Low quality pictures

juliachencoding.blogspot.ca

1. 2015

1.1 June

1.1.1 Breadth first search algorithm - calculate steps from room to guard using matrix (2015-06-09 18:49)

March 31, 2015

Question is from careercup:

http://www.careercup.com/question?id=4716965625069568

```
Given a
2
-D matrix represents the room, obstacle and guard like the following (
0

is
 room, B->obstacle, G->
 Guard):


0

0

0


B G G
B

0

0


calculate the steps

from
 a room to nearest Guard and
set
 the matrix, like
this
```

```
2

1

1


B G G
B

1

1


Write the algorithm, with optimal solution.
```

[illegible]G[illegible]B[illegible]-[illegible] [illegible]-[illegible]

[illegible]-§Ķ[illegible]-[illegible] [illegible]BFS[illegible]-[illegible]O(n^2)[illegible]

Read the blog, and understand the solution; and then, implement it using C #

http://www.cnblogs.com/zhuli19901106/p/3710623.html

Try to implement the algorithm, so excited to practice a question used by Google interview. [illegible]Google[illegible].

Share code written in C # language:

https://github.com/jianminchen/BFS _RoomToGuard/blob/master/Program.cs

1.1.2 Longest Substring Without Repeating Characters (2015-06-09 18:55)

Given a string, find the length of the longest substring without repeating characters. For example, the longest substring without repeating letters for "abcabcbb" is "abc", which the length is 3. For "bbbbb" the longest substring is "b", with the length of 1.

Leetcode: https://leetcode.com/problems/longest-substring-without-repeating-c haracters/

The code is accepted by online judge of leetcode, on April 1, 2015.

https://github.com/jianminchen/longestSubstringWithoutRepeatingChars/blob/ma ster/Program.cs

continue to read blogs, and then, write C # code:

https://github.com/jianminchen/longestSubstringWithoutRepeatingChars/blob/ma ster/longestSubstringWithoutRepeatingChar-s.cs

1.1.3 leetcode 164: Maximum Gap - a Distribution sort (bucket, counting, radix) algorithm (2015-06-09 18:57)

Here is the leetcode link I am reading in the morning on April 4, 2015.

http://yucoding.blogspot.ca/2014/12/leetcode-question-maximum-gap.html

And then, find out myself to read this article from the following links, try to find readable explanation about the algorithms related to distribution sort algorithms.

https://www.cs.princeton.edu/ rs/AlgsDS07/18RadixSort.pdf

May 22, 2016 (30 minutes at least)

Review the solution on this blog:

https://github.com/jianminchen/LeetCode-Java-Solutions/blob/master/164.maxim um-gap.java

Read blogs talking about algorithm first,

https://ask.julyedu.com/question/213

http://2hwp.com/LeetCode/164 %20Maximum %20Gap/

Distribution Sort - □□□□

Radix Sort - □□□□

Actionable Items:

Spend 4 hours + work on code implementation, Java, C++ -> C # code.

1.1.4 leetcode Question: Find Peak Element (2015-06-09 18:58)

Practice daily, and train myself to know the algorithms better. Notes from April 8, 2015

"find peak element" question: A peak element is an element that is greater than its neighbors. For example, in array [1, 2, 3, 1], 3 is a peak element and your function should return the index number 2.

So, the websites I read:

1. http://yucoding.blogspot.ca/2014/12/leetcode-question-find-peak-element.h tml
2. http://www.geeksforgeeks.org/find-a-peak-in-a-given-array/
3. http://courses.csail.mit.edu/6.006/spring11/lectures/lec02.pdf
4. https://leetcode.com/problems/find-peak-element/
5. https://www.youtube.com/watch?v=HtSuA80QTyo (MIT lecture, 53 minutes, really good one!)

So, I like to write something to help myself ace the question next practice. Using analysis, it may not perform under stress; how about memorization, it will not last after a few months.

Here, using a special case, an array in increasing order. Also, people can help you do analysis if you present a special test case, know where you are in the problem solving.

For example, the array, [1, 2,3,4,5,6,7,8,9], if we go through one by one, 9 is the peak number; but it takes 9 steps from checking 1, then 2, and last is 9. It is not efficient, O(n) steps. So, how to do log(n) step, first one is to check the middle number, 5; and then, 5 is not a peak element, then, analyze and conclude that: [6,7,8,9] has one peak element; and then, check number 8, which is not a peak element; [9] has a perk element. First check is number 5, last check is number 9, only 3 checks; 3 checks is 5, 8, 9.

1.1.5 Find the two numbers with odd occurrences in an unsorted array (2015-06-09 18:59)

March 25, 2015

Study the algorithm, and work on one algorithm every day.

Here is the website with the question.

http://www.geeksforgeeks.org/find-the-two-numbers-with-odd-occurences-in -an-unsorted-array/

One thing is about bit operation, any number a, a XOR a = 0.

Study the question a few times; sometimes, you just cannot remember the trick.

How about going through an example, for example, an unsorted array, only two numbers with odd occurrences, one is 2, one is 8, how to find those two numbers?

So, 2 is 0010 in binary format, and 8 is 1000 in binary format; since two numbers are different, at least one bit is different; second to rightmost bit is different between 2 and 8. one is 1, and another one is 0. The sum of XOR operation for those numbers if second to rightmost bit is 0 will be the one of numbers, and then, second number will be the sum of XOR operation for those numbers if second to rightmost bit is 1.

1.1.6 Websites - learn from others...... (2015-06-09 19:00)

[illegible]! 2010[illegible], [illegible], [illegible]; [illegible], [illegible], [illegible], [illegible], [illegible]; [illegible], [illegible]; [illegible], [illegible], [illegible], [illegible], [illegible]; [illegible], [illegible], [illegible]; [illegible], [illegible]; [illegible], [illegible]

My favorite one is this book about facebook culture in Chinese:

Facebook[illegible], [illegible]Facebook[illegible]-ţ[illegible], [illegible],[illegible].[illegible], [illegible].

http://book.51cto.com/art/201308/405778.htm

And all others in English:

online judge: http://ac.jobdu.com/hhtproblems.php

LeetCode: http://yucoding.blogspot.ca/

LeetCode: http://blog.csdn.net/kenden23/article/details/18696083

Facebook engineers:

LeetCode: https://plus.google.com/103133232094630409192/posts

LeetCode: http://fmars.me/write/leetcodesolutions.html

Facebook engineer:

LeetCode: https://plus.google.com/111516613991361838862/about

[illegible]-Ĩæ[illegible]:

LeetCode: http://codeganker.blogspot.ca/2014/04/palindrome-partition ing-ii-leetcode.html

Google [illegible]:

LeetCode: https://plus.google.com/109529513871383885049/posts

Microsoft:

https://plus.google.com/+XinghuaLiracky/posts

□□□□□□:

LeetCode: http://www.cnblogs.com/TenosDoIt/p/3434579.html

□□:

http://blog.csdn.net/v _july _v/article/details/6543438

https://github.com/julycoding/The-Art-Of-Programming-By-July/blob/master/eb ook/zh/Readme.md

□□□□:

LeetCode solution: https://github.com/soulmachine/leetcode

Algorithms: http://blog.csdn.net/v _july _v/article/details/6543438

Answers lookup: http://www.jiuzhang.com/solutions/

□□□□□□□, □□□□□, □□□□□! □□□□□□□□□□□□□□□Ě!

