# 1\ Tips

## 1.1 pre-judgement for simple optimization.

e.g. determine a string has unique characters.

decide two string are permutation.

## 1.2 if the distinct cases are limited, use array instead of map.

## 1.3 check character encoding.

1.4 Examplify. For huge and abstract problem, study its base case and use base case to build a larger case until problems can be solved.

1.5 For characters, if they are in a limited alphabet, use array to store info instead of map.

# 2\ Data Structure

## 2.1 Stack and Queue

e.g. use two stacks to implement queue: lazy solution. Simulate the usage situaion and optimize.

## 2.2 Hash Table

哈希表（Hash Table）是一种根据关键字直接访问内存存储位置的数据结构。HS is not for permanent store. It’s just about when you load data into memory, how could you find it as fast as possible.

## 2.3 Array

Array problems work with DP, two pointers, binary search, backtracking.

Take care of int/double input, overflow, size=0, negative, duplicate number, zeros.

Typical problems: max product subarray, combination, contain most water.

# 3\ Algorithms

## 3.1 Calm down and really analyze it.

Try multiple data structures, especially BST, hash table, stack.

Try dynamic programming. Define your problem in state n.

Try recursive solution

Try divide and conquer.

If you meet problems, solve it instead of stop thinking or move back. Or suppose you get its API, just leave it as a subproblem and solve it later.

e.g. find the next node in binary search tree.

## 3.2 Bit operation

try not to use while. Use more simple expression n <<= (zeros+ones); n |= (1 << ones )- 1;

bit operation: xor(how many different bits in a and b) ~ -1 << >> (\*2 /2)

# 4\ Others

## 4.1 Test

Make the problem/object very specific by asking questions: who are going to use it for what purpose? What functionality does it have? children? What age? Drawing? On what? How will you solve it in real world(don’t think about it’s an interview)

Design plans: Think about use cases. Breaking down big object/problem into small pieces. If object/software has different functionality, test each step(ATM: log in, withdraw, deposit, transfer, check balance). For object, fact check / intended usage / unintended usage / other aspects: safety, comfort, etc. When testing, clarify what aspect do you care about. For functions, normal input, extreme input, wrong input, etc.

1. Test a function:
   1. syntax test: size and type
   2. functional test:
      * 1. normal case: more detail according to your implementation , odd, even, etc
        2. extreme case: empty, large, small?
        3. Null and illegal: negative, overflow, null?
2. Troubleshooting Questions:
   1. Understand the scenario, get a deeper understand of what is happening
      * 1. Who is using it? What are they doing about it? How long has user been experiencing the issue?
        2. How often does it happen?
        3. What’s the version of it? What’s the os platform?
        4. Is there any error report? Any other information?
   2. Break down the problem. Find out the reasons step by step.

e.g How would you load test a webpage without using any test tools?(do not think about the detailed knowledge, use logical and structured thinking to solve it.)

1\ to load test a webpage, basically we are supposed to give the webpage from a minimum value to the maximum value we expected or until it crashes. The load should increase unit by unit. We are supposed to record its behavior for each load.

2\ the behavior should include: response time, throughput, resource utilization, etc.

3\ Because we have no test tools, we can develop our own tools. We use a multithread process to simulate lots of people load the page at the same time.

## 4.2 Brain Teaser

(1) Examplify: if you are given an abstract situation, or a situation with many states, or statement with n, just exemplify it by make detailed example, study one of its cases. Make some assumption to simplify the situation. Suppose there are XXX, Suppose at the beginning we have XXX.

(2) Generalize: if you are given detailed number, use x and y to represent it and study its general rule.

(3) Try to abstract some rules from your detailed example. Like dropping eggs. If you meet problems, declare your situation again to make sure you make full use of every condition and understand them right.

## 4.3 Scalability and Memory Limits

(1) First suppose there are no memory limits and how you would solve the problem.

(2) Explain how much memory this problem needs and show it’s not possible to load them into memory.

(3) So you have to **split data into machines or files or buckets,** and merge the results somehow later.how to split data:

a. by order of appearance b. by hash table c. by relation d. random

Example: find the missing number from 4billion integers with 1GB and 10MB memory.

Find documents that contain a list of words.

Find duplicate URL from 10billons URLs.

