivot - 1

If pivot left to (k - 1)th position, left = pivot + 1

```
class Solution {
public:
    int findKthLargest(vector<int>& nums, int k) {
        int left=0, right=nums.size()-1;
        while(1){
            int p = partition(nums, left, right);
            if (p == k - 1) return nums[p];
            if (p < k - 1) left = p + 1;
            else right = p - 1;
        }
    }

private:
    //left and right defines the boundaries for partition. left is the pivot.
    int partition(vector<int>& nums, int left, int right){
        int pivot = nums[left];
        int l = left + 1, r = right;
        while(l <= r){ //ERROR: l<r or l<=r???
            if (nums[l] < pivot && nums[r] > pivot) swap(nums[l++], nums[r--]);
            if (nums[l] >= pivot) l++; //ERROR: not else if here!!!
            if (nums[r] <= pivot) r--;
        }
        swap(nums[left], nums[r]);
        return r;
    }
};
```

Follow up: input 是stream怎么办, heap=> push all to priority queue and then pop k

√ 347. Top K Frequent Elements

```
vector<int> topKFrequent(vector<int>& nums, int k) {
    unordered_map<int, int> m_map;
```

```

for(int i : nums) m_map[i]++;
vector<int> ret;
priority_queue<pair<int, int>> pq;
for(auto it = m_map.begin(); it != m_map.end(); it++){
    pq.push(make_pair(it->second, it->first));
}
while (k--){
    ret.push_back(pq.top().second);
    pq.pop();
}
return ret;
}

```

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=283358>

√ 252. Meeting Rooms +

```

bool canAttendMeetings(vector<Interval>& intervals) {
    vector<pair<int, int>> time;
    for (Interval I : intervals){
        time.push_back({I.start, 1});
        time.push_back({I.end, -1});
    }
    sort(time.begin(), time.end());
    int accu = 0;
    for (auto t : time) {
        accu += t.second;
        if (accu > 1) return false;
    }
    return true;
}

```

Follow up: what if adjacent means can't make it!

Make -1 as start and 1 as end. Then after sort, start will be calculated after end. Ceck if accu < -1 !

√ 253. Meeting Rooms II ++

```

int minMeetingRooms(vector<Interval>& intervals) {
    vector<pair<int, int>> Time;
    for (int i = 0; i<intervals.size(); i++){
        Time.push_back({intervals[i].start, 1});
        Time.push_back({intervals[i].end, -1});
    }
    sort(Time.begin(), Time.end());
    int max = 0;
    int accu = 0;
    for (int i = 0; i < Time.size(); i++){
        accu += Time[i].second;
        max = max<accu ? accu:max;
    }
}

```

```
        return max;
    }

```

Variant:

有点变化，比如[[1,2], [2,3], [3,4]]要返回2， [[1,1], [1,1], [1,1]]要返回3。

X 4. Median of Two Sorted Arrays

Simply Merge? 复杂度不过关 $O(m+n)$

Binary Search! $O(\log(m+n))!!$

X 295. Find Median from Data Stream

Two Heaps

```
class MedianFinder {
    priority_queue<int> max_heap;
    priority_queue<int, vector<int>, greater<int>> min_heap;
public:
    /** initialize your data structure here. */
    MedianFinder() { }
    void addNum(int num) {
        max_heap.push(num);
        min_heap.push(max_heap.top());
        max_heap.pop();
        if (max_heap.size() < min_heap.size()) {
            max_heap.push(min_heap.top());
            min_heap.pop();
        }
    }
    double findMedian() {
        if (max_heap.size() > min_heap.size()) {
            return max_heap.top();
        }
        else{
            return (max_heap.top() + min_heap.top()) / 2.0;
        }
    }
};

```

X Two pointer

X 33. Search in Rotated Sorted Array ++

```
int search(vector<int>& nums, int target) {
    if (nums.empty()) return -1;
    int start = 0, end = nums.size() - 1;

```

```

while (start + 1 < end) {
    int mid = start + (end - start) / 2;
    if (nums[mid] >= nums[start]) {
        if (target <= nums[mid] && target >= nums[start]) {
            end = mid;
        }
        else {
            start = mid;
        }
    }
    else {
        if (target >= nums[mid] && target <= nums[end]) {
            start = mid;
        }
        else {
            end = mid;
        }
    }
}
if (nums[start] == target) {
    return start;
}
if (nums[end] == target) {
    return end;
}
return -1;
}

```

√ 280. Wiggle Sort

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=279863>

<http://www.geeksforgeeks.org/rearrange-array-maximum-minimum-form-set-2-o1-extra-space/>

Sort and swap:

```

void wiggleSort(vector<int>& nums) {
    if (nums.size()==0) return;
    sort(nums.begin(), nums.end());
    for (int i = 1; i < nums.size() - 1; i += 2){
        swap(nums[i], nums[i + 1]);
    }
}

```

One path

```

void wiggleSort(vector<int>& nums) {
    if (nums.size() == 0) return;
    bool less = true;
    for (int i = 0; i < nums.size() - 1; i++) {
        if (less) {
            if (nums[i] > nums[i+1]) swap(nums[i], nums[i+1]);
        }
        else {
            if (nums[i] < nums[i+1]) swap(nums[i], nums[i+1]);
        }
        less = !less;
    }
}

```

```

    }
    else {
        if (nums[i] < nums[i+1]) swap(nums[i], nums[i+1]);
    }
    less = !less;
}
}

```

√ 69. Sqrt(X) +

```

int mySqrt(int x) {
    if (x == 0) return 0;
    uint64_t left = 1, right = x;
    while (left + 1 < right){
        int mid = left + (right - left) / 2;
        if (mid == x / mid) { //写成 mid * mid == x就overflow咯
            return mid;
        }
        else if (mid > x / mid) {
            right = mid;
        }
        else{
            left = mid;
        }
    }
    if (right == x / right) {
        return right;
    }
    return left;
}

```

√ 278. First Bad Version ++

```

bool isBadVersion(int version);
class Solution {
public:
    int firstBadVersion(int n) {
        uint64_t start = 1;
        uint64_t end = n;
        while (start + 1 < end){
            int mid = start + (end - start) / 2;
            if (isBadVersion(mid)){
                end = mid;
            }
            else{
                start = mid + 1;
            }
        }
        if (isBadVersion(start)) {
            return start;
        }
    }
}

```

```

    }
    return end;
}
};

```

DP

91. Decode Ways +++

DP

```

int numDecodings(string s) {
    int len = s.length();
    if (len == 0) return 0;
    // r2: decode ways until s[i-1] , r1: decode ways until s[i]
    int r1 = 1, r2 = 1;
    for (int i = 0; i < len; i++){
        // zero voids ways of the last because zero cannot be used separately
        if (s[i] == '0'){
            r1 = 0;
        }
        // possible two-digit letter, so new r1 is sum of both while new r2 is the old r1
        if (s[i - 1] == '1' || (s[i - 1] == '2' && s[i] <= '6')){
            r1 = r1 + r2;
            r2 = r1 - r2; //original r1
        }
        else{ //no new ways added
            r2 = r1;
        }
    }
    return r1;
}

```

Follow up:

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=283430>

1. What if * included
2. Print out all possible decodes

一亩三分地大神：

```

int numDecodings(string& s) { // dfs + memory
    vector<int> m(s.size()+1); 1point3acres.com
    function<int(int)> dfs = [&](int i){ return m = (m ? m : (s=='0' ? 0 : (dfs(i+1) + ((i<s.size()-1 && (s=='1' || (s=='2'&&s[i+1]<='6')) ? dfs(i+2) : 0)))));};
    return s.empty() ? 0 : (m[s.size()] = 1, dfs(0));
}

```

Follow up: print out all possible decodes:

```

void helper(vector<vector<int> >& dp, vector<string>& res, string& s, int cur, const string& ss) {
    if (cur == 0) {

```

```

    string tmp = s;
    reverse(tmp.begin(), tmp.end());
    res.push_back(tmp);
    return;
}
for (auto prevIdx: dp[cur]) {
    int offset = stoi(ss.substr(prevIdx, cur - prevIdx));. visit 1point3acres.com for more.
    char c = 'a' - 1 + offset;
    s.push_back(c);
    helper(dp, res, s, prevIdx, ss);
    s.pop_back();
}
}
vector<string> decode(const string& s) {
    if (s.empty() || s[0] == '0')
        return {};
    vector<vector<int> > dp(s.size() + 1, {});
    dp[1].push_back(0);
    for (int i = 0; i < s.size() - 1; ++i) {
        char a = s[i];
        char b = s[i + 1];
        if (b == '0') {
            if (a == '0' || a >= '3') {
                return {};
            }
            dp[i + 2].push_back(i);. 1point3acres.com/bbs
        }
        else if (a == '1' || (a == 2 && b <= '6')) {
            dp[i + 2].push_back(i + 1);
            dp[i + 2].push_back(i);
        }
        else {
            dp[i + 2].push_back(i + 1);
        }
    }
    vector<string> res;
    string ss;
    helper(dp, res, ss, s.size(), s);
    return res;
}

