1. **Minimum window subsequence**
    a. description
    Given strings S and T, find the minimum (contiguous) substring W of S, so that T is a subsequence of W.
    dp[i][j]

		0	1	2	3	4	5	6	7	8	9
			a	**b**	**c**	**d**	**e**	**b**	**d**	**d**	**e**
0		1	2	3	4	5	6	7	8	9	10
1	b	0	0	2	2	2	2	6	6	6	6
2	b	0	0	0	0	0	0	2	2	2	2
3	e	0	0	0	0	0	0	0	0	0	2

1 2 3 4 5 6 7 8 9 10

0 0 3 4 5 6 7 8 9 10

length = 9 - 2 + 1 = 8

statIndex[i][j] represents the startIndex to make the substring end with S(j-1) be valid to T(i-1)
ifS(j-1) == T(i-1) startIndex[i][j] = startIndex[i-1][j-1]
else startIndex[i][j] = startIndex[i-1][j]

Use temp array to optimize space to be O(n)
Time = O(mn)

2. **get prime factors**
    a. description
    15 = 3 * 5
    20 = 2^2 * 5
    20 = 5 * 2^2
    b. solution
    15 / 3 = 5
    5 / 3 x
    5 / 5 = 1

3 * 5
Time = O(n^3) Space = O(1)

3. **valid number (2018.1.28)**
    a. description
       no exp, but all others and space
    b. solution
       123 true
       a123 flase
       +123 true
       1+2 flase
       - 123 true
       0.1 true
       .1 true
       1.1 true
        1.05. flase
       1 5 true?
       1e6 true
       0e-6 flase
       only leading and space

       1. trim spaces
       2. if digit: always right
       3. if e: no previous e, must after number, must have number after
       4. if +: i == 0 || i-1 == e
       5. if . : no previous dots, no previous e, must have number after

4. **add(number, number) (2018.1.28)**
    a. description
       add two numbers without +
    b. solution
       idea:
       bit operation

        1. Sum of two bits can be obtained by performing XOR (^) of the two bits.
        2. Carry bit can be obtained by performing AND (&) of two bits.
        3. https://www.geeksforgeeks.org/add-two-numbers-without-using-arithmetic-operators/

5. **range sum query (LC 304) (2017.09.02)**
    a. description
    b. solution
        i. idea:
            1. brute force, for each start to end ,sum O(n^2) →there are n^4 rectangles in the matrixs so O(n^6)
            2. using dp to optimize,
            3. dp[i][j] represents the subsum from[0][0] as top left and [i][j] as bot right
            4. target = dp[r2][c2] + dp[r1-1][c1-1] - dp[r1-1][c] - dp[r1][c1-1]
            5. Time = O(n^2)
6. **group anagrams**
    a. description
        i. Follow up - Print lists according to the input order
            1. use a list to store the key
            2. if need to maintain inside order by abs, use TreeSet
        ii. Time and Space
            1. O(mlogm * n)
            2. Space(n)
7. **findpeak**
    a. description
    b. solution
        i. idea
            1. O(n) check if left or right is smaller than it
            2. corner case : i == 0, left = MIN_VALUE, i == array.length - 1 , right = MIN_VALUE
            3. consider binary search, logn
8. **all longest subsequences**
    a. description

1	3	4	3	4	5
<1>	<1,3>	<1,3,4>	<1,3>	<1,3,4>	<1,3,4,5>
				<1,3,4>	<1,3,4,5>
					<1,3,4,5>
					<1,3,4,5>
1	2	3	2	3	4

follow up : longest subsequence , nlogn

    b. solution
        i. idea
            1. brute force 2^n find all ascending then find max

            2. dp[i] = Math.max[dp[j] + 1, dp[i]] if (dp[i] > dp[j]) for (j < i)
               else dp[i] = 1
            3. meantime, maintain list of (curLongest result), for (j = i, j--){
                   only update the longest result;