Princeton university, Algorithms: https://www.youtube.com/watch?v=pR4vbq3GraA

http://www.julyedu.com/video/index

January 6, 2016

http://xiaohuahua.org/hwang/2013/11/03/re-f %E5 %AE %B6 %E9 %9D %A2 %E7 %BB %8F/

favorite advice :

□□□□□□□□□□□□□□□□□□□□□□□□□□-□□□□□□□□□□bug

free□□□□□□□□□□□□□□□□-Ь□□□□□□□□□□□□□□□6□

communication□

□□□□□□□-Ш□□□□□□□□□discussion□□interactive session□□□□□

□□□□□

1.1.7 Leetcode: Convert Sorted List to Balanced Binary Search Tree (I) (No. 109) (2015-06-09 19:01)

Problem description: Given a singly linked list where elements are sorted in ascending order, convert it to a height balanced BST.

□□□□1□□□□, □□□□□□□□C□□□, □□□□□; □□□□□□; □□□□□□□□□□□□□□Y□□□□□□□□□□□□□□ □□□□, □□□□; □□□, □□□□□LeetCode□160□□□, □□□□□□, □□□□□□.

□□□□□□□:

http://codeganker.blogspot.ca/2014/11/leetcode.html

http://codeganker.blogspot.ca/2014/04/convert-sorted-list-to-binary-sea rch.html

□□□□□□□□□□; □□□□□□□□□□□□□□□м□□□□□. □□□□□□□□□□□□C□□□□□□□□□□□□□□□□-ў□

https://github.com/soulmachine/leetcode

Write C # code based on the analysis of the above Leetcode solution.

https://github.com/jianminchen/Leetcode _C-/blob/master/109ConvertSortedListToBinarySearchTreeB.cs

1.1.8 LeetCode: triangle (2015-06-09 19:01)

April 4, 2014 - work on one leetcode algorithm - dynamic programming about the triangle.

Given a triangle, find the minimum path sum from top to bottom. Each step you may move to adjacent numbers on the row below.

Read 3 websites:

https://leetcode.com/problems/triangle/

Best solution:

http://fmarss.blogspot.ca/2014/10/triangle.html

Not optimal one:

http://yucoding.blogspot.ca/2013/04/leetcode-question-112-triangle.html

Cloud not figure out the best solution with the example, and then, read this website with an example:

http://www.mathblog.dk/project-euler-18/

I got it. Using dynamic programming, and also from bottom to up. Here is the summary of algorithm using the example:

We start with a triangle that looks like

3
7 4
2 4 6
8 5 9 3

Applying the algorithm to the small problem we will need three iterations. The first iteration we apply the rule a + max(b,c) which creates a new triangle which looks as

3
7 4
10 13 15

Making the second iteration of the algorithm makes the triangle look

3
20 19

And if we run the algorithm once more, the triangle collapses to one number – 23 – which is the answer to the question.

1.1.9 Leetcode 109: sorted list to binary search tree (II) (2015-06-09 19:02)

problem description:

Given a singly linked list where elements are sorted in ascending order, convert it to a height balanced BST.

□□□□□□□□□□□□□, □□, □□□□□□□□□□; □□□□□□□□□□; □□□□□□□□□□. Spent over 3 hours on April 24, 2015, try to figure out the problem. Before, instead of using global variable list, a local variable is used as the recursive function argument. Learn a lesson, make my day more exciting.

C # code is accepted by leetcode on April 25, 2015

https://github.com/jianminchen/LinkedListToBST/blob/master/Program.cs

1.1.10 Algorithms: Peak element, Document distance (2015-06-09 19:02)

1. Find peak element - on April 23, 2015

https://www.youtube.com/watch?v=HtSuA80QTyo

2. Document distance - on April 24, 2015

https://www.youtube.com/watch?v=Zc54gFhdpLA

□□□□□□□□, □□□, □□□□. Reading is a best way to understand the lectures as well. Spend time to read the notes. on May 3, 2015

http://courses.csail.mit.edu/6.006/spring11/lectures/

1.1.11 Algorithm blog - 30 questions (2015-06-09 19:03)

April 26, 2015

Spent over a few months to go over 30 questions about algorithm starting from January 13, 2013. □□□□, □□, □□□□.

Some of questions are my favorites ones.

http://www.ardendertat.com/

□□□-□□□□□□□□: (Most memorized example:)

http://www.ardendertat.com/2011/09/20/programming-interview-questions-2- matrix-region-sum/

Later, in website administration, IIS cache configuratuion, another important lesson learned is to set up time to refresh the cache. That is also an important thing to keep people informed how refresh the content on the web page. Personally, I designed a simple cache with time expiration 60 seconds on the website.

□□, □□□□□□□□□, □□, □□□□□□□□, □□□□□□□□□, □□□□□□□□. □□□□□, □□□□□, □□□□□□□□□□□□□, □□□□□□□Cache.

□1998□□□□□□□□□□, □□□□□□□□, □□□□□□□□□□□□□□□□□□□, □□□□□, □□, □□□, □□□□. □□□□□□□□□, □□□□□, □□□□□□□□□□□□.

1.1.12 Dynamic programming: longest increasing subsequence (2015-06-09 19:04)

Took a break from Leetcode questions, and then, watched the video in the evening of April 27, 2015; and there is a short lecture about the question: find a longest increasing subsequence; □□□□□□□□, □□□□□□□□□, □□□□□□□, □□, □□□□□, □□□□□, □□□□□□□.

Here is the website (around 37minutes):

https://www.youtube.com/watch?v=jZbkToeNK2g

And then, I plan to study the question and practice it using C # code.

Another website to read:

http://www.geeksforgeeks.org/dynamic-programming-set-3-longest-increasing-su bsequence/

Reading time, catch up more from this website:

http://www.cs.illinois.edu/class/fa07/cs473ug/Lectures/lecture14.pdf

Need to write C # code and get the idea how difficult the problem is.