```

DFS!

```

class Solution {
public:
    int numDecodings(string s) {
        int ret = 0;
        if (s.length() == 0) return 0;
        return dfs(s);
    }
private:
    int dfs(string s){

```

```

    int ret = 0;
    if (s.length() == 0) return 1; //not 0!! 这个区别导致必须分开一个dfs
//一直就从0开始加，加到没了说明这是一个decode的方法。所以最后要加一！
    if (s.length() == 1 && isValid(s)) return 1; //排除单一个0
    if (isValid(s.substr(0, 1))) {
        ret += dfs(s.substr(1)); //这样写就是从1到最后

    if (isValid(s.substr(0, 2))) {
        ret += dfs(s.substr(2));
    }
    return ret;
}
bool isValid(string s) {
    if (s.length() == 1 && s[0] != '0') {
        return true;
    }
    if (s.length() == 2) {
        if (s[0] == '1' || (s[0] == '2' && s[1] <= '6')) {
            return true;
        }
    }
    return false;
}
};

```

38. Count and Say

```

string countAndSay(int n) {
    string ret = "1";
    string curr = "1";
    while (--n) {
        ret = "";
        for (int i = 0; i < curr.size(); i++){
            int count = 1;
            while (i < curr.size() - 1 && curr[i] == curr[i + 1]) {
                count ++;
                i ++;
            }
            ret += to_string(count) + curr[i];
        }
        curr = ret;
    }
    return ret;
}

```

139. Word Break ++

```

bool wordBreak(string s, vector<string>& wordDict) {
    if (s.length()==0) return false;
    unordered_set<string> m_dict;

```



```

    for (string word : wordDict) {
        m_dict.insert(word);
    }
    vector<bool> dp(s.length() + 1, false);
    dp[0] = true;
    for (int i = 1; i <= s.length(); i++) {
        for (int j = i - 1; j >= 0; j--) {
            if (dp[j]) {
                if (m_dict.find(s.substr(j, i - j)) != m_dict.end()) {
                    dp[i] = true;
                    break;
                }
            }
        }
    }
    return dp[s.length()];
}

```

Tianhao's solution

```

bool wordBreak(string s, vector<string>& wordDict) {
    vector<bool> visit = vector<bool>(s.size(), false);
    queue<int> todo;
    todo.push(0);
    while(todo.size() != 0){
        int temp = todo.front();
        todo.pop();
        if(temp == s.size()){
            return true;
        }
        if(!visit[temp]){
            visit[temp] = true;
            for(int i = 0; i < wordDict.size(); i++){
                if(s.substr(temp, wordDict[i].size()) == wordDict[i]){
                    todo.push(temp + wordDict[i].size());
                }
            }
        }
    }
    return false;
}

```

X 140. Word Break II ++

```

class Solution {
public:
    vector<vector<vector<int>>> dp;
    vector<bool> exist;
    void search(int length, vector<string>& wordDict, string& s){
        if(!exist[length]){
            for(int i = 0; i < wordDict.size(); i++){

```

```

        int length_next=length-wordDict[i].size();
        if(length_next>=0 && wordDict[i]==s.substr(length_next,
wordDict[i].size())){
            if(length_next==0){
                dp[length].push_back(vector<int>(1, i));
            }else{
                search(length_next, wordDict, s);
                vector<vector<int>> temp=dp[length_next];
                for(int k=0; k<temp.size(); k++){
                    temp[k].push_back(i);
                }
                for(int k=0; k<temp.size(); k++){
                    dp[length].push_back(temp[k]);
                }
            }
        }
        exist[length]=true;
    }
}
vector<string> wordBreak(string s, vector<string>& wordDict) {
    vector<string> ret;
    dp=vector<vector<vector<int>>>(s.size()+1, vector<vector<int>>());
    exist=vector<bool>(s.size()+1, false);
    exist[0]=true;
    search(s.size(), wordDict, s);
    for(vector<int> it:dp[s.size()]){
        ret.push_back(string());
        for(int i:it){
            if(!ret.back().empty()){
                ret.back()+=' ';
            }
            ret.back()+=wordDict[i];
        }
    }
    return ret;
}
};

```

只返回一种结果，算worst time complexity。 follow up 减枝优化

10. Regular Expression Matching +

√ Recursive:

```

bool isMatch(string s, string p) {
    if (p.empty()){
        return s.empty();
    }
    if (p[1] == '*') {

```

```

        return isMatch(s, p.substr(2)) || !s.empty() && (s[0] == p[0] || p[0] == '.')
&& isMatch(s.substr(1), p);
    }
    else{
        return !s.empty() && (s[0] == p[0] || p[0] == '.') && isMatch(s.substr(1),
p.substr(1));
    }
}

```

DP

```

bool isMatch(string s, string p) {
    int m = s.length(), n = p.length();
    vector<vector<bool>> dp(m + 1, vector<bool>(n + 1, false));
    dp[0][0] = true;
    //dp[i][j] means match or not by s[i-1] and p[j-1]
    //Preprocessing: all preceding "x*" can match empty string!
    for (int j = 2; j <= n; j++) {
        dp[0][j] = dp[0][j - 2] && p[j - 1] == '*';
    }
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++){
            if (p[j - 1] != '*') {
                dp[i][j] = dp[i - 1][j - 1]&&(s[i - 1] == p[j - 1] || p[j - 1] == '.');
            }
            else{
                //p[j - 1] == '*'
                1) "x*" match nothing
                2) "x*" match s[i - 1]. If mutiple, this will repeat
                dp[i][j] = dp[i][j-2]||(s[i-1] == p[j-2]||p[j-2]=='.')&&dp[i - 1][j];
            }
        }
    }
    return dp[m][n];
}

```

√ 44. Wildcard Matching

```

bool isMatch(string s, string p) {
    int m = s.size(), n = p.size();
    vector<vector<bool>> dp = vector<vector<bool>>(m + 1, vector<bool>(n + 1, false));
    dp[0][0] = true;
    for (int i = 1; i <= n; i++) {
        dp[0][i] = dp[0][i - 1] && p[i - 1] == '*';
    }
    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (p[j - 1] != '*') {
                dp[i][j]=dp[i - 1][j - 1] && (s[i - 1] == p[j - 1] || p[j - 1] == '?');
            }
            else {

```

```

        dp[i][j] = dp[i - 1][j] || dp[i][j - 1];
    }
}
}
return dp[m][n];
}

```

300. Longest Increasing Subsequence

72. Edit Distance

322. Coin Change

304. Range Sum Query 2D - Immutable +++

✓ Caching each row!

```

class NumMatrix {
vector<vector<int>>> cum_matrix;
public:
    NumMatrix(vector<vector<int>>> matrix) {
        cum_matrix.resize(matrix.size());
        for (int r = 0; r < matrix.size(); r++) {
            cum_matrix[r].resize(matrix[0].size() + 1);
            cum_matrix[r][0] = 0;
            for (int c = 1; c <= matrix[0].size(); c++) {
                cum_matrix[r][c] = cum_matrix[r][c - 1] + matrix[r][c - 1];
            }
        }
    }

    int sumRegion(int row1, int col1, int row2, int col2) {
        int sum = 0;
        for (int r = row1; r <= row2; r++) {
            sum += cum_matrix[r][col2 + 1] - cum_matrix[r][col1];
        }
        return sum;
    }
};

```

X Caching smarter!

? Length of Longest Arithmetic Progression in a sorted array

<http://www.geeksforgeeks.org/length-of-the-longest-arithmetic-progression-in-a-sorted-array/>

要求返回最长sequence

Trees

✓ 109. Convert Sorted List to Binary Search Tree

```
TreeNode* sortedListToBST(ListNode* head) {
    if (!head) return NULL;
    if (!head->next) return new TreeNode(head->val);
    ListNode* slow = head;
    ListNode* pre = head;
    ListNode* fast = head;
    if (!head->next->next){
        TreeNode* root=new TreeNode(head->next->val);
        root->left = new TreeNode(head->val);
        return root;
    }
    while(fast->next && fast->next->next){
        pre = slow;
        slow = slow->next;
        fast = fast->next->next;
    }
    TreeNode* root = new TreeNode(slow->val);
    ListNode* right = slow->next;
    pre->next=NULL;
    root->left = sortedListToBST(head);
    root->right = sortedListToBST(right);
    return root;
}
```

297. Serialize and Deserialize Binary Tree +++++

```
class Codec {
public:
    // Encodes a tree to a single string.
    string serialize(TreeNode* root) {
        ostringstream out;
        serialize(root, out);
        return out.str();
    }
    // Decodes your encoded data to tree.
    TreeNode* deserialize(string data) {
        istringstream in(data);
        return deserialize(in);
    }
private:
    void serialize(TreeNode* root, ostringstream &out){
        if (!root) out << "# ";
        else{
            out << root->val << ' ';
        }
    }
}
```

```

        serialize(root->left, out);
        serialize(root->right, out);
    }
}
TreeNode* deserialize(istringstream &in) {
    string val;
    in >> val;
    if (val == "#") return NULL;
    TreeNode* root = new TreeNode(stoi(val));
    root->left = deserialize(in);
    root->right = deserialize(in);
    return root;
}
};

```

Variation: print the serialization as vector!