## 4.4 Mathematics and Probability

(1) For geometry problem, remember not to assume int unconsciously. DOUBLE is required for most geometry problem: point, slope,etc. When comparing double, DO NOT USE ==. Keep a static epsilon for your class.

## 4.5 Object Design

(1) Understand the object: ask questions instead of making assumptions.

(2) File System Design:

A file system mainly include file and directory. A file consists of blocks. File system usaually use index to record which blocks the file use.

So first we are going to design block.There are two types of block. One is inode. It stores the file property and strorage blocks of this file. Storage blocks store the real content of the file. So we design a base class Block. It contains block\_id, block\_size, fileds to map to physical memory address. Inode inherites from Block. It contains file property and vector<StorageBlock>. Storage block inherites from Block, it contains file content.

So next we are going to design file. File.create(), File.delete(), File.open()… When delete a file, free the blocks. When create a file, put its inode to its parent inode’s storage blocks so that you can find it later.

Then we are going to design Directory. Very similar to File. Directory.create(), Directory.delete(), Directory.open()

Oh, I think we should create a base class for file and directory.

Then we should have a RootObject. It manages all blocks. It records which nodes is occupied, which is valid. It assigns a new block when we need to create file or directory. So besides real storage blocks, there are bit map(record free blocks), inode, disk information to take memory.

# 5\ Good Examples to Think Twice

## 5.1 Find all documents that contain a list of word

There is thinking flow of this problem. You propose a solution, analyze its performance and try to improve it using another method.

a. A naïve solution is to deal with documents one by one. Iterate through one document blablabla… If the average size of document is N, it takes O(N) time complexity and O(x) space.

b. However, the problem about this solution is that if we change the list, XXXXX

c. So we maintain a hash table and iterate through all documents. The hash value is word, the key is which document it appears. Now we get a list of words, we just need to find the numbers which appear in all the values.

d. How could this problem be solved? Bit operation &.

e. finally, the time complexity is , the space is. The advantage is that if list changed, we are able to ……

f. now the problem is millions of documents: the issue comes up with the situation is that we may not able to hold our documents in one machine. Or even hash table is to large to fit in one machine. We can solve it by either divide hash tables or divide documents.

h. Then each machine handles a small part of the data. Then we merge the results from each machine.

## 5.2 Find missing integers

suppose there are 2^31-1 distinct integers in range[0, 2^31-1]. Find the missing integers. what if the memory is 1GB? What if the memory if 10MB?

1GB is 8billion bits. So each bit can stand for a number. Use bit array(define by yourself) to solve problems. There are 2^31 numbers so we need 4billion bits. But now we only hae 80\*2^10bits. So divide into buckets and iterate twice.

## 5.3 Given two line. Decide if they are intersect.

Fro problems like this:

1. **Ask questions to clarify problems.** Especially you should consider about some corner cases.

What line it is? Segement or infinite line? What if two lines are the same?

1. **When possible, design and use data structure to show you have good sense of object oriented programming**. Think about different ways to design the data structure. Explain your choice.

I would like use y = slope\*x+yintercept to represent a line because…

class Line{

public static double epsilon = 0.000001;

public double slope, yintersect;

public Line(double slope, double yintersect)

}

1. Do not assume int in design. Use double for some math problems. Understant limitations of floating point representation. Do not compare two float pointing number using ==. Keep a static epslon inside your data structure.

## 5.4 Write methods to implement the multiply, substract and division use only add operator.

Fro problems like this:

1. Remember all good interview problems can be approached in a logical and methodical way. The interviewer is looking for this sort of **logical work-your-way-through-it** approach.

for this problem, analyse what exactly mutiply, substract and division do and how it connects to add.

1. Write clean code – if you find you use some methods multiple times, encapsulate it into method..

Focus on the main workflow. If you meet some unknown issues, use the API and implement it later.

1. **careful about coding details**. Do not glance over special cases on each step.

## 5.5 Given a two-dimensional graph with points on it, find a line which passes the most number of points.

1) logical work-your-way-through-it.

2) Generally two methods: for each line, check all points. O(N^3)

compute all lines first, check how many duplicate

1. if it is int, check duplicate could be O(N) with map. For double, it’s much more difficult.
2. First hash each lines into spot. So the potential equal line could be hashd into the same or consecutive spots. And it looks like you narrow the range in which you are looking for equal lines.