Sept. 6, 2016

https://www.hackerrank.com/challenges/longest-increasing-subsequent

1.1.13 LeetCode article: Finding the Minimum Window in S which Contains All Elements from T (2015-06-09 19:04)

April 26, 2015

Problem statement:

Finding the Minimum Window in S which Contains All Elements from T

□□□□, 1□28□, 2015□, □□□□□, □□□□□, □□□□O(N)□□□. □□□□□□□, □□□□□□□□□□□□□□, □□□□□□, □□□□. □□□□, □□□□□□□□□□□.

□□, abc□□□□□□□, □□□□□□□abbca; □□□abbc, □□bca? □□□□□□. □□□□□, □□□□□□□□□□, □□□□□□□.

□□□□□□□O(NxN), □□□□, □□□□□, □□□□, □□, □□, □□□□, □□□□□□□□□O(N)□□□□ , □□□□□. □□□□□□□□□□□.

□□□□:

http://articles.leetcode.com/2010/11/finding-minimum-window-in-s-which.html

on June 5, 2015

came cross this webpage, excellent article:

http://www.cnblogs.com/lichen782/p/leetcode _minimum _window _substring _3.html

1.1.14 Leetcode: Majority element (2015-06-09 19:05)

Given an array of size n, find the majority element. The majority element is the element that appears more than □ n/2 □ times.

Read the website, very good one. April 29, spent half hour to read the solution, from 10:00pm-10:22pm.

http://hjhjw1991.github.io/Leetcode-Majority-Element/

Here is the notes:

□□

□□□□□□□□□□□ϴ□□□□□□n□□□□□□□-Ч□□□□□□□floor(n/2)□□□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□-□□□/-

□□□□

1. Runtime: O(n2) — Brute force solution: Check each element if it is the majority element.
2. Runtime: O(n), Space: O(n) — Hash table: Maintain a hash table of the counts of each element, then find the most common one.
3. Runtime: O(n log n) — Sorting: As we know more than half of the array are elements of the same value, we can sort the array and all majority elements will be grouped into one contiguous chunk. Therefore, the middle (n/2th) element must also be the majority element.
4. Average runtime: O(n), Worst case runtime: Infinity — Randomization: Randomly pick an element and check if it is the majority element. If it is not, do the random pick again until you find the majority element. As the probability to pick the majority element is greater than 1/2, the expected number of attempts is < 2.
5. Runtime: O(n log n) — Divide and conquer: Divide the array into two halves, then find the majority element A in the first half and the majority element B in the second half. The global majority element must either be A or B. If A == B, then it automatically becomes the global majority element. If not, then both A and B are the candidates for the majority element, and it is suffice to check the count of occurrences for at most two candidates. The runtime complexity, T(n) = T(n/2) + 2n = O(n log n).

6. Runtime: O(n) — Moore voting algorithm: We maintain a current candidate and a counter initialized to 0. As we iterate the array, we look at the current element x:
    - If the counter is 0, we set the current candidate to x and the counter to 1.
    - If the counter is not 0, we increment or decrement the counter based on whether x is the current candidate. After one pass, the current candidate is the majority element. Runtime complexity = O(n).
7. Runtime: O(n) — Bit manipulation: We would need 32 iterations, each calculating the number of 1's for the ith bit of all n numbers. Since a majority must exist, therefore, either count of 1's > count of 0's or vice versa (but can never be equal). The majority number's ith bit must be the one bit that has the greater count.

1.1.15 Leetcode: zigzag order traversal of binary tree (2015-06-09 19:06)

Spent a few hours to study the question. The website I read:

http://www.jiuzhang.com/solutions/binary-tree-zigzag-level-order-traversal/

One thing I do not like is to use two while loop, and make me wonder how to make it more simple and readable.

This is a better solution, I like it more:

http://rleetcode.blogspot.ca/2014/02/binary-tree-zigzag-level-order.html

Here is C # code I write and test on April 29, 2015:

https://github.com/jianminchen/zigzagOrderTraversal/blob/master/Program.cs

1.1.16 Leetcode: longest common prefix (2015-06-09 19:06)

First time to study the question on Aprill 29. And read the website:

□□□□
// LeetCode, Longest Common Prefix
// □□□□□□□□□0 □□□□□□□□□□□□□□□□□□□□□□□□□□□□
// □□□□□O(n1+n2+...)
// @author □□(http://weibo.com/zhouditty)
class Solution {
public:
string longestCommonPrefix(vector &strs) {
if (strs.empty()) return "";
for (int idx = 0; idx < strs[0].size(); ++idx) { // □□□□
for (int i = 1; i < strs.size(); ++i) {
if (strs[i][idx] != strs[0][idx]) return strs[0].substr(0,idx);
}
}
return strs[0];
}
};

1.1.17 Learn to write a good program from string function - practice, drills using string functions (2015-06-09 19:07)

□□□□□□□□□□□, □□□□□. □□□□□□□□□□, □□□□□□□□□□□□□□□. □□□□□□□□□□, □□□□□!

To write String functions is a best drill for me to practice, and then, improve my coding skills in short time period. Time well spent. My most favorite is to read the following notes. - April 30, 2015

□□□□□□,

http://blog.csdn.net/v _JULY _v/article/details/6417600

□□□□□□, □□!

□□□□□□□□□□
□□□□□
□□□□□□□□strcpy□
□-□□□□□extern char *strcpy(char *dest,char *src);
□□□□src□□□NULL□□□□□□□□□□dest□□□□□□□□□
□□□src□dest□□□□□□□-□□□□□□□□dest□□□□□□□□□□□□□src□□□□□
□□□-gƃdest□□□□

□□□□□□□□□□□□strcpy□□□□□□□10□□□□□□□□□□□□□□□-ȯ□

```
// 2
void strcpy( char *strDest, char *strSrc )
{
 while( (*strDest++ = * strSrc++) != '' );
}

// 4
void strcpy( char *strDest, const char *strSrc )
{
 //   const      2
 while( (*strDest++ = * strSrc++) != '' );
}

// 7
void strcpy(char *strDest, const char *strSrc)
{
 //      0  3
 assert( (strDest != NULL) && (strSrc != NULL) );
 while( (*strDest++ = * strSrc++) != '' );
}

// 9
//          2
char * strcpy( char *strDest, const char *strSrc )
{
 assert( (strDest != NULL) && (strSrc != NULL) );
 char *address = strDest;
 while( (*strDest++ = * strSrc++) != '' );
 return address;
}

// 10
//           1
char * strcpy( char *strDest, const char *strSrc )
```