✓ 98. Validate Binary Search Tree +

```

class Solution {
public:
    bool isValidBST(TreeNode* root) {
        return isValidBST(root, NULL, NULL);
    }
private:
    bool isValidBST(TreeNode* root, TreeNode* min, TreeNode* max){
        if (!root) return true;
        if (min && root->val <= min->val)    return false;
        if (max && root->val >= max->val)    return false;
        return isValidBST(root->left, min, root) && isValidBST(root->right, root, max);
    }
};

```

✓ 173. Binary Search Tree Iterator ++

```

class BSTIterator {
    stack<TreeNode*> st;
public:
    BSTIterator(TreeNode *root) {
        find_left(root);
    }
    /** @return whether we have a next smallest number */
    bool hasNext() {
        return !st.empty();
    }
    /** @return the next smallest number */
    int next() {
        TreeNode *top = st.top();
        st.pop(); //The node's right is smaller than the node's parent!
        if (top->right) find_left(top->right);
        return top->val;
    }
};

```

```

}
void find_left(TreeNode *root){
    while(root){
        st.push(root);
        root = root->left;
    }
}
};

```

285. Inorder Successor in BST

```

TreeNode* inorderSuccessor(TreeNode* root, TreeNode* p) {
    TreeNode* candidate = NULL;
    while(root){
        if (root->val > p->val){
            candidate=root;
            root = root->left;
        }
        else    root =root->right;
    }
    return candidate;
}

```

√ 111. Minimum Depth of Binary Tree

```

int minDepth(TreeNode* root) {
    int ret=0;
    if (!root) return ret;
    queue<TreeNode*> bfsq;
    bfsq.push(root);
    while(!bfsq.empty()){
        ret++; //Count number of nodes, not number of edges!
        int sz = bfsq.size();
        for (int i=0; i<sz; i++){
            TreeNode* temp = bfsq.front();
            bfsq.pop();
            if (temp->left) bfsq.push(temp->left);
            if (temp->right) bfsq.push(temp->right);
            if (!temp->left && !temp->right) return ret;
        }
    }
}

```

116. Populating Next Right Pointers in Each Node +

Recursive:

```

void connect(TreeLinkNode *root) {
    if (!root) return;
    if (root->left){
        root->left->next = root->right;
    }
}

```

```

        if (root->next){
            root->right->next = root->next->left;
        }
    }
    connect(root->left);
    connect(root->right);
}

```

Iterative

```

void connect(TreeLinkNode *root) {
    if (!root) return;
    while (root->left){ //perfect binary tree
        TreeLinkNode* p = root;
        while(p){
            p->left->next = p->right;
            if (p->next){
                p->right->next = p->next->left;
            }
            p = p->next;
        }
        root=root->left;
    }
}

```

此题follow up凶险，最右node要指向下一行最左...

Follow up needs to be done by iterative method?

```

void connect(TreeLinkNode *root) {
    if (!root) return;
    while (root->left){ //perfect binary tree
        TreeLinkNode* p = root;
        while(p){
            p->left->next = p->right;
            if (p->next){
                p->right->next = p->next->left;
            }
            else{
                p->right->next = root->left;
            }
            p = p->next;
        }
        root=root->left;
    }
}

```

X 117. Populating Next Right Pointers in Each Node II

```

void connect(TreeLinkNode *root) {
    TreeLinkNode* curr;
    TreeLinkNode* curr_next;
    TreeLinkNode* prev_next;
    curr=root;

```



```

while(curr){
    if(curr->left){
        if(!curr_next){
            curr_next=curr->left;
        }
        if(!prev_next){
            prev_next=curr->left;
        }else{
            prev_next->next=curr->left;
            prev_next=curr->left;
        }
    }
    if(curr->right){
        if(!curr_next){
            curr_next=curr->right;
        }
        if(!prev_next){
            prev_next=curr->right;
        }else{
            prev_next->next=curr->right;
            prev_next=curr->right;
        }
    }
    curr=curr->next;
    if(!curr){
        curr=curr_next;
        curr_next=nullptr;
        prev_next=nullptr;
    }
}
}

```

298. Binary Tree Longest Consecutive Sequence

Top down:

```

class Solution {
    int longest = 0;
    void dfs(TreeNode* root, TreeNode* parent, int length){
        if (!root) return;
        length = (parent && root->val == parent->val + 1) ? length + 1 : 1;
        longest = max(longest, length);
        dfs(root->left, root, length);
        dfs(root->right, root, length);
    }
public:
    int longestConsecutive(TreeNode* root) {
        dfs(root, nullptr, 0);
        return longest;
    }
};

```

Bottom Up:

```
class Solution {
    int longest = 0;
    int dfs(TreeNode* root){
        if (!root) return 0;
        int L = dfs(root->left) + 1;
        int R = dfs(root->right) + 1;
        if (root->left && root->val + 1 != root->left->val) {
            L = 1;
        }
        if (root->right && root->val + 1 != root->right->val) {
            R = 1;
        }
        int length = max(L, R);
        longest = max(length, longest);
        return length;
    }
public:
    int longestConsecutive(TreeNode* root) {
        dfs(root);
        return longest;
    }
};
```

110. Balanced Binary Tree

```
class Solution {
    bool balanced = true;
public:
    bool isBalanced(TreeNode* root) {
        depth(root);
        return balanced;
    }
private:
    int depth(TreeNode* root){
        if (!root) return 0;
        int left_depth = depth(root->left);
        int right_depth = depth(root->right);
        if (abs(left_depth - right_depth) > 1) balanced = false;
        return max(left_depth, right_depth) + 1;
    }
};
```

$O(n)$

```
bool isBalanced(TreeNode* root) {
    if (!root) return true;
    return height(root) != -11; //-11 is a marker of unbalanced
}
int height(TreeNode* root){
```

```

        if (!root) return -1;
        int l = height(root->left);
        int r = height(root->right);
        if (l == -11 || r == -11 || labs(l - r) > 1) {
            return -11;
        }
        return 1 + max(l, r);
    }
}

```

208. Implement Trie (Prefix Tree) ++

```

class TrieNode{
public:
    bool is_word;
    TrieNode* next[26];
    TrieNode(bool b=false){
        memset(next, 0, sizeof(next));
        is_word = b;
    }
};

class Trie {
TrieNode* root;
public:
    /** Initialize your data structure here. */
    Trie() {
        root = new TrieNode();
    }
    /** Inserts a word into the trie. */
    void insert(string word) {
        TrieNode* p = root;
        for (int i = 0; i < word.size(); i++){
            if (p->next[word[i] - 'a'] == NULL) {
                p->next[word[i] - 'a'] = new TrieNode();
            }
            p = p->next[word[i] - 'a'];
        }
        p->is_word = true;
    }
    /** Returns if the word is in the trie. */
    bool search(string word) {
        TrieNode* p = root; //p is the runner
        for (int i = 0; i < word.size(); i++){
            if (p) {
                p=p->next[word[i] - 'a'];
            }
            else return false;
        }
        return p != NULL && p->is_word;
    }
    /** Returns if there is any word in the trie that starts with the given prefix. */

```

```

    bool startsWith(string prefix) {
        TrieNode* p = root;
        for (int i = 0; i < prefix.size(); i++){
            if (p) p = p->next[prefix[i] - 'a'];
            else return false;
        }
        return p != NULL;
    }
};

```

212. Word Search II +

```

class Solution {
public:
    struct TrieNode {
        TrieNode* child[26];
        string str;
        TrieNode() : str(""){
            for (auto &a : child) a = NULL; //must be "&a", not "a" here!!!
        }
    };
    struct Trie {
        TrieNode* root;
        Trie() : root(new TrieNode()) {}
        void insert(string s) {
            TrieNode* p = root;
            for (char c : s) {
                int i = c - 'a';
                if (!p->child[i]) {
                    p->child[i] = new TrieNode();
                }
                p = p->child[i];
            }
            p->str = s; //reached the leaf!!!
        }
    };
    vector<string> findWords(vector<vector<char>>& board, vector<string>& words) {
        vector<string> ret;
        if (words.size() == 0 || board.size() == 0 || board[0].size() == 0) {
            return ret;
        }
        Trie T;
        for (string a : words) {
            T.insert(a);
        }
        vector<vector<bool>> visited(board.size(), vector<bool>(board[0].size(), false));
        for (int i = 0; i < board.size(); i++) {
            for (int j = 0; j < board[0].size(); j++){
                if (T.root->child[board[i][j] - 'a']) { //won't continue if not first char
                    search(board, visited, T.root->child[board[i][j] - 'a'], i, j, ret);
                }
            }
        }
    }
};