9. **Calendar AddDate (2018-10-5)**
    a. description
        i. 日历加日期
    b. solution
        i. idea
            1. 闰年： % 4 , ! %100, % 400
            2. two

10. **merge k sorted arrays**
    a. description
    b. solution
        i. idea
            1. minHeap, each time add the element to array, kn*logk
            2. binary merge, nk + n k /2 +

11. **check 一个graph是不是bipartite**
    a. description
        i. no self containing, no containing any node twice
        ii. return boolean
        iii. any separate nodes?
    b. solution
        i. idea:
            1.

```
            1 ---------- 2
            |            |
            3 ----------- 4
```

            2. Using BFS,
                a. 1 set 0, then 2(1), 3(1), put into q
                b. 2(1) , check others if (1) return false, if (-1) set 0 ,put into q
                c. 3 (1), check others if (1) return false, if (-1) set 0, put into q
                d. 4(0)
                   Time = O(V + E)
            3. Using DFS
                a. 1 set 0, then 2(1), then 4(0), then 3 (1), if ending check !=
                   1- starting , return false
                b. then back to 3()
                c. Time = O(V + E)

12. **unfair coin,怎么disign一个process可以返回同样几率的0和1**
13. **singleton设计模式Word break**
    a. description

            i. singleton design
    b. idea
            i. s = "applepenapple", wordDict = ["apple", "pen"] t
            ii. s = "catsandog", wordDict = ["cats", "dog", "sand", "and", "cat"] f
                1. primitive idea: for cut position, if left hand side is true keep going right handside, else keep index++
                2. using dp right hand side can not be cut , left hand side part try cut at position
                    a. dp[i] represents if cut at this position, if (any (dp[j]) is true, and dict.contain(substring(j,i))) true
                    b. return dp[length]

14. Next permutation

    a. description
            i. singleton设计模式
            ii. inplace and const extra memory
    b. solution:
            i. idea
                1. inplace => void
                2. e.g.
                    a. 123 ⇒ 132
                    b. 145 ⇒ 154
                    c. 54365 ⇒ 54536
                        i. 54365 ⇒ 54563 ⇒ 545 36

15. max rectangle 输出其坐标

16. Tic tac toe

    a. description
                1. implement tic tac toe, 给a 跟 b 两个user 和 当前棋盘，第一问是给定棋盘，判断现在棋盘谁赢(a win or b win)? (我自己自由 发挥出来了两个别的状态， tie and unfinished)
                2. return 所有能使给定user win的路径 (我用了dfs)

    b. solution
            i. ideas:
                1. first this is a design problem, to play the game, we need two players and a board
                2. go through an example
                    a. let’s say the board is 3 * 3

```
_ X _
X O O
_ X _
```

                    b. from the example
                    c. we need to check the winner every time User moved, whether has a winner or not

            i. if winner occurs go on, clear the board
            ii. if no keep going on
        d. check winner ()
            i. lets say X player, if cur.row, cur.col
                need to check 8 times , in 8 directions

17. Huffman Decoding

18. text Justification

- a. description
    - i. edge cases, i.e. word length > max length
- b. solution:
    - i. idea:
        1. two kinds,
            - a. last line special: words first then spaces
            - b. above lines, calculate the spaces left and evenly distributed
        2. edge cases
            - a. word.length > max length, then add ‘-’ in the end and change line
        3. data structure, ArrayList<String> lines, int counter to count length int pointer to point to words

19. 求data stream最后k个输入的平均值

- a. description
- b. solution：
    - i. idea:

20. 在一个matrix里从左上角到右下角的路径.