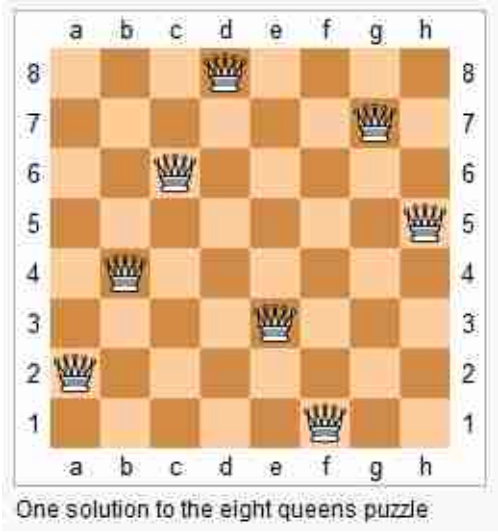
```
{
 if(strDest == strSrc) { return strDest; }
 assert( (strDest != NULL) && (strSrc != NULL) );
 char *address = strDest;
 while( (*strDest++ = * strSrc++) != '' );
 return address;
}
```

1.1.18 Leetcode: N puzzle queens (2015-06-09 19:07)

Read the blogs, and understand the algorithm first:

N-Queens

The *n*-queens puzzle is the problem of placing *n* queens on an *n*×*n* chessboard such that no two queens attack each other.



One solution to the eight queens puzzle

Given an integer *n*, return all distinct solutions to the *n*-queens puzzle.
Each solution contains a distinct board configuration of the *n*-queens' placement, where 'Q' and '.' both indicate a queen and an empty space respectively.

http://www.acmerblog.com/leetcode-solution-n-queens-6254.html

http://blog.csdn.net/feixiaoxing/article/details/6877965

and later, wrote the program using C #, share the code:

Algorithm: Eight puzzle queen

https://github.com/jianminchen/eightPuzzleQueen/blob/master/Program.cs

https://github.com/jianminchen/Leetcode _C-/blob/master/51NQueenProblems.cs

January 12, 2016

Review the algorithm again. The following blog gives a good explanation, help me understand the algorithm better.

http://blog.csdn.net/linhuanmars/article/details/20667175

http://buttercola.blogspot.ca/search/label/Leetcode

http://yucoding.blogspot.ca/2013/01/leetcode-question-59-n-queens.html

https://polythinking.wordpress.com/2013/02/27/leetcoden-queens-i-and-ii/

It is fun to read solutions in C++/Java.

1.1.19 Algorithm: Sort the array placing negative numbers before positive numbers, and keep the orders. (2015-06-09 19:08)

Study the article about sorting the array by even/odd number, placing odd numbers before even number. And then, next question, sorting the array by negative/positive number, and keep the order. - on May 4, 2015, 8;30pm - 11:00pm, read 2. 5 hours.

First question, ??????????????????????????-ųŇ???????????????????????????????? ????????O(n)

https://github.com/jlycoding/The-Art-Of-Programming-By-July/blob/master/ebo ok/zh/02.06.md

http://www.cnblogs.com/legendmaner/archive/2013/04/22/3035050.html

http://www.cnblogs.com/legendmaner/archive/2013/04/04/2999769.html

http://www.careercup.com/question?id=5201559730257920

Read the paper: STABLE MINIMUM SPACE PARTITIONING IN LINEAR TIME Next question: ??-???????????? input: 1,7,-5,9,-12,15 ans: -5,-12,1,7,9,15 ????????O(n),??O(1)

1.1.20 Write a strtok function (2015-06-09 19:08)

May 4, 2015

Spent time to read the website:

http://blog.csdn.net/v _JULY _v/article/details/6417600

http://www.cplusplus.com/reference/cstring/strtok/

http://fengl.org/2013/01/01/implement-strtok-in-c-2/

and http://www.careercup.com/question?id=2755

1.1.21 dynamic programming: cutting wood (2015-06-09 19:09)

Problem statement:

Cutting Wood Your favorite sawmill charges by length to cut each board of lumber. For example, to make one cut anywhere on an 8 ft. board of lumber costs $8. The cost of cutting a single board of wood into smaller boards will depend on the order of the cuts

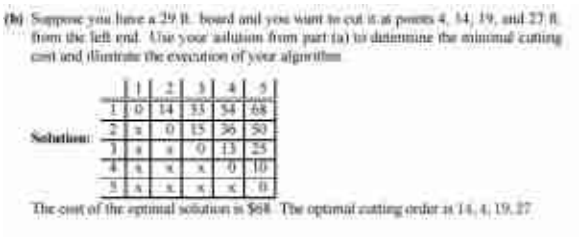
May 6, 2015

□□□, □□, □□□; □□, □□□□. □□□, □□□□□□□□□□□□.

https://www.youtube.com/watch?v=hkAONP0aC9w &index=47 &list=PLUl4u3cNGP61Oq3tWYp6V _F-5jb5L2iHb

http://courses.csail.mit.edu/6.046/spring04/handouts/ps6sol.pdf

Suppose you have a 29 ft. board and you want to cut it at points 4, 14, 19, and 27 ft. from the left end. Use your solution from part (a) to determine the minimal cutting cost and illustrate the execution of your algorithm.


(b) Suppose you have a 29 ft. board and you want to cut it at points 4, 14, 19, and 27 ft. from the left end. Use your solution from part (a) to determine the minimal cutting cost and illustrate the execution of your algorithm.

Solution:

	1	2	3	4	5
1	0	14	33	54	68
2	x	0	15	36	50
3	x	x	0	13	25
4	x	x	x	0	10
5	x	x	x	x	0

The cost of the optimal solution is $68. The optimal cutting order is 14, 4, 19, 27.

Similar question and much easy to figure out:

http://www.geeksforgeeks.org/dynamic-programming-set-13-cutting-a-rod/

1.1.22 3-way partition problem: Dutch national flag problem and its variants (2015-06-09 19:10)

May 6, 2015

First, I read the article about this algorithm related to partition, (in Chinese) - May 6, 2015

https://github.com/julycoding/The-Art-Of-Programming-By-July/blob/master/eboo k/zh/02.07.md

and then, find similar article to read later:

http://www.geeksforgeeks.org/dynamic-programming-set-13-cutting-a-rod/

February 2, 2016

Julia reviewed the quick sort algorithm, and then, practice the C # code to write partition:

Quick sort algorithm writing in C # - less than 20 minutes (18 minutes to write the code with comments)

https://shepherdyuan.wordpress.com/2014/08/03/sorting-algorithms/

Let us describe how quick sort works: - explain it using 5 minutes - 10 minutes to write the code

1. First, using partition, and then, divide and conquer, use recursive function calls to solve the problem.

Most important technique, is to design a partition - 2-way partition, how to design the partition, choose pivot point, and how to do partition step by step.