```

```

    }
}
return ret;
}
void search(vector<vector<char>>& board, vector<vector<bool>>& visited, TrieNode* p,
int M, int N, vector<string>& ret) {
    if (!p->str.empty()){
        ret.push_back(p->str);
        p->str.clear();//Can't return! continue in case others have this str as prefix!
    }
    int d[][2] = {{0, 1}, {0, -1}, {-1, 0}, {1, 0}};
    visited[M][N] = true;
    for (auto &a : d){ //same as above. Essentially pointer
        int nM = a[0] + M, nN = a[1] + N;
        if (nM >= 0 && nM < board.size() && nN >= 0 && nN < board[0].size() &&
!visited[nM][nN] && p->child[board[nM][nN]-'a']) { //trie terminates search fast!
            search(board, visited, p->child[board[nM][nN]-'a'], nM, nN, ret);
        }
    }
    visited[M][N] = false;
}
};

```

X 211. Add and Search Word - Data structure design +++

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=274778>

```

class WordDictionary {
struct TrieNode{
    TrieNode* next[26];
    bool isKey;
    TrieNode() : isKey(false) {
        memset(next, NULL, sizeof(TrieNode*) * 26);
    }
};
public:
    /** Initialize your data structure here. */
    WordDictionary() {
        root = new TrieNode();
    }
    /** Adds a word into the data structure. */
    void addWord(string word) {
        TrieNode* run = root;
        for (char c : word) {
            if (!run->next[c - 'a']){
                run->next[c - 'a'] = new TrieNode();
            }
            run = run->next[c - 'a'];
        }
        run->isKey = true;
    }
}

```

```

    /** Returns if the word is in the data structure. A word could contain the dot
    character '.' to represent any one letter. */
    bool search(string word) {
        return query(word, root);
    }
private:
    TrieNode* root;
    bool query(string word, TrieNode* root) {
        TrieNode* run = root;
        for (int i = 0; i < word.length(); i++) {
            if (run && word[i] != '.') {
                run = run->next[word[i] - 'a'];
            }
            else if (run && word[i] == '.') {
                TrieNode* temp = run;
                for (int j = 0; j < 26; j++) {
                    run = temp->next[j];
                    if (query(word.substr(i + 1), run)) {
                        return true;
                    }
                }
            }
            else break;
        }
        return run && run->isKey;
    }
};

```

X 236. Lowest Common Ancestor of a Binary Tree +

```

TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
    if (!root || root == p || root == q) {
        return root;
    }
    TreeNode* left = lowestCommonAncestor(root->left, p, q);
    TreeNode* right = lowestCommonAncestor(root->right, p, q);
    if (left && right) {
        return root;
    }
    if (left) {
        return left;
    }
    if (right){
        return right;
    }
    return NULL;
}

```

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=280332>

没看懂这个变形:

<http://www.1point3acres.com/bbs/thread-292085-1-1.html>

X 282. Expression Add Operators

Similar to it, 给一个数字组成的字符串，可以在任意两个数字之间放加号或乘号，求可以得到的最大值。就是用divide + memorization解之

Linked List

2. Add Two Numbers

关键在于如何操作Pointer，比如new...

一年之前的流氓解法：

```
ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
    uint64_t L1_value = 0;
    uint64_t L2_value = 0;
    uint64_t SUM = 0;
    uint64_t i;
    for (i = 1; l1 != NULL; i*=10, l1 = l1->next) L1_value = L1_value + l1->val * i;
    for (i = 1; l2 != NULL; i*=10, l2 = l2->next) L2_value = L2_value + l2->val * i;
    SUM = L1_value + L2_value;
    ListNode *node = new ListNode(SUM%10);
    ListNode *head = node;
    ListNode *tail = node;
    for (i = 10; SUM/i > 0 ; i*=10) {
        uint64_t integer;
        integer = (SUM % (i * 10)) / i;
        tail->next = new ListNode(integer);
        tail = tail->next;
    }
    return head;
}
```

老老实实用pointer:

```
ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
    ListNode* Ret = new ListNode(0);
    ListNode* p = Ret;
    int c=0;
    int sum;
    while (l1 || l2){
        if (l1&&l2){
            sum = l1->val+l2->val+c;
            l1 = l1->next;
            l2 = l2->next;
        }
        else if (l1){
            sum = l1->val + c;
            l1 = l1->next;
        }
    }
}
```

```

        else{
            sum = l2->val + c;
            l2 = l2->next;
        }
        p->val = sum % 10;
        c = sum / 10;
        if (l1|| l2){ //already advanced l1 and l2!!!
            p->next = new ListNode(0);
            p=p->next;
        }
    }
    if (c != 0)    p ->next = new ListNode(c);
    return Ret;
}

```

114. Flatten Binary Tree to Linked List ++

```

void flatten(TreeNode* root) {
    while (root){
        if (root->left && root->right){
            TreeNode* t = root->left;
            while (t->right)    t = t->right;
            t->right = root->right;
        }
        if (root->left){
            root->right = root->left;
            root->left = NULL;
        }
        root = root->right;
    }
}

```

More difficult:

X Convert Binary Tree to circular doubly-linked List +

[http://jianlu.github.io/2016/11/08/BST to CDLL/](http://jianlu.github.io/2016/11/08/BST%20to%20CDLL/)

Closely related to 144:

√144. Binary Tree Preorder Traversal

<http://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/>

Recursive

```

vector<int> preorderTraversal(TreeNode* root) {
    vector<int> ret;
    if (!root) {
        return ret;
    }
    ret.push_back(root->val);
    preorderTraversal(root->left, ret);
}

```



```

        preorderTraversal(root->right, ret);
        return ret;
    }
    void preorderTraversal(TreeNode* root, vector<int>& ret) {
        if (!root) {
            return;
        }
        ret.push_back(root->val);
        preorderTraversal(root->left, ret);
        preorderTraversal(root->right, ret);
    }
}

```

Iterative

```

vector<int> preorderTraversal(TreeNode* root) {
    vector<int> ret;
    if (!root) return ret;
    stack<TreeNode*> my_stack;
    my_stack.push(root);
    while(!my_stack.empty()) {
        TreeNode* temp = my_stack.top();
        my_stack.pop();
        ret.push_back(temp->val);
        if (temp->right) {
            my_stack.push(temp->right);
        }
        if (temp->left) {
            my_stack.push(temp->left);
        }
    }
    return ret;
}

```

X 206. Reverse Linked List

```

ListNode* reverseList(ListNode* head) {
    ListNode* pre = NULL;
    ListNode* nextNode = NULL;
    while(head){
        nextNode = head->next;
        head->next = pre;
        pre = head;
        head = nextNode;
    }
    return pre;
}

```

234. Palindrome Linked List

```

class Solution {
public:

```

```

bool isPalindrome(ListNode* head) {
    if (!head || !head->next) return true;
    ListNode* slow = head;
    ListNode* fast = head;
    while (fast->next && fast->next->next){
        slow = slow->next;
        fast = fast->next->next;
    }
    //slow reach mid-point
    slow->next=reverseList(slow->next);
    slow = slow->next;
    while (slow){
        if (head->val!=slow->val) return false;
        head = head->next;
        slow = slow->next;
    }
    return true;
}

ListNode* reverseList(ListNode* head) {
    ListNode* pre=NULL;
    ListNode* next=NULL;
    while(head!=NULL){
        next=head->next;
        head->next=pre;
        pre=head;
        head=next;
    }
    return pre;
}
};

```

X 143. Reorder List ++

```

void reorderList(ListNode* head) {
    //1. break the List into two
    if (!head || !head->next) return;
    ListNode* p1=head;
    ListNode* p2=head->next;
    while(p2 && p2->next){
        p1 = p1->next;
        p2 = p2->next->next;
    }
    //2. reverse the second, now p1 is the middle
    ListNode* reverse = reverseList(p1->next);
    p1->next = NULL; //has to terminate p1!!!
    //3. merge
    for(p1 = head, p2 = reverse; p1; ){
        auto temp = p1->next;
        p1->next = p2;
        p1 = p1->next;
    }
}

```

```

        p2 = temp;
    }
    return;
}
ListNode* reverseList(ListNode* head) {
    ListNode* pre=NULL;
    ListNode* next=NULL;
    while(head!=NULL){
        next=head->next;
        head->next=pre;
        pre=head;
        head=next;
    }
    return pre;
}

```

341. Flatten Nested List Iterator +

```

class NestedIterator {
    stack<NestedInteger> nodes;
public:
    NestedIterator(vector<NestedInteger> &nestedList) {
        for (int i = nestedList.size() - 1; i >= 0; i--){
            nodes.push(nestedList[i]);
        }
    }
    int next() {
        int res = nodes.top().getInteger();
        nodes.pop();
        return res;
    }
    bool hasNext() { //always called before next(). So no problem next just returns the
first nodes.
        while(!nodes.empty()){
            NestedInteger curr = nodes.top();
            if (curr.isInteger()) return true;
            nodes.pop();
            vector<NestedInteger> currList = curr.getList();
            for(int j = currList.size() - 1; j >= 0; j --){
                nodes.push(currList[j]);
            }
        }
        return false;
    }
};