- **a.** description
    - i. 开始的问题是只能往右往下走，dp解
    - ii. 后来follow up是四个方向都可以走，dfs解
- b. solution
    - i. example

        XXX
        XX**X**
        XX**X**

    - ii. idea1:
        1. if only right down path, means best paths
        2. paths[i,j] = represents number of paths to this point (i,j)
        3. path[0,j] = 1;
        4. path[i,0] = 1;
        5. induction rule: path[i,j] = path[i-1,j] + path[i,j-1];
    - iii. idea 2:
        1. if all four directions, DFS backtrack, maintain a globalvalue to update numbers, maintain a list to store the current path

21. **coin change**
    a. 一堆不同面值的硬币，一个target，找到最少硬币数，求和等于target
    b. solution:
        i. idea:
            1. [1,2,5] 18 ⇒ 5 + 5 + 5 + 2 + 1   num = 5
            2. [1,3,5] 19 => 5*3, remain 4 % 3, remain = 1, remain = 0
            3. [3,5]   6 => 3*2 + 5 *0 / 5*1, for (3/each) {}
            4. [3,5]   10 => 5*2

```
if (everytime use max number of larger, if remain == 0, return
num)
if (remain != 0) {
        level <= 0, decrease the max number  of large)
        else level;

}
if (large number == array[0] && remain != 0, return 0)
```

22. **给一个序列的定义 n（奇数）：3n＋1 n（偶数）：n/2 每个数都能到1，比如：5->16->8->4->2->1, 长度为5。 找到1-1000中序列最长为多少。 houzz给的链接里，用recursive写会爆栈，iterative的话可以用for loop，每次循环把路径存在stack中，每个数对应的序列长度保存在map中**
    a.
23. **是有一个integer stream, 不知道到底多少个，然后输出随机 k 个sample。**
    a. reservoir sample
        i. 要求每个元素sample 到的概率完全一样,内存不能用太多并且和 performance 要好
    b. solution
24. **LCA of deepest leaf nodes in a binary tree，好像是个fb面经题，写出递归解法没写出iteration解法**
25. **top k visited website this day**
    a. description
        i. idk
    b. solution
26. **basic calculator II**
    a. description lc 227
    b. solution
        i. idea:
            1. we have two level calculations which are + - | * /
            2. use stack to store the pre result ⇒ if higher order, calculate first and insert back to stack, otherwise just calculate in the end
            3. Time = O(n) + (k number) Space = O(k)

            4. But since we only have two levels of calculation, then we can use two variables to store the results and add together
            5. can be optimized to Time = O(n) + Space = O(1)
        ii. if calculator is 2 + (3 + 4)
27. 给一个 数字恩，假设这是一个 恩乘恩 的正方形单元网格，给一个圆，半径以及圆心点坐标，求有多少小格子完全在园内？ 不依不饶： 有没有 欧恩的解法
    a. 本人当时的想法是：根据圆的坐标很容易找到第一个格子，其部分顶点包含在园内，部分在圆外。以这个格子为起始做dfs，找出所有类似的格子，（这些格子全在圆的边界上）。那么所有有相同x坐标的格子，y坐标差减一就是这两个格子之间一定在园内的格子数。这样traverse by x坐标，累积的y坐标差应该就是在园内的格子数。

欢迎大家补充更优解！！

28. 第一题是Given a DAG, a source, a target, and number n, 找出从source到target的长度为n的路径的个数。第二题是given a undirectd and acyclic graph, 一个source和一个target，找出source到target的最长path。
29. LeetCode 277
30. LC 416 partitionEqualSubset Sum
31. calculate bits distances
32. Fibonacci recursive and nonrecursive
33. OOD
    a. Design Twitter
34. 语言问题。
    a. **比如Java和JS的主要区别？**
        i. Key differences between Java and JavaScript: Java is an OOP programming language while JavaScript is an OOP scripting language. Java creates applications that run in a virtual machine or browser while JavaScript code is run on a browser only. Java code needs to be compiled while JavaScript code are all in text
        ii. JavaScript Objects are prototype based. java class based
        iii. js use less memory
        iv. js in single thread, use event based to concurrency, java use thread to handle concurrency
    b. **interface passed as parameter**
        i. Collections.sort(List<T> list)
    c. **linkedlist & arraylist**
        i. not consecutive memory
        ii. can only loop from head, O(n)
        iii. search time O(n)
        iv. operations
            1. size()