Let us work on a simple example and have discussion, show the procedure of partition.

Let us say that array 1, 2, 3, 4, 5, 6, sorted

and then, we like to derive the above sorted array using this one, 1, 3, 2, 5, 4, 6

The idea of pivot position, is to choose **last number** in the array, for example, choose last one in the array, 6,

the position of pivot point, left side <= 6, right side > 6

So, do you see the problem, 6 is not ideal case.

Let us work on another order: 1, 6, 2, 5, 4, 3,

Overview of test case design 1, 6, 2, 5, 4, 3 **:**

Tips:

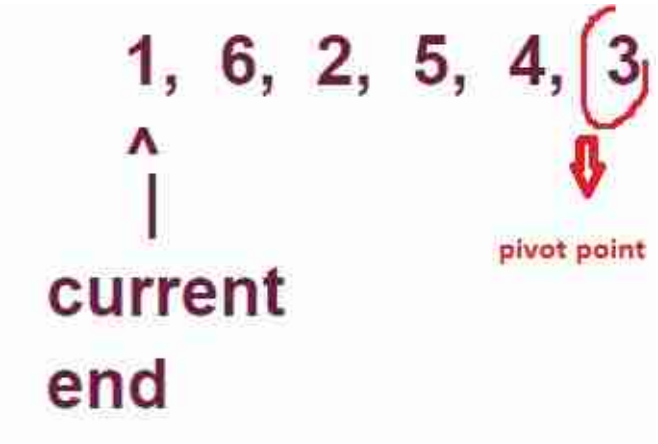
left partition: 2 values, right partition 3 values, partition only needs two swaps, first swap for left partition, second one is for pivot point positioning. <- remember my choice, design test case. Julia tried to make her test case easy to follow.

So, this way, the test case is set up to demo the algorithm clearly and efficient enough.

work on last one in the array, 3, using it as pivot: 1, 2 left side,

6, 5, 4 right side; and then, put the pivot point in-between, leave as is, left/ right side goes to subproblem, use recursive call.

Use a diagram to show progress:

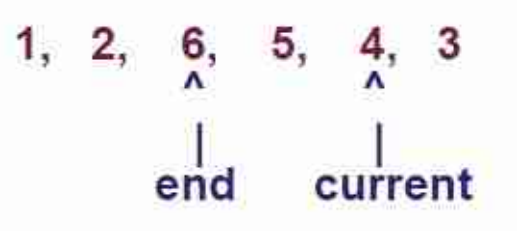


current <- iterate from start - 0 to last one - right -1, leave A[right] for pivot point

another pointer -> mark first node > pivot point value 3, s

So, two pointers, one is moving to iterate whole array except last one - pivot point.

Second one is to mark the position of first node > pivot value, let us call it **end** .



After review, two pointers will mark the right partition:

right side start and end position.

6 5 4

end - 2

current - 4 // not 5, start from 0

left partition:

left -> end -1

Does this tip help to program? Relax a little bit.

Julia's practice:

https://github.com/jianminchen/AlgorithmsPractice/blob/master/quickSortA lgorithm.cs

1.1.23 An array only three unique number, all other numbers are duplicated twice. (2015-06-09 19:11)

May 8, 2015

Spent 30 minutes to work on the problem; read two web pages; and later, need to write down the function. on May 8, 2015.

http://zhedahht.blog.163.com/blog/static/254111742012830842464 12/

http://blog.csdn.net/wypblog/article/details/8032898

1.1.24 Array shuffle in place, O(1) - perfect shuffle (2015-06-09 19:11)

May 7, 2015

Problem statement:

An array with 2n numbers, {a1,a2,a3,...,an,b1,b2,b3,...,bn }, sort them in the following order:

{a1,b1,a2,b2,....,an,bn }, and optimal solution time is O(n), and space is O(1).

□□□□□2n□□□ {a1,a2,a3,...,an,b1,b2,b3,...,bn }□□□□□□□ {a1,b1,a2,b2,....,an,bn }□□□□□□□□□□o(n)□□□□□□□0(1)□□□□

□□□□□□□□, □□□□□. □□□□, □□□□□□□□□□, □□□□□□□□□□. So much fun to read the those two articles, and then, understand the idea of perfect shuffle. Well spent time to enjoy the reading around 1 hour from 9:00pm - 10:13pm, May 6, 2015.

https://github.com/julycoding/The-Art-Of-Programming-By-July/blob/master/ebo ok/zh/02.09.md

And then, go through one example by reading the article:

http://blog.sina.com.cn/s/blog _604d011e0100l9bh.html

It is fun and exciting to learn something new. Definitely good experience.

Here are some notes written down by Julia:

Brute force algorithm: n=4, a1, a2, a3, a4, b1, b2, b3, b4

the order after the change: a1, b1, a2, b2, a3, b3, a4, b4

step 1: b1 is the 5th position in the original array, and then, b1 swaps with the one before, in other words, b1 swaps a4 first; and then, b1 swaps with a2; and last, b1 swaps with a2. How many swaps, 3.

Step 2: b2 is the 6th position in the original array. b2 swaps with a3, a4;

Step 3: b3 swaps with a4

Step 4: b4 is the last one, no swap.

The brute force solution time complexity is O(N^2) since (n-1)+(n-2)+..+1= 2n*(n-1).

In the year of 2004, the paper: "A Simple In-Place Algorithm for In-Shuffle" shares the idea to implement using time O(N).

Like to spend more time to read the article more carefully and then write the code.

January 18, 2016

Review the algorithm. Julia, you should focus on brute force algorithm, learn how to do analysis to solve the problem first. Most of important, by any means, is to solve the problem first, using time complexity O(NxN) solution, O(1) space. O(1) space means that solution only consists of swapping two nodes.

Optimal solution is O(N) time, O(1) space. There is also a solution to use O(N) time, but O(N) space, called perfect shuffle.

To understand the solution, and then, also know that there is a better solution. What ideas are to support the improvement here?

Please remember ideas used in the algorithm:
Array - brute force - 1. □□□□ 2.

□□□□

Array - brute force -

□□□□□□/ perfect shuffle (1)/ Cycle-leader (2)

1.1.25 Two sorted array ascending order, find kth smallest sum of a+b, a from first array, b from second one. (2015-06-09 19:12)

Problem statement:

Two sorted array ascending order, find kth smallest sum of a+b, a from first array, b from second one.