```

160. Intersection of Two Linked Lists +

```

ListNode *getIntersectionNode(ListNode *headA, ListNode *headB) {
    if (!headA || !headB) {

```

```

        return NULL;
    }
    ListNode* a = headA;
    ListNode* b = headB;
    int count = 0;
    while (a != b ){
        a = a->next;
        b = b->next;
        if (a == NULL){
            a = headB;
            count ++;
        }
        if (b == NULL){
            b = headA;
        }
        if (count == 2) {
            return NULL;
        }
    }
    return a;
}

```

X Interleave list of lists

([[1,2,3], [4,5], [6,7,8,9]] => [1,4,6,2,5,7,3,8,9]) follow up: 能否in-place

X 23. Merge k Sorted Lists +

Priority Queue:Space $O(k)$, Time $O(nk \log k)$

```

class Solution {
    struct compNode{ //define comparator to get min heap instead of default max
        bool operator () (ListNode* p, ListNode* q) const {
            return p->val > q->val;
        }
    };
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        ListNode* ret;
        priority_queue<ListNode*, vector<ListNode*>, compNode> pq;
        ListNode* dummy = new ListNode(0);
        ListNode* tail = dummy;
        //push the head of each list into priority_queue
        for (int i = 0; i < lists.size(); i++) {
            if (lists[i]) pq.push(lists[i]);
        }
        while (!pq.empty()) {
            tail->next = pq.top();
            tail = tail->next;
            pq.pop();
            if (tail->next) pq.push(tail->next);
        }
    }
}

```

```

    }
    return dummy->next;
}
};

```

Divide Conquer :Space $O(1)$, Time $O(nk \log k)$

```

class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        if (lists.empty()) return NULL;
        int end = lists.size() - 1;
        while (end > 0) {
            int begin = 0;
            while (begin < end) {
                lists[begin] = merge2Lists(lists[begin], lists[end]);
                begin++;
                end--;
            }
        }
        return lists[0];
    }
private:
    ListNode* merge2Lists(ListNode* p1, ListNode* p2){
        ListNode* dummy = new ListNode(0);
        ListNode* tail = dummy;
        while (p1 && p2) {
            if (p1->val < p2->val){
                tail->next = p1;
                p1 = p1->next;
            }
            else{
                tail->next = p2;
                p2 = p2->next;
            }
            tail = tail->next;
        }
        tail->next = p1 ? p1 : p2;
        return dummy->next;
    }
};

```

Math

168. Excel Sheet Column Title +

```

string convertToTitle(int n) {
    string ret = "";
    while (n > 0) {
        int m_mod = (n - 1) % 26;

```

```

        ret = char (m_mod + 'A') + ret;
        n = (n - 1) / 26;
    }
    return ret;
}

```

X 29. Divide Two Integers +

```

int divide(int dividend, int divisor) {
    if (!divisor || dividend == INT_MIN && divisor == -1) {
        return INT_MAX;
    }
    int sign = (dividend < 0) ^ (divisor < 0) ? -1 : 1;
    long long dvd = labs(dividend);
    long long dvs = labs(divisor);
    int ret = 0;
    while (dvd >= dvs) {
        long long temp = dvs, mutiple = 1;
        while (dvd >= (temp << 1) ) {
            temp <<= 1;
            mutiple <<= 1;
        }
        dvd -= temp;
        ret += mutiple;
    }
    return ret * sign;
}

```

273. Integer to English Words +

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=283641>

```

string numberToWords(int num) {
    if (num==0){
        return "Zero";
    }
    unordered_map<int, string> INT{
        {0, ""},
        {1, "One"},
        {2, "Two"},
        {3, "Three"},
        {4, "Four"},
        {5, "Five"},
        {6, "Six"},
        {7, "Seven"},
        {8, "Eight"},
        {9, "Nine"}
    };
    unordered_map<int, string> Teens{
        {10, "Ten"},
        {11, "Eleven"},

```

```

        {12, "Twelve"},
        {13, "Thirteen"},
        {14, "Fourteen"},
        {15, "Fifteen"},
        {16, "Sixteen"},
        {17, "Seventeen"},
        {18, "Eighteen"},
        {19, "Nineteen"}
    };

    unordered_map<int, string> Tens{
        {2, "Twenty"},
        {3, "Thirty"},
        {4, "Forty"},
        {5, "Fifty"},
        {6, "Sixty"},
        {7, "Seventy"},
        {8, "Eighty"},
        {9, "Ninety"}
    };

    unordered_map<int, string> THS{
        {3, "Thousand"},
        {6, "Million"},
        {9, "Billion"},
    };

    string result = "";
    int d = 0;
    while (num != 0){
        if (d % 3 == 1){
            if(num % 10 != 0 && num % 10 != 1){
                result = Tens[num % 10] + ' ' + result;
            }
        }
        else if(d%3==2){
            if(num%10!=0){
                result = INT[num%10]+' '+'Hundred "+result;
            }
        }
        else{
            if(d==0){ //the last digit
                if ((num%100)/10==1){ //10-19
                    result = Teens[num%100]+' '+result;
                }
                else{
                    if(num%10!=0){ //last digit not zero
                        result = INT[num%10]+' '+result;
                    }
                }
            }
            else{ //d = 3, 6, 9, thousand, million or billion digit
                if ((num%100)/10==1){
                    result = Teens[num%100]+' '+THS[d]+' '+result;
                }
            }
        }
        num /= 10;
        d++;
    }
    return result;
}

```

```

    }
    else{
        if(num%10!=0){
            result = INT[num%10]+' '+THS[d]+' '+result;
        }
        else if(num%1000!=0){
            result = THS[d]+' '+result;
        }
    }
}
}
d++;
//cout<<d<<"\n";
num=num/10;
}

result.erase(result.end()-1, result.end());
return result;
}

```

Follow up: minimize number of swap (or assignment)

67. Add Binary ++

```

string addBinary(string a, string b) {
    string ret="";
    int alen = a.length();
    int blen = b.length();
    int c=0, s=0;
    for (int i = 1; i <= alen || i <= blen || c>0; i++){ //begin with 1
        if (i <= alen) c += a[alen-i] - '0';
        if (i <= blen) c += b[blen-i] - '0';
        s = c % 2;
        c = c / 2;
        ret = char( s + '0') + ret; //1) not "+=", append before! 2) convert to char!
    }
    return ret;
}

```

Follow up: 支持不同的进制

415. Add Strings +

```

string addStrings(string num1, string num2) {
    string ret = "";
    int l1 = num1.length();
    int l2 = num2.length();
    if (l1==0 || l2==0) return ret;
    int c = 0;
    int s;
    for (int i=1; i<=l1 || i<=l2; i++){
        s=c;

```



```

        if (i<=l1) s+= num1[l1-i]-'0';
        if (i<=l2) s+= num2[l2-i]-'0';
        c = s/10;
        s = s%10;
        ret = char(s+'0')+ret;
    }
    if (c!=0)        ret = '1'+ret;
    return ret;
}

```

X 224. Basic Calculator

```

int calculate(string s) {
    if (s.length()==0) return 0;
    stack<int> nums, ops;
    int digit, num=0, sign=1;
    int ret=0;
    for (char c:s){
        if (isdigit(c)){
            digit = c-'0';
            num = num*10 + digit;
        }
        else{
            ret += sign*num;
            num=0; //num need to be cleared!!!
            if (c=='+') sign = 1;
            else if (c=='-') sign = -1;
            else if (c=='('){
                nums.push(ret);
                ops.push(sign);
                ret = 0;
                sign = 1;
            }
            else if (c==')'){
                ret = nums.top()+ops.top()*ret;
                nums.pop();
                ops.pop();
            }
        }
    }
    ret += num*sign;
    return ret;
}

```

X 227. Basic Calculator II

```

int calculate(string s) {
    stack<int> myStack;
    char sign = '+';
    int res = 0, tmp = 0;

```

```

    for (unsigned int i = 0; i < s.size(); i++) {
        if (isdigit(s[i]))
            tmp = 10*tmp + s[i] - '0';
        if (!isdigit(s[i]) && !isspace(s[i]) || i == s.size() - 1) {
            if (sign == '-')
                myStack.push(-tmp);
            else if (sign == '+')
                myStack.push(tmp);
            else {
                int num;
                if (sign == '*')
                    num = myStack.top()*tmp;
                else
                    num = myStack.top()/tmp;
                myStack.pop();
                myStack.push(num);
            }
            sign = s[i];
            tmp = 0;
        }
    }
    while (!myStack.empty()) {
        res += myStack.top();
        myStack.pop();
    }
    return res;
}

```

X 50. POW

```

double myPow(double x, int n) {
    if (n == 0) return 1;
    if (n < 0){
        if (n == INT_MIN){ //ERROR: corner case here!
            n = INT_MAX;
            x = (1 / x) * (1 / x);
        }
        else{
            n = - n;
            x = 1 / x;
        }
    }
    return (n%2==0) ? myPow(x*x, n/2):x*myPow(x*x, n/2);
}

