http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=dataStructures
http://courses.cs.vt.edu/csonline/DataStructures/Lessons/\

https://docs.google.com/document/d/1DJ7m3jwh35OWPhjYQZe0X8UT0T38VYc9WROFnhjkREo/edit

http://www.programcreek.com/simple-java/
http://www.programcreek.com/2012/11/top-10-algorithms-for-coding-interview/

http://en.wikiversity.org/wiki/Java_Collections_Overview

http://www.developer.com/mgmt/three-types-of-interview-questions-software-developers-should-expect.html

https://mail.google.com/mail/u/1/?pli=1#search/interview+schedule/14c19ecaa707dc79?projector=1

Array-> index
1) Arrays: fixed, index

```
int [] intArray= new int[6];
intArray[7]   // will give ArraysOutOfBoundException.
```

2) ArrayList: resizable Array

get a in the middle -> fast
add / remove a in the middle -> slow
not thread-safe
increase by 50% size

```
List list = new ArrayList();
list.add('11');
list.add('33');
list.remove(0);
```

3) LinkedList: Queue implemented, FIFO

change size, no index(iterate),
get a in the middle -> slow
add / remove a in the middle -> fast

```
LinkedList<String> list = new list<String>();
list.addFirst("Item0")
list.add("Item1");
list.add("Item2");
```

list.remove(0);

4) Vector: internally Array
add(), get(int i) has synchronized block
thread-safe
increase by double size (double)

5) Stack: LIFO
6) Queue: FIFO
7) Trees
8) Hash
9) **Hash Tables**

- List : order O, duplication O
- Set : order X, duplication X
- Map : order X, duplication X, value duplication O

Set O(1) ~ O(n)
array O(1)
arrayList O(1) ~ O(n)
linkedList, queue O(n) ~ O(1)
binary search tree O(log n)
hashmap / hashtable O(1)

Both collections implement Map. and Both collections store value as key-value pairs.
The key differences between the two are
1. Hashmap is not synchronized in nature but hashtable is.
2. HashMap's iterator is fail-safe but Hashtable's enumerator isn't
3. HashMap permits null values and only one null key, while Hashtable doesn't allow key or value as null.

LinkedHashMap - HashMap : order exist or not

Depth-first search: *DFS*
Breadth-first search, ***BFS***

https://docs.google.com/spreadsheets/d/1mki42mNkaxG1wowAXFp4BeSXWfmyR-mim85nP8Se8v0/edit#gid=6

In thread
- yield: yield to next thread, to ready state
- wait: with block, in synchronized block, to waiting state
- sleep: with block
- suspend: like sleep without time, resume
- (time-out), notify(), resume(), interrupt(): to ready state
  - interrupt(): with InterruptedException

Depth-first search: DFS
Breadth-first search, BFS
asymptotic
- constant
- logarithmic
- linear
- exponential

- Object GC status
Created
In Use
Invisible
Unreachable: GC의 후보로 지명
Collected: finalize, Unreachable 상태에서 1회 실시
Finalized
Deallocated: GC의 최종 단계

O(n^2) = n^2

precision
palindrome

exponent = power

squared (2승)
cubed (3승)

10^4
= 10 to the power of 4
= 10 to the 4th (power)

hexadecimal : 16진수

BA의 16 진수
16 * 11 + 10
11
= 186

10자리 글자로 구성된 이름의 case는?
36^10
첫번째가 숫자가 아닌 10자리 글자로 구성된 이름의 case는?
26 * 36^9

1/2 * 3/51 : 1 over 2 times 3 over 51

```
String arry[] = str.split("");
int[] arry2 = new int[2];
int[] arry3 = new int[]{1,2,3};
int[] arry3_1 = {1,2,3};
String[] arry4 = new String[]{"1","2","3"};
String[] arry4_1 = {"1","2","3"};
List list2 = Arrays.asList(arry4);
```

- 생일자 중복 가능성 계산
http://gmaedu.co.kr/bbs_01/read.asp?kind=1&mode=&top=&field=&word=&page=12&webid=139

- linkedlist with queue or stack
- power function with recursion and iteration

Permutations

http://www.programcreek.com/2012/11/top-10-algorithms-for-coding-interview/

- You are responsible for maintaining a web service that sits behind a load balancer. If the web service starts failing, how will you go about fixing it?

- What development language are you most comfortable in? Can you rank it out of 1-10 scale?

⬭ complexity
http://b-jay.tistory.com/110
http://www.slideshare.net/skku_npc/computational-complexity

http://skmagic.tistory.com/164
http://www.hanbit.co.kr/network/view.html?bi_id=1117

인터뷰 예제 메일
https://mail.google.com/mail/u/1/?pli=1#search/Gayle+Laakmann+McDowell/14c35cbd80e15221

Big-O Reference: http://personal.crocodoc.com/CiTPkjz
Just basically understanding the gist of big-O is not enough. You need to be very, very comfortable with it.

Interview Process: http://personal.crocodoc.com/I5jKwlJ
A quick read about what these interviews are all about.

How to Walk Through a Technical Question: http://personal.crocodoc.com/V3Acq1m
Very important read.

Handout:
http://www.crackingthecodinginterview.com/uploads/6/5/2/8/6528028/handout_-_cracking_the_coding_skills.png
Ideally print this out (make sure you print that resized to normal paper) and use it as you go through problems. This is a good handout to go along with the previous link.

Observer

http://javarevisited.blogspot.sg/2011/12/observer-design-pattern-java-example.html

⬭ Average Load
https://lunatine.net/2016/02/19/about-load-average/
vagrant@graphite:~$ lscpu
Architecture: x86_64
CPU op-mode(s): 32-bit, 64-bit
Byte Order: Little Endian
CPU(s): 2

vagrant@graphite:~$ cat /proc/loadavg
0.08 0.14 0.06 2/233 2310
1분, 5분, 15분의 수치

2(cpu갯수) 보다 작으면 정상

ㅇ
vagrant@graphite:~$ cat /proc/meminfo
MemTotal: 1017672 kB
MemFree: 519008 kB

ㅇ yahoo
-. make perfect shuffle
-. get count with date, specific uri from access log
-. how to work https between browser and server.
-. get each element sum from a number
        ex) int sumfunc(2343326)
                = 2+3+4+3+3+2+6
-. move all 0 to the end of array
        ex) 5,6,0,7,0,4,1 -> 5,6,7,4,1,0,0
-. check cpu, memory, io commands in linux
-. explain possible case when cpu, memory, network is ok
-. make weighted loadbalancing
-. How to know disk failure
-. How to get stats of server for a long term.