        2. get(index)
        3. add(index, E)
    v. Compare ArrayList and LinkedList time

Compare	ArrayList	LinkedList
1. get head/tail	O(1)	O(1)
2. get middle	O(1)	O(n)
3. set head/tail	O(1)	O(1)
4. get middle	O(1)	O(n)
5. add middle	**O(n)**	**O(1)**
6. add head	**O(n)**	**O(1)**
7. add tail	**O(1) if no expand**	**O(1)**
8. remove head	**O(n)**	**O(1)**
9. remove middle	**O(n)**	**O(n)**

        10. **if random access -> arraylist**
        11. **if always add at end, use arrayList**
        12. **when same time complexity use arraylist -> use memory efficiently**

d. **TreeSet**
    i. TreeSet is implemented using a tree structure(red-black tree in algorithm book). The elements in a set are sorted, but the add, remove, and contains methods has time complexity O(log (n)). It offers several methods to deal with the ordered set like `first(), last(), headSet(), tailSet()`

e. **set, list**

f. **舉幾個實作Set & List的類 (HashSet, LinkedList, ArrayList)**

g. **interface和 abstract class**
    i. both can declare method without implementation
    ii. cannot instantiate an object
    iii. general semantic : interchangeable implementation
    iv. Java doesn't have multiple inheritance. In Java, a class can only derived from one class, but can implement many interfaces
    v. abstract class is still a class, can have member fields and non-abstract methods. Also can have constructors; while interface can only have method heading
    vi. abstract class is a class; interface has a function
    vii. When to use which one
        1. abstract class: plan on using inheritance to share common based methods; also good if wanna declare non-public members cause all methods in interface has to be public; Puls, if want to add methods or fields later, abstract class is good choice
        2. Interface is good that API will not change for a while; And also good when you want to have multiple inheritance like implement multiple

h. **TreeMap和HashMap的比较**

   i. HashMap makes absolutely no guarantees about the iteration order. It can (and will) even change completely when new elements are added
   ii. TreeMap will iterate according to the "natural ordering" of the keys according to their compareTo() method (or an externally supplied Comparator). Additionally, it implements the SortedMap interface, which contains methods that depend on this sort order
   iii. LinkedHashMap will iterate in the order in which the entries were put into the map

i. TreeSet and HashSet
   i. HashSet is fast
   ii. HashSet allows null object but TreeSet doesn't allow null Object
   iii. HashSet is backed by HashMap while TreeSet is backed by TreeMap in Java
   iv. HashSet uses equals() method to compare two object in Set and for detecting duplicates while TreeSet uses compareTo() method for same purpose

j. how to implement equals method

```
@Override
public boolean equals(Object obj) {
  if (this == obj) {
     return true;
  }

  if (! (obj instanceof OverridePractice) ) {
     return  false;
  }

  OverridePractice another = (OverridePractice) obj;
  return another.equals(obj);
}
```

k. java polymorphism
   i. Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.
   ii. It plays an important role in allowing objects having different internal structures to share the same external interface

l. what is a deadlock
   i. A **lock** occurs when multiple processes try to access the same resource at the same time.One process loses out and must wait for the other to finish.

   ii. A **deadlock** occurs when the waiting process is still holding on to another resource that the first needs before it can finish.

m. **what happen when you go to**
   i. check browser cache, then OS host file to find if IP was visited, if not, go to DNS to get the IP;
   ii. 2. request get to the correct IP, could be a web/app server, or a Load Balancer and forward you to the web server
   iii. 4. server reply back with 200 OK, and some HTTP contents
   iv. 5. based on the HTTP contents your browser would build up a DOM tree, and get the static contents from CDN, and send other HTTP requests to the server.
   v. 6. there could be some AJAX requests as well.