```

9. Palindrome Number

```

bool isPalindrome(int x) {
    if(x < 0 || (x != 0 && x % 10 == 0)) return false;
    int sum = 0;

```

```

while(x > sum) {
    sum = sum * 10 + x % 10;
    x = x / 10;
}
return x == sum || x == sum/10;
}

```

461. Hamming Distance +

Fast and tricky

```

int hammingDistance(int x, int y) {
    int z = x ^ y;
    int count = 0;
    while (z){
        z &= (z - 1);
        count ++;
    }
    return count;
}

```

Slow, iterate all digits

```

int hammingDistance(int x, int y) {
    int ret = 0;
    int x_xor_y = x ^ y;
    while (x_xor_y > 0) {
        ret += x_xor_y & 0x1;
        x_xor_y = x_xor_y >> 1;
    }
    return ret;
}

```

477. Total Hamming Distance +

```

int totalHammingDistance(vector<int>& nums) {
    int cnt[32] = {0};
    int n = nums.size();
    int ret = 0;
    for (int num : nums){
        int i = 0;
        while(num > 0) {
            cnt[i++] += num & 0x1;
            num=num >> 1;
        }
    }
    for (int j = 0; j < 32; j++){
        ret += cnt[j] * (n - cnt[j]);
    }
    return ret;
}

```

```
}
```

DFS/Backtracking/BFS

? 新题: 几度好友

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=291190&ctid=519>

用户id是用整数型的, 又一个api是可以获得该用户的所有朋友id, 如何得到用户a 和b之间的是几度好友需要 coding, follow up, 空间复杂度, 如何优化memory。当时答的用BST和双向BST, 跪了。

257. Binary Tree Paths ++

```
class Solution {
public:
    vector<string> binaryTreePaths(TreeNode* root) {
        vector<string> ret;
        if (!root){
            return ret;
        }
        printPaths(root, ret, to_string(root->val));
        return ret;
    }
private:
    void printPaths(TreeNode* root, vector<string>& ret, string t) {
        if (!root->left && !root->right) {
            ret.push_back(t);
            return;
        }
        if (root->left){
            printPaths(root->left, ret, t + "->" + to_string(root->left->val));
        }
        if (root->right){
            printPaths(root->right, ret, t + "->" + to_string(root->right->val));
        }
        return;
    }
};
```

X 536. Construct Binary Tree from String ++

My first trial, not very clean

```
TreeNode* str2tree(string s) {
    if (s == "") return NULL;
    int i=0;
    int len = s.length();
    while(s[i] != '(' && i < len) {
        i++;
    }
```

```

    }
    TreeNode* ret = new TreeNode(stoi(s.substr(0, i)));
    if (i == len) {
        return ret;
    }
    int balance = 1;
    int j = i + 1;
    while (balance != 0 && j < len) { //不能无限循环下去!
        if (s[j] == '('){
            balance ++;
        }
        if (s[j] == ')') {
            balance --;
        }
        j++;
    }
    ret->left = str2tree(s.substr(i + 1, j - i - 2));
    if (j != len) {
        ret->right = str2tree(s.substr(j + 1, len - j - 2)); //各种想不明白...
    }
    return ret;
}

```

X 247. Strobogrammatic Number II +

First trial: works but slow...

```

class Solution {
public:
    vector<string> findStrobogrammatic(int n) {
        vector<int> s_lut = {0, 1, 8};
        vector<string> ret;
        if (n % 2 == 1){
            for (int i : s_lut){
                dfs(ret, to_string(i), n / 2);
            }
            return ret;
        }
        dfs(ret, "", n / 2);
        return ret;
    }
private:
    vector<pair<int, int>> lut = {{0, 0}, {1, 1}, {6, 9}, {8, 8}, {9, 6}};
    vector<pair<int, int>> lut_nz = {{1, 1}, {6, 9}, {8, 8}, {9, 6}};

    void dfs(vector<string>& ret, string s, int cnt){
        if (cnt == 0){
            ret.push_back(s);
            return;
        }
        if (cnt == 1){

```

```

        for (auto P : lut_nz){
            dfs(ret, to_string(P.first) + s + to_string(P.second), cnt - 1);
        }
        return;
    }
    for (auto P : lut){
        dfs(ret, to_string(P.first) + s + to_string(P.second), cnt - 1);
    }
    return;
}
};

```

79 Word Search +++

```

class Solution {
private:
    bool search(vector<vector<char>>& board, vector<vector<bool>>& track, string word, int
offset, int i, int j) {
        if (board[i][j] != word[offset])    return false;
        if (offset == word.length() - 1)    return true; //didn't thought of this!!! This
is required to terminate the dfs!!!
        track[i][j] = true;
        if (i > 0 && !track[i - 1][j] && search(board, track, word, offset + 1, i - 1, j)){
            return true;
        }
        if (i < board.size() - 1 && !track[i + 1][j] && search(board, track, word, offset +
1, i + 1, j)){
            return true;
        }
        if (j < board[0].size() - 1 && !track[i][j + 1] && search(board, track, word,
offset + 1, i, j + 1)){
            return true;
        }
        if (j > 0 && !track[i][j - 1] && search(board, track, word, offset + 1, i, j -
1)){
            return true;
        }
        track[i][j] = false; 经典的backtracking!!!
        return false;
    }
public:
    bool exist(vector<vector<char>>& board, string word) {
        int r = board.size();
        if (r == 0) return false;
        int c = board[0].size();
        if (c == 0) return false;

        vector<vector<bool>> track;
        track.resize(r);
        for (int i = 0; i < r; i++){
            track[i].resize(c);

```

```

        for (int j = 0; j < c; j++){
            track[i][j] = false;
        }
    } 用track来防止重复利用!!!
    for (int i = 0; i < r; i++){
        for (int j = 0; j < c; j++){
            if (search(board, track, word, 0, i, j)){
                return true;
            }
        }
    }
    return false;
}
};

```

A shorter way of “search”:

private:

```

bool search(vector<vector<char>>& board, vector<vector<bool>>& track, string word, int
offset, int i, int j) {
    if (board[i][j] != word[offset])    return false;
    if (offset == word.length() - 1)    return true; //didn't thought of this!!!
    track[i][j] = true;
    int d[][2] = {{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
    for (auto &a : d) {
        int ni = i + a[0], nj = j + a[1];
        if (ni >= 0 && ni < board.size() && nj >= 0 && nj < board[0].size() &&
!track[ni][nj]&& search(board, track, word, offset + 1, ni, nj)) {
            return true;
        }
    }
    track[i][j] = false;
    return false;
}

```

210. Course Schedule II +

BFS

```

vector<int> findOrder(int numCourses, vector<pair<int, int>>& prerequisites) {
    vector<int> ret;
    vector<unordered_set<int>> dep = vector<unordered_set<int>>(numCourses,
unordered_set<int>());
    vector<unordered_set<int>> remove = vector<unordered_set<int>>(numCourses,
unordered_set<int>());
    for(pair<int, int> it : prerequisites){
        dep[it.first].insert(it.second);
        remove[it.second].insert(it.first);
    }
    vector<int> todo;
    for(int i = 0; i < numCourses; i++) {

```

```

        if(dep[i].empty()) { //i has no dependency
            todo.push_back(i);
        }
    }
    while(!todo.empty()){
        int temp = todo.back();
        ret.push_back(temp);
        todo.pop_back();
        for(int it : remove[temp]){
            dep[it].erase(temp);
            if(dep[it].empty()){
                todo.push_back(it);
            }
        }
    }
    if(ret.size() != numCourses){
        ret=vector<int>();
    }
    return ret;
}

```

46. Permutations (no duplicates!)

Solve with DFS? Looks like generic backtracking instead of DFS as it's not tree!

```

class Solution {
public:
    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> result;
        dfs(nums, 0, result);
        return result;
    }
    //solve with backtracking
private:
    void dfs(vector<int>& nums, int begin, vector<vector<int>>& result){
        if (begin>=nums.size()){ //== should be just fine
            result.push_back(nums);
            return;
        }
        for (int i = begin; i<nums.size(); i++){
            swap(nums[i], nums[begin]);
            dfs(nums, begin+1, result); //ERROR: not dfs(nums, i, result)! it's begin+1!!!
            //The next position to swap.
            swap(nums[i], nums[begin]);
        }
    }
};

```

47. Permutations II (with duplicates!)

```

class Solution {

```



```

public:
    vector<vector<int> > permuteUnique(vector<int> &num) {
        vector<vector<int> > result;
        permuteRecursive(num, 0, result);
        return result;
    }

    void permuteRecursive(vector<int> &num, int begin, vector<vector<int>> &result) {
        if (begin >= num.size()) {
            result.push_back(num);
            return;
        }

        // detect duplicate
        unordered_set<int> set;
        for (int i = begin; i < num.size(); i++) {
            if (set.count(num[i]) > 0)
                continue; //Haven't fully understand how it works!!!
            set.insert(num[i]);

            swap(num[begin], num[i]);
            permuteRecursive(num, begin + 1, result);
            swap(num[begin], num[i]);
        }
    }
};