Try to figure out the solution by myself, first 20 minutes,

and then, go through google, read this article: on May 8, 2015

http://www.cnblogs.com/wuchanming/p/3840345.html

http://articles.leetcode.com/2011/01/find-k-th-smallest-element-in-uni on-of.html

1.1.26 Leetcode Solution C++ (2015-06-09 19:13)

5/9/2015

□□, □LeetCode□□□□□□, 200□□; □□□□□; □□□, □□□□

□□□□□□□□□□đ(C++)□□□□, □□□.

Spent time to print out leetcode solution, and then, start to read the solution; Read first, and then, practice later.

And also, print out another book about hand writing preparation handbook (C++).

And watch the videos of algorithms in Chinese.

□□□:

http://www.julyedu.com/video/index

1.1.27 Leetcode 164: Maximum gap - Distribution Sort (Radix Sort, Counting Sort) study (II) (2015-06-09 19:14)

May 10, 2015

Problem statement:

Given an unsorted array, find the maximum difference between the successive elements in its sorted form.

Try to solve it in linear time/space.

Return 0 if the array contains less than 2 elements.

You may assume all elements in the array are non-negative integers and fit in the 32-bit signed integer range.

□□□□□□□□□□□□□B□ (O(nlogn), □□, □□□□□□□□□□. □□, □□□n□□, □□□□□□□□□□, □□, □□□□□□□□□□□□□□. □□, □□□□□□□□□□□□; □□□□□□□□□□□□□. □□□□□, □□□□, □□□.

Read the article:

http://cgm.cs.mcgill.ca/ godfried/teaching/dm-reading-assignments/Maximum -Gap-Problem.pdf

And then,

http://yuanhsh.iteye.com/blog/2187685

http://www.programcreek.com/2014/03/leetcode-maximum-gap-java/

http://zh.wikipedia.org/wiki/ %E6 %A1 %B6 %E6 %8E %92 %E5 %BA %8F

Practice the code and then figure out things learned through the process.

May 22, 2016

Read the blog, and copy the analysis from the blog as well:

http://yuanhsh.iteye.com/blog/2187685

□□□□□□O(nlgn)□□□□□-à□□□□□□□□□□gap□□□□□□□□□□O(n)□□□□□□□□-¦□□□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□□□□□□□□□□□ç□□□- □□□□ □□□□□□□□-□□□□—□v□n, a b□gap−ceiling(b−a)/(n−1), □−¦ceiling(b−a)/(n−1)□gap□□-ceiling(b − a)/(n − 1)□□□□b − a)/[ceiling(b − a)/(n − 1)] + 1□ − □□—□□□□□□□□□□□□□□□č□□□O(n),□□□□□□□O(n)□

Read the blog, and also like the idea to use Radix sort - check number from insignificant digit to most significant digit:

https://helloyuantechblog.wordpress.com/2015/11/17/leetcode-164-maximum-gap -hard/

Read the blog, have some code using C # in short future:

http://www.geeksforgeeks.org/radix-sort/

C # practice based on the above blog:
https://gist.github.com/jianminchen/0da5c79679d26f4682af328c922ccca9

1.1.28 Read blogs and study (2015-06-09 19:14)

May 9, 10, 2015

Rabin-karp algorithm - □□□□□

Read several websites about string search algorithms. □□□□□□□, □□□□□□□□□□□□□.

http://www.julyedu.com/video/play/id/32

Boyer-moore algorithm:

http://www.ruanyifeng.com/blog/2013/05/boyer-moore _string _search _algorithm.html

Read the article about Rabin-karp algorithm.

http://www.cs.princeton.edu/courses/archive/spr04/cos226/lectures/strin g.4up.pdf

1.1.29 Leetcode: Median of two sorted arrays (2015-06-09 19:15)

Problem statement: There are two sorted arrays A and B of size m and n respectively. Find the median of the two sorted arrays. The overall run time complexity should be O(log(m+n)).

May 17, 2015

□□□□□□□□□□□

,

□-

□□□□

,

□□□

□□□□□□□□□

Leetcode

□

□□

,

□

□□□□-¤□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□

□□□

□□□□

:

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□-чŽ□□□□□□□□□□□

□

□□□□

k

20

□□□□□

1. O(m + n)

□□□□

□□□□□□

merge

□□□

□□□□□□

k

□□□□□

2. O(k)

□□□

O(1)

□

□

;

□□□□

k

□□□

m + n

□

□□□□□□□□□

O(m + n)

□□

□

□□□□□□□□

k

□□□□□□□□□

“

□□

”

□□□□□□□□□□□□□□□□□□□

□

□□□□□□□□□

m

□□□□□□□

□□□□□□□□□□

pA

□

pB

□□

□□□

A

□

B

□

□□

□

□□□□□□□

□□□□

merge sort

□□□□□□□

□

A

□□□□□□□□

pA++

□□

□

m++

□□□

□

□

B

□□□□□□□□

pB++

□□

□

m++

□

□

□□

22

m

□□

k

□

□□□□□□□□□□□□□□□

O(k)

□□□

O(1)

□

□□□□□□

k

□□□

m + n

□

□□□□□□□□□

O(m + n)

□□

3.

□□□□□

□□□□□□□□□□□□□

□□□□□□

k

□□□□□□

□□□□□□□□□□□□□□□□□

k

□□

□□□□□□□□□□

□□□□□□

k

□□□□□□□□□

□□□□□□□□□□□□

A

□

B

□□□□

□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

“

□□

”

□-□

□

□

A

□

B

□□□□□□□□□

k/2

□□

□□

A

□□

k/2

□□□□□

A[k/2-1]

□□

B

□□

k/2

□□□□□

B[k/2-1]

□

□□-□□□□□□□□□□□□□□□□□□□□□□□□

k

□□□□□□□□□□

□□□

k

□□□□□□□□□□□

• A[k/2-1] == B[k/2-1]

• A[k/2-1] > B[k/2-1]

• A[k/2-1] < B[k/2-1]

□□

A[k/2-1] < B[k/2-1]

□□□□

A[0]

□

A[k/2-1]

□□□□

A [B]

□

top k

□□□□

□

□□

□□□□□

A[k/2-1

□□□□□

A [B

□□

k

□□□□

□

□□□□□□

□□□□

□□□□□□□□□

A

□

□□□

k/2

□□□□□□□□

A[k/2-1] > B[k/2-1]

□□□□□□

B

□

□□

k/2

□□□□

□

A[k/2-1] == B[k/2-1]

□□□□□□□□

k

□□□□□□□□□

A[k/2-1]

□

B[k/2-1]

□□□

□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□

□

A

□

B

□□

□□□□□□

B[k-1]