n. **如何實現Ajax**
   i. Read data from a web server - after the page has loaded
   ii. Update a web page without reloading the page
   iii. Send data to a web server - in the background

o. RestFul
   i. A REST API defines a set of functions which developers can perform requests and receive responses via HTTP protocol such as GET and POST
   ii. it’s a requirement of a REST API that the client and server are independent of each other
   iii. The REST API should specify what it can provide and how to use it, details such as query parameters, response format, request limitations, public use/API keys, method (GET/POST/PUT/DELETE), language support, callback usage, HTTPS support and resource representations should be self-descriptive
   iv. interact with Parse from anything that can send an HTTP request
   v. Stateless – No client data is stored on the server between requests and session state is stored on the client.
   vi. Cacheable – Clients can cache response (just like browsers caching static elements of a web page) to improve performance

p. 如何實現session (cookies).
   i. cookies are name value pairs, store the session_id in the cookie, send within http request header every time
   ii. Server opens a session (sets a cookie via HTTP header)
   iii. Server sets a session variable.
   iv. Client changes page
   v. Client sends all cookies, along with the session ID from step 1.
   vi. Server reads session ID from cookie.
   vii. Server matches session ID from a list in a database (or memory etc).

viii. Server finds a match, reads variables which are now available on `$_SESSION` superglobal.

q. **cookie localStorage**
    i. Cookies are primarily for reading **server-side**, local storage can only be read by the **client-side**
    ii. It stores data with **no expiration date**, and gets cleared **only** through JavaScript, or clearing the Browser Cache / Locally Stored Data - unlike cookie expiry
    iii. The biggest difference here is that the user's state is not stored on the server, as the state is stored inside the token on the client side instead.
    iv. use JWT for authentication for reasons including scalability and mobile device authentication

r. **Scalability. Now we have one server, one database, what if response time is slow? How to optimize?**
    i. 先找出哪里slow；
    ii. if it is the network problem, go talk to the OPS team, buy more server and add load balancer.
    iii. If it is server memory shortage, go find out whether it is the problem of the web server(switch to a light weight engine like Nginx), or the backend itself(optimize code, like change it to multi-threading).
    iv. If the database is slow, find out which query is sluggish and optimize it, or add db server and dedicate each one to reading or writing

s. **http request get 和 post的区别**
    i. GET
        1. only request data from server
        2. query string (name/value pairs) is sent in the URL of a GET request:
        3. /test/demo_form.php?name1=value1&name2=value2
            a. can be cached
            b. remain in the browser history
            c. can be bookmarked
            d. have length restrictions (maximum URL is 2048 characters)
            e. never deal with sensitive data
    ii. POST
        1. send data to server and create/update data
        2. The data sent to the server with POST is stored in the request body of the HTTP request
        3. POST /test/demo_form.php HTTP/1.1
           Host: w3schools.com
           name1=value1&name2=value2
            a. all gets number ⇒ can't

iii. HEAD
    1. requests are useful for checking what a GET request will return before actually making a GET request - like before downloading a large file or response body.

t. bootstrap库里面是怎么实现responsive

u. front

i. ES6
1. const: ES6 allows for ‘immutable’ variables — before const you could not make your variables non-reassignable without the help of object properties.
2. block-scoped variables: No hoisting these! These variables stay safe and sound in their scope.
3. arrow functions: These are like JavaScript candy. It looks clean
4. default parameter values:
5. class inheritance using extends
6. a template literal
7. Promises — resolve/reject/.then: Promises became less clunky too. The
8. Date-time formatting

ii. differences between capturing and bubbling
1. are two ways of event propagation
2. bubbling, the event is first captured and handled by the innermost element and then propagated to outer elements.
3. capturing, the event is first captured by the outermost element and propagated to the inner elements.
4. `addEventListener(type, listener, useCapture)`

iii. js write onclick

iv. event.preventDefault()
1. will not let the default action

v. this in js
1. In a method, this refers to the owner object.
2. Alone, this refers to the global object.
3. In a function, this refers to the global object.
4. In a function, in strict mode, this is undefined.
5. In an event, this refers to the element that received the event.
6. Methods like call(), and apply() can refer this to any object

vi. scoping
1. Local scope : function scope
2. Global scope
    a. If you assign a value to a variable that has not been declared, it will automatically become a GLOBAL variable.