```

543. Diameter of Binary Tree +

```

class Solution {
    int diameter=0;
public:
    int diameterOfBinaryTree(TreeNode* root) {
        depth(root);
        return diameter;
    }
private:
    int depth(TreeNode* root){
        if (!root) return 0;
        int left_depth = depth(root->left);
        int right_depth = depth(root->right);
        diameter = diameter < left_depth + right_depth ? left_depth + right_depth :
diameter;
        return max(left_depth, right_depth)+1;
    }
};

```

124. Binary Tree Maximum Path Sum

```

class Solution {

```

```

int max_path_sum = INT_MIN;
public:
    int maxPathSum(TreeNode* root) {
        dfs(root);
        return max_path_sum;
    }
private:
    int dfs(TreeNode* root){
        if (!root) return 0;
        int left = dfs(root->left);
        int right = dfs(root->right);
        left = left < 0 ? 0 : left;
        right = right < 0 ? 0 : right;
        //if (max_path_sum< max(left, right)+root->val) max_path_sum=max(left,
right)+root->val;
        if (max_path_sum < left + right + root->val){
            Max_path_sum = left + right + root->val;
        }
        return max(left, right) + root->val;
    }
};

```

102. Binary Tree Level Order Traversal +

```

vector<vector<int>> levelOrder(TreeNode* root) {
    vector<vector<int>> ret;
    if (!root) return ret;
    queue<TreeNode*> q;
    q.push(root);
    while(!q.empty()){
        vector<int> row;
        int sz = q.size();
        for (int i = 0; i < sz; i++) {
            row.push_back(q.front()->val);
            if(q.front()->left) q.push(q.front()->left);
            if(q.front()->right) q.push(q.front()->right);
            q.pop();
        }
        ret.push_back(row);
    }
    return ret;
}

```

314. Binary Tree Vertical Order Traversal ++

```

unordered_map<int, vector<int>> res;
vector<vector<int>> verticalOrder(TreeNode* root) {
    queue<pair<TreeNode*, int>> todo;
    vector<vector<int>> ret;
    todo.push(pair<TreeNode*, int> (root, 0));
}

```

```

while(!todo.empty()){
    pair<TreeNode*, int> temp = todo.front();
    if(temp.first){
        res[temp.second].push_back(temp.first->val);
        todo.push(pair<TreeNode*, int>(temp.first->left, temp.second - 1));
        todo.push(pair<TreeNode*, int>(temp.first->right, temp.second + 1));
    }
    todo.pop();
}
int min = INT_MAX;
int max = INT_MIN;
for(auto it : res){
    min = min > it.first ? it.first : min;
    max = max < it.first ? it.first : max;
}
for (int i = min; i <= max; i++) {
    ret.push_back(res[i]);
}
return ret;
}

```

X 301. Remove Invalid Parentheses ++++

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=285339&>

<http://rainykat.blogspot.com/2017/01/leetcodef-301-remove-invalid-parentheses.html>

```

class Solution {
private:
    bool isValid(string s){
        int sum = 0;
        for (char c:s){
            sum += c=='(';
            sum -= c==')';
            if (sum < 0)    return false;
        }
        return sum==0;
    }
    void dfs(string s, int beg, int num1, int num2, vector<string> &ret){
        if (num1==0 && num2==0){
            if (isValid(s)) ret.push_back(s);
        }
        else{
            for (int i = beg; i<s.size(); i++){
                string tmp = s;
                if (num1>0 && num2==0 && tmp[i]=='('){
                    if(i == beg || tmp[i] != tmp[i-1]){
                        tmp.erase(i,1); //ERROR: didn't used erase correct, used erase(i)
instead of erase(i, 1)
                        dfs(tmp, i, num1-1, num2, ret);
                    }
                }
            }
        }
    }
}

```

```

        //else if(num1==0 && num2>0 && tmp[i]==''){ ERROR. can't test num1==0
here! it checks the case like )*(, wrong order. ( can't cancel )!!!
        else if(num2>0 && tmp[i]==''){
            if(i==beg || tmp[i]!=tmp[i-1]){
                tmp.erase(i,1); //same above
                dfs(tmp, i, num1, num2-1, ret);
            }
        }
    }
}
}
public:
    vector<string> removeInvalidParentheses(string s) {
        vector<string> ret;
        int num1 = 0;
        int num2 = 0;
        for (char c:s){
            num1 += c=='(';
            if (num1==0){
                num2 += c==')';
            }
            else{
                num1 -= c==')';
            }
        }
        dfs(s, 0, num1, num2, ret);
        return ret;
    }
};

```

X 17. Letter Combinations of a Phone Number ++++

<http://www.1point3acres.com/bbs/thread-266070-1-1.html>

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=222632>

Iterative

```

    vector<string> letterCombinations(string digits) {
        vector<string> result;
        if (digits.size() == 0){
            return result; //call a constructor to exit
        }
        result.push_back("");
        static const vector<string> table = {"", "", "abc", "def", "ghi", "jkl", "mno",
"pqrs", "tuv", "wxyz"};
        for (int i = 0; i < digits.size(); i++) {
            int index = digits[i] - '0';
            if (index < 0 || index > 9){
                return result;
            }
            const string& candidate = table[index];

```

```

        vector<string> tmp;
        for (int j = 0; j < candidate.size(); j++){ //loop through all possible letters
in this candidate
            for (int k = 0; k < result.size(); k++){ //add the letter to all current
results
                tmp.push_back(result[k] + candidate[j]);
            }
        }
        result.swap(tmp);
    }
    return result;
}

```

Recursive/Backtracking

```

class Solution {
private:
    const vector<string> table = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv",
"wxzyz"};
    void helperComb(string prefix, string& digits, vector<string>& result, int offset){
        if (offset >= digits.length() ){
            result.push_back(prefix);
            return;
        }
        string letters = table[digits[offset] - '0'];
        for (int i = 0; i < letters.size(); i++) {
            helperComb(prefix + letters[i], digits, result, offset + 1);
        }
    }
public:
    vector<string> letterCombinations(string digits) {
        vector<string> result;
        if (digits.size() == 0){
            return result;
        }
        helperComb("", digits, result, 0);
        return result;
    }
};

```

133. Clone Graph ++

Follow up:

讨论图的存储方式，如果图太大了怎么分db来存

BFS:

```

UndirectedGraphNode *cloneGraph(UndirectedGraphNode *node) {
    unordered_map<UndirectedGraphNode*, UndirectedGraphNode*> mp;
    if (!node) return NULL;
    UndirectedGraphNode* copy = new UndirectedGraphNode(node->label);

```

```

mp[node] = copy;
queue<UndirectedGraphNode*> toVisit;
toVisit.push(node);
while (!toVisit.empty()) {
    UndirectedGraphNode* cur = toVisit.front();
    toVisit.pop();
    for (UndirectedGraphNode* neigh : cur->neighbors) {
        if (mp.find(neigh) == mp.end()){
            mp[neigh] = new UndirectedGraphNode(neigh->label);
            toVisit.push(neigh); //push only unvisited neighbours !!!!
        }
        mp[cur]->neighbors.push_back(mp[neigh]);
    }
}
return copy;
}

```

DFS

```

class Solution {
    unordered_map<UndirectedGraphNode*, UndirectedGraphNode*> mp;
public:
    UndirectedGraphNode *cloneGraph(UndirectedGraphNode *node) {
        if (!node) return NULL;
        if (mp.find(node) == mp.end()){
            mp[node] = new UndirectedGraphNode(node->label);
            for (UndirectedGraphNode* neigh : node->neighbors)
                mp[node]->neighbors.push_back(cloneGraph(neigh));
        }
        return mp[node];
    }
};

```

X 77. Combinations

Backtracking

```

class Solution {
public:
    vector<vector<int>> combine(int n, int k) {
        vector<vector<int>> ret;
        if (n < k) return ret;
        vector<int> temp;
        combine(ret, temp, 0, n, k);
        return ret;
    }

    void combine(vector<vector<int>>&res, vector<int>& temp, int start, int n , int k){
        if(temp.size() == k){
            res.push_back(temp);
            return;
        }
    }
};

```

```

        for(int i = start; i < n; i++) {
            temp.push_back(i + 1);
            combine(res, temp, i + 1, n, k);
            temp.pop_back();
        }
    }
};

```

X Iterative

78. Subsets ++

Recursive

```

class Solution {
public:
    vector<vector<int>> subsets(vector<int>& nums) {
        vector<vector<int>> ret;
        vector<int> sub;
        helperSubset(ret, nums, sub, 0);
        return ret;
    }
    void helperSubset(vector<vector<int>>& ret, vector<int>& nums, vector<int>& sub, int
pos) {
        //if (pos == nums.size()) return; 不能有这句！！需要下面一句把sub加到ret里面！
        ret.push_back(sub);
        for (int i = pos; i < nums.size(); i++){
            sub.push_back(nums[i]);
            helperSubset(ret, nums, sub, i + 1);
            sub.pop_back();
        }
    }
};

```

Iterative

```

class Solution {
public:
    vector<vector<int>> subsets(vector<int>& nums) {
        vector<vector<int>> ret;
        ret.push_back(vector<int>());
        for (int num : nums){
            int sz = ret.size();
            for (int i = 0; i < sz; i++){
                vector<int> new_ele = ret[i]; //复制现有的所以 再把当前数字放上
                new_ele.push_back(num);
                ret.push_back(new_ele);
            }
        }
        return ret;
    }
};

```

```
};
```

90. Subsets II +

```
class Solution {
public:
    vector<vector<int>> subsetsWithDup(vector<int>& nums) {
        vector<vector<int>> ret;
        if (nums.size() == 0) return ret;
        sort(nums.begin(), nums.end());
        vector<int> sub;
        helperSubset(ret, sub, nums, 0);
        return ret;
    }
private:
    void helperSubset(vector<vector<int>>& ret, vector<int>& sub, vector<int>& nums, int
pos){
        ret.push_back(sub);
        for (int i = pos; i < nums.size(); i++) {
            if (i != pos && nums[i] == nums[i - 1]) {
                continue;
            }
            sub.push_back(nums[i]);
            helperSubset(ret, sub, nums, i + 1);
            sub.pop_back();
        }
    }
};
```

Follow up: No backtracking! Iterative and use multiset to handle duplicates?