□

A[k-1]

□

•

□

k=1

□□□□

min(A[0], B[0])

□

•

□

A[k/2-1] == B[k/2-1]

□□□□

A[k/2-1]

□

B[k/2-1]

□□

Read some blogs:

http://articles.leetcode.com/2011/01/find-k-th-smallest-element-in-union-of .html

http://fisherlei.blogspot.ca/2012/12/leetcode-median-of-two-sorted-arrays.htm l

Practice code sharing: C # code:

https://github.com/jianminchen/medianOfTwoSortedArrays/blob/master/Program. cs

1.1.30 web programming videos (2015-06-09 19:15)

Study those videos, and get started to work on sQL server, sQL server solution, and the C # programming languages
http://channel9.msdn.com/Browse/Series
.NET framework

MVC - study the framework

http://channel9.msdn.com/Series/Introduction-to-ASP-NET-MVC
Financials in Microsoft Dynamics SL 2015

Twenty C # questions explained

Watch it first, Feb. 5, 2015

Fundamental Javascript closures

http://channel9.msdn.com/Series/Javascript-Fundamentals-Development-for-A bsolute-Beginners/Fundamentals-of-JavaScript-Closures-20

Feb. 5, 2015 - watch the video.

Javascripts for fundamental

http://channel9.msdn.com/Series/Javascript-Fundamentals-Development-fo r-Absolute-Beginners?page=3

1.1.31 Build a career, have some coaching... (2015-06-09 19:15)

May 11, 2015

First time I read the article, and then, strongly agree that the argument makes good sense. Great developer is rarely on the market.

http://www.joelonsoftware.com/articles/FindingGreatDevelopers.html

So, I am so relax, and enjoy myself to stay as is.

May 12, 2015

great website:

https://github.com/justicezyx/puzzles

https://plus.google.com/105852295213412135941/posts

January 7, 2016

Hangout on Air: Candidate Coaching Session: Tech Interviewing

https://www.youtube.com/watch?v=oWbUtlUhwa8

Hangout on Air: Candidate coaching session: Tech Interviewing

– 45 minutes interview –

Engineering skills Assessment

1. write good code
2. Algorithm and data structure
3. Analysis skills
4. Sound design

We want to see you can write good code

How to apply to the problem?

How to approach the bits?

Sound design in your code?

limitation or corner case

More detail about “Analytical skills” - make note - do this kind of checks

More detail about “Sound design”

Need to understand what exactly can go wrong.

How to prepare your interview?

Go through the questions, the interviewer will go over

Advices from coaching:

1. Practice on white board or paper
2. Don’t use pseudo code - we want actual code
3. Interviewer will ask which language you prefer

And, the best of advice Julia likes:

Write a lot of code - no better way to write a lot of code

Practice to write on a white board

Like to see the code will be neat, and will work

We don’t want the pseudo code - which ever main stream language you will use

We really are interested in how you approach the problem.

Think about loud, no idea if you understand the problem, walk through the problem.

Easy on guiding you through,

Questions are in-depth

The questions might have simple solution. A lot of them are set up, simple, but we expand to cover other situation.

Show them how to approach, and how to expand the problem.

Nice answer is kind of relative. Will not criticize you

Simple solution you can start, given the assumption, we can change the assumption to simple, or complex, and then, the problem can go on and on.

Sample interview question :

"Write a program that breaks up a string of words with no spaces into a string with the appropriate ...

———————————— one slide

Clarify the problem

. Consider an example that's rich enough but not tedious:

"fastman" -> "fast man"

Disambiguous expected result:

More than two words? Empty string?

. State and clarify key assumptions: expected result, any memory or performance requirements, etc.

Where do the words come from? Use a dictionary

. Clarify the function signature

———————————— next slide

it is not best solution, but we can go from there - a fair straightforward solution

brute force solution - nxn solution - not good solution but solve the problem

—

Working through the problem - part 2

Advices:

Start with the first solution that comes to mind

. Feel free to say that first solution will be refined later.

. This will usually be a brute force solution

. Run through at least one or two examples to check for correctness

. Check again for edge cases

. Try to use reasonable variable names or clean up the code after the first pass

. Ask if the interviewer has any question before refinement

Advice:

Allow the interviewer to ask question.

Working through the problem - part 3

Refine the solution

. Clarify assumptions (e.g., improving performance)

. Rinse, repeat

. Compare the solution

Working through the problem - part 3

Refine the solution

. Allow for an arbitrary number of words

"applesauceman" -> "apple sauce man"

"iamdoingwell" -> "I am doing well"

Multiple solutions of greatly different complexity!

What if the dictionary cannot fit in memory?

How would you print the most likely of multiple choices?

What if the words are not spelled correctly?

Advice:

The interview should be fun. Two engineers are discussing the problem. How to solve the problem? How to make it better?

Example solution

—————–

Tips

Semi brute force solution:

—

Tips

optimizing

. Talk through naive solution(s). see if you can come up with a reasonably efficient one.

. Time/space tradeoffs

. Assumptions: what if the input is big? small?

API

. If you can't remember the name of a method, or how % behaves with negative numbers, ask!

Testing

. Say what cases would you test

. Try to reserve time to work small case (iron out boundary conditions, etc.)

More interesting

Q &A

Few questions to go over:

1. Do Google consider culture difference? Every googler learns the culture.

Speaking English is a plus

2. Google learns how to train interviewer.

Strict rules - what to look for:

You observe them, and they observe you. Two people show up, an interviewer is under training. One will be quiet in the corner.

3. Ask questions part

- initial part - break ice

- try to make them feel more relax

- become conversational

- try to help them out

- what angle you look at

- start to talk about the problem -

People communicate the ideas

- We do understand the …

- Julia's comment: write down more on this

4. C++, Java,

/ Certification - does not matter

Just good coding skills - show case in five interviews

(not C # for Google, Julia, is time to learn advanced C++, and read more C++ code/ Java code)

5. Tech interview

We want you how you solve the problem, how you think;

First a few months, relearn everything at Google. So, it is important to show you how to learn things, solve problems.

ramp up - weeks, we are talking about months.

6. Interviewer how to choose questions?

Mock interview

A set of questions I used, helps me to help you

I understand the question, some of them goes away the rail; we are not talking about it today.

They already see all the solutions; do not want you go down some way, waste all of us time.

Understand the problem, know math,

Advice:

We want you to prepare. Preparation is a big deal . Multiple interviews a day.

Binary tree / Binary search tree / Just a conversation / a peer / Last but least / delay and reply

Interviewer - hard to say - recruiter will tell you what's going on.