    b. Global variables are not created automatically in "Strict Mode".
3. block scope
    a. Variables declared with the var keyword can not have Block Scope
    b. let keyword have block scope
    c. aslo let re-declaring a variable inside a block will not redeclare the variable outside the block:
    d. const → It does NOT define a constant value. It defines a constant reference to a value.

vii. **hoisting**

1. Hoisting is JavaScript's default behavior of moving declarations to the top.
2. Variables and constants declared with let or const are not hoisted!
3. JavaScript only hoists declarations, not initializations.

viii. **closure**

1. In other words, a closure gives you access to an outer function's scope from an inner function.
2. closures are the primary mechanism used to enable data privac

```
var add = (function () {
   var counter = 0;
   return function () {counter += 1; return counter}
})();

add();
add();
add();

function add() {
   var counter = 0;
   counter += 1;
}

add();
add();
add();
```

ix. **bind, apply call**

x. **class and prototype**

1. Class Inheritance: A class is like a blueprint — a description of the object to be created. Classes inherit from classes and create subclass relationships: hierarchical class taxonomies
2. Prototypal Inheritance: A prototype is a working object instance. Objects inherit directly from other objects.

xi. **absolute and relative**
1. relative: is positioned relative to its normal position
2. absolute:is positioned relative to the nearest positioned ancestor

xii. **inline and block**
1. inline <a> <span> <img> <button> <label> <input>
2. block <div> <h1> <main> <nav> <video>

xiii. **display:none and visibility :hidden**
1. **display**:**none** will not be available in the page and does not occupy any space.**visibility**:**hidden** hides an element, but it will still take up the same space as before. The element will be **hidden**, but still affect the layout. **visibility**:**hidden** preserve the space, whereas **display**:**none** doesn't preserve the space
2. 如果元素的display为none,那么元素不被渲染,position,float不起作用

v. **backend**

i. shell command, log file , put all err starts to another file (grab)
1. grab “error” log > output.text

ii. SQL query （如 select count distinct）

iii. sql里的join是怎么实现的
1. A JOIN clause is used to combine rows from two or more tables, based on a related column between them.
2. (INNER) JOIN: Returns records that have matching values in both tables
LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
FULL (OUTER) JOIN: Return all records when there is a match in either left or right table

35. bq

a. 为什么选择houzz

i. Houzz seems like an exciting new place to work that is fun with opportunity for growth. I want to be a part of that excitement, fun, and growth.

ii. room innovation and all related platform

Onsite :

1. Remove any number containing 9 like 9, 19, 29, ... 91, 92, ... 99, 109... Write a function that returns the nth number. E.g. newNumber(1) = 1 newNumber(8) = 8, newNumber(9) = 10, 最后给了hint把数变成9-based
2. . kSum.... expect better runtime than backtracking... Consider 3sum's ,backtracking complexity is worse than n^2, which use's two sum's two pointer approach
3. Android lock pattern from leetcode
    a. solution
        i. idea
            1.
4. test justification
5. fibo   recur and iterative
6. edit distance
7. Given a n*m size 2D array with integers from 1 to n*m - 1 in it. Integers are not sorted. The last position of the matrix stays a movable block. For each time, you can swap the movable block with any adjacent number. And eventually you will have the integers sorted and the movable block returned to its starting position. Think about an approach to print the path. (You can assume it always have at least a solution)
8. longest increasing subsequence.
9. find max number of points in one line
10. 有向有环图中 找source 到 target 的距离为n的可能性
11. wordbreak 2 (follow up : NLP problems)
12. Design Excel. Implement int get(string cell) void put(string cell, string expr). expr can be "A1= B1+1". 这题的关键在于，要解决各个cell的dependence问题. 比如说call put(B1, "3") 之后，同时也要update A1的值。会牵扯到topo sort的问题。总之这题是design题，就看你有没有意识到这种dependence。
13. Design Instagram
14. design calendar class