39. Combination Sum

```
class Solution {
private:
    void combinationSum(vector<int>& candidates, int target, vector<vector<int>>& ret,
vector<int>& combination, int begin) {
        if (target == 0){
            ret.push_back(combination);
            return;
        }
        for (int i = begin; i < candidates.size() && target >= candidates[i]; i++) {
            combination.push_back(candidates[i]);
            combinationSum(candidates, target - candidates[i], ret, combination, i);This
allows repeated numbers!!!
            combination.pop_back();
        }
    }
public:
    vector<vector<int>> combinationSum(vector<int>& candidates, int target) {
        std::sort(candidates.begin(), candidates.end());
```



```

        vector<vector<int>> ret;
        vector<int> combination;
        combinationSum(candidates, target, ret, combination, 0);
        return ret;
    }
};

```

X 127. Word Ladder

为什么不能DFS!

Not completely correct...

```

class Solution {
public:
    int ladderLength(string beginWord, string endWord, vector<string>& wordList) {
        unordered_set<string> wordDict;
        for (string s : wordList) {
            wordDict.insert(s);
        }
        if (wordDict.find(endWord) == wordDict.end()) {
            return 0;
        }
        // wordDict.insert(endWord);
        queue<string> toVisit;
        addNextWords(beginWord, wordDict, toVisit);
        int dist = 2;
        while (!toVisit.empty()) {
            int sz = toVisit.size();
            while (sz--) {
                string word = toVisit.front();
                toVisit.pop();
                if (word == endWord) return dist;
                addNextWords(word, wordDict, toVisit);
            }
            dist++;
        }
        return 0;
    }

    void addNextWords(string word, unordered_set<string> wordDict, queue<string>& toVisit) {
        wordDict.erase(word);
        for (int p = 0; p < word.size(); p++) {
            char letter = word[p];
            for (int k = 0; k < 26; k++) {
                word[p] = 'a' + k;
                if (wordDict.find(word) != wordDict.end()) {
                    toVisit.push(word);
                    wordDict.erase(word);
                }
            }
        }
    }
};

```

```

    }
    word[p] = letter;
  }
}
};

```

51. N-Queens

新题

parse HTML and build a DOM tree

Josephus ring

Stack

X 71. Simplify Path

```

string simplifyPath(string path) {
    string res, tmp;
    stack<string> stk;
    stringstream ss(path);
    while(getline(ss, tmp, '/')) {
        if (tmp == "" or tmp == ".") continue;
        if (tmp == ".." and !stk.empty()) stk.pop();
        else if (tmp != "..") stk.push(tmp);
    }
    while(!stk.empty()) {
        res = res == "" ? stk.top() : stk.top() + "/" + res;
        stk.pop();
    }
    return res.empty() ? "/" : "/" + res;
}

```

Data Structure Design

348. Design Tic-Tac-Toe +

```

class TicTacToe {
private:

```

```

    int total;
    vector<int> row_judge;
    vector<int> col_judge;
    int diag, adiag;
public:
    /** Initialize your data structure here. */
    TicTacToe(int n) {
        total = n;
        for (int i = 0; i < n; i++) {
            row_judge.push_back(0);
            col_judge.push_back(0);
        }
        diag = 0;
        adiag = 0;
    }
    /** Player {player} makes a move at ({row}, {col}).
    @param row The row of the board.
    @param col The column of the board.
    @param player The player, can be either 1 or 2.
    @return The current winning condition, can be either:
            0: No one wins.
            1: Player 1 wins.
            2: Player 2 wins. */
    int move(int row, int col, int player) {
        int multiplier = player == 1 ? 1 : -1;
        row_judge[row] += multiplier;
        col_judge[col] += multiplier;
        diag += row == col ? multiplier : 0;
        adiag += row + col + 1 == total ? multiplier : 0;
        if (abs(row_judge[row]) == total || abs(col_judge[col]) == total || abs(diag) ==
total || abs(adiag) == total) {
            return player;
        }
        return 0;
    }
};

```

略变形，设计数据结构，return的类型是boolean，玩家一个是'W'，一个是'Z'，要求考虑：第一，如果重复了之前的坐标怎么处理，第二，不能一个玩家连续走两次。他描述问题用了5分钟，20分钟写代码问follow-up。

X 146. LRU Cache

```

class LRUCache {
    size_t m_capacity;
    unordered_map<int, list<pair<int, int>>::iterator> m_map;
    list<pair<int, int>> m_list;
public:
    LRUCache(int capacity) {
        m_capacity = capacity;
    }
};

```

```

}
int get(int key) {
    auto found_iter = m_map.find(key);
    if (found_iter == m_map.end()) {
        return -1;
    }
    m_list.splice(m_list.begin(), m_list, found_iter->second);
    return found_iter->second->second;
}
void put(int key, int value) {
    auto found_iter = m_map.find(key);
    if (found_iter != m_map.end()) {
        m_list.splice(m_list.begin(), m_list, found_iter->second);
        found_iter->second->second = value;
        return;
    }
    if (m_map.size() == m_capacity) {
        int key_to_del = m_list.back().first;
        m_list.pop_back();
        m_map.erase(key_to_del);
    }
    m_list.emplace_front(key, value);
    m_map[key] = m_list.begin();
}
};

```

Give weights, return characters as

<http://www.1point3acres.com/bbs/thread-189034-1-1.html>

- Method 1:
 - Create an array, which contains the words. For example, if the input is words = ["a", "b", "c"], weights = [1, 2, 3], the array should arr = {"a", "b", "b", "c", "c", "c"};
 - Generate a random number mod by the arr.size() as the index, return the word in that index.
 - Time: O(1), Space: O(N), where N is the size of generated array
 - Method 2:
 - calculate an array, where sums[i] is the sum of weights[0..i]. For the same example, sums = {1, 3, 6}.
 - Generate a random number: index = rand() % sums.back() + 1, find the first element in sums that is larger or equal to index using binary search, say the element is sums[j]
 - Return words[j], note that the index j is the same with sums[j].
- Time: O(MlogM), Space: O(M), where M is the size of input arrays.

System Design

设计instagram

534. Design TinyURL +

design messengers's online/offline status.

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=289190>

memcache + hashtable

POI+

网页event售票系统

设计脸书search

<http://www.1point3acres.com/bbs/thread-291061-1-1.html>

设计一个完整的search功能，能存，能读。比如你搜索一个key word，返回post，返回人，返回文章。然后就是问怎么存，怎么读，怎么优化读

写了后端实现到前端读取，什么aggregation, cache, elasticsearch, schema, content access level都答了，但是最后面试官问：如果你从trie里拿的结果太多的posts了怎么办？我想了半天说rank top k然后只返回那些top的。对方不置可否，反正就是感觉没达到点子上的感觉。

如果只是英语的话，应该提取keyword 然后建立invert index，然后对于keyword做shading，然后上面加一个aggregation service来聚合结果。我看不出这为啥一定要说手机app的呀，和网站搜索差别不大呀 我感觉那个follow up结果太多 返回top k应该是正确答案呀，或者根据user习惯来进行定制什么的

Status更新系统

可以write status（比如：我好开心），可以search status，search可以用and / or（比如search，天气 and 汽车，只有“天气不好我要坐汽车”这条返回。天气 or 汽车，那就得返回“天气不错”和“汽车坏了”两条）讨论了如何存，用什么数据库，如何search，分析了QPS，讨论如何scale, sharding，如何建index（我把inverted index全程说成了reversed index, 面完才意识到，大慌，感觉药丸。但还好这个他没当回事）。

Design a simple message system.

是个国男大哥，很严肃，但估计也没有刁难我。就是基本的怎么发信息，怎么收信息，然后怎么通知receiver，如果有新的消息了。