During the interviewer, the interviewer will take some notes.

June 6, 2016

https://www.linkedin.com/pulse/5-simple-rules-success-new-job-hamid-al-azzawe ?trk=hp-feed-article-title-like

1.1.32 Leetcode□□□□ (2015-06-09 19:16)

□□□□□□□□□□□, □□□□□; □□□□□□□□, □□□□□□□, □□□□; □□□□□. □□, □□□□□□□□□□□□, □□□□. □□□□□□□, □□□□□.

□□□□□, □□□□□□□, □□□□. □□, □□□□□□□□.

http://www.julyedu.com/video/index

May 23, 2015

http://www.julyedu.com/video/play/id/28

□□□□□□□□□, Leetcode question 3 □□□□□□□□□□.

Recently, I was too busy at work, and go to play tennis after work; I did not have chance to review Leetcode questions, felt frustrated, and my body weight goes up; difficult to climb stairs from first floor to 3rd floor. So, back to normal, go to bed early and then get up early. Now, it is the time to start to work on, and learn more things about coding.

My favorite video is to talk about string manipulation. □□□□□□□□□□□□□□□, □□□□□□□□□□□□□. □□□□□□:

□□: □□□□□, in-place O(1) space, □□

□□O(1)□□

□□, □□□□□□□□□□□

□□□□□□□□□□:

□□□□□□□□, □□□□

□□partition□□

□□□□□ա

□□□O(n)□□□□□□□□

O(1)□□□□□□□□

□□□□□

□□(□□): KMP□□□□□

Manacher - □□□□□□□□□□

1.1.33 Leetcode: strstr function (2015-06-09 19:16)

problem statement:

Returns a pointer to the first occurrence of needle in haystack, or **null** if needle is not part of haystack.

May 17, 11:30pm

□□□□□□□□□□□□□.

http://bangbingsyb.blogspot.ca/2014/11/leetcode-implement-strstr-kmp.html

http://fisherlei.blogspot.ca/2012/12/leetcode-implement-strstr.html

1.1.34 Leetcode: Binary tree preorder traversal (2015-06-09 19:17)

Note: Recursive solution is trivial, could you do it iteratively?

May 26, 2015 - Read the website:

http://en.wikipedia.org/wiki/Tree _traversal

Read the leetcode solution in C++ first:
class Solution {
public:
vector preorderTraversal(TreeNode *root) {
vector result;
const TreeNode *p;
stack s;

p = root;
if (p != nullptr) s.push(p);
while (!s.empty()) {
p = s.top();
s.pop();
result.push _back(p->val);
if (p->right != nullptr) s.push(p->right);
if (p->left != nullptr) s.push(p->left);
}
return result;
}
};

Here is the C # code:

https://github.com/jianminchen/leetcode-tree/blob/master/TreePreOrderTravers al.cs

Morris preorder:

http://www.geeksforgeeks.org/inorder-tree-traversal-without-recursion-an d-without-stack/

original paper (read it if I need to challenge my understanding)

http://www.sciencedirect.com/science/article/pii/0167642388900639

The explanation in the following blog is more clear compared to Leetcode solution.

http://codingrecipies.blogspot.ca/2013/11/morris-inorder-traversal.html

https://plus.google.com/+PrateekRathore/posts

Here is the pseudo code:

```
preorder(node)
  if node == null then return
  visit(node)
  preorder(node.left)
  preorder(node.right)
```

□□□□□□, □□□□□□□□□□, □□□□□□□□□□.

```
iterativePreorder(node)
  parentStack = empty stack
  while (not parentStack.isEmpty() or node   null)
    if (node   null)
      visit(node)
      if (node.right   null) parentStack.push(node.right)
      node = node.left
    else
      node = parentStack.pop()
```

1.1.35 Leetcode video time (2015-06-09 19:17)

May 24, 2015

Spent time to watch the video about stack. Very good video. on May 24, 2015

http://www.julyedu.com/video/play/id/30

There is a lecture to explain the question to find last k number maximum number.

Look up Catalan combination problem.

http://en.wikipedia.org/wiki/Catalan _number

http://www.geeksforgeeks.org/k-largestor-smallest-elements-in-an-array/

1.1.36 Binary tree algorithms (2015-06-09 19:18)

May 28, 2015

To expand my knowledge, challenge my understand of binary tree, read through over 1000 lines Java code, and then, memorize all the algorithms. □□□□, □□□□□□□□□□□□, □□, □□□□□□□□□□□□□. □□□□□□, □□□; □□□□, □□□□□□□□□, □□□□□□□□□.

http://blog.csdn.net/fightforyourdream/article/details/16843303

□□□, □□□□□□□□, □□□□□□:

1. □□□□BST□□□□□□(DLL)
2. □□□□□□□□□□□□□
3. □□□□□□□□□□
4. □□□□□□□□□□□□□□ □□□□□□ O(n) (□□□□□□LevelOrderTraversal□□□□□□□□Queue)
5. □□□□□□□□□□□□□□□□□□ □□□□□ (□□inorder traversal□□□)

1.1.37 Leetcode: Post order binary tree traversal iteratively (2015-06-09 19:18)

May 28, 2015

Problem statement:

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

For example: Given binary tree {1, #,2,3 },

1

\

2

/

3

return [3,2,1].

Note: Recursive solution is trivial, could you do it iteratively?

May 27, 2015

□□□□□□□□

,

□□□

□□□□□□□□

;

□

□□□□□

,

□

□□□□□□□

;

□□□□□□□□□□□□□

,

□□

,

□

□□

,

□□□□

□

.

http://codeganker.blogspot.ca/2014/03/binary-tree-postorder-traversal-leetc ode.html

http://blog.sina.com.cn/s/blog _eb52001d0102v1si.html

□□□□

EMC

□□□□□□□□□

,

□

□□□□□□□□□

.

https://github.com/yuzhangcmu/LeetCode/blob/master/tree/TreeDemo.java

□□□

,

□□

C #

□□

□□□□

,

□□□□□

□□□□

.

http://blog.csdn.net/fightforyourdream/article/details/16843303

https://github.com/yuzhangcmu/LeetCode

□□□□

:

http://www.cnblogs.com/yuzhangcmu/p/4199673.html

LeetCode / LintCode □□ □□

http://www.jiuzhang.com/solutions/?source=githubyz

http://www.weibo.com/3948019741/profile?topnav=1 &wvr=6

July 7 2015

Share C # code:

https://github.com/jianminchen/leetcode-tree/blob/master/TreePostOrderIterat ive.cs

1.1.38 Morris inorder traversal - binary tree (2015-06-09 19:18)

May 26, 2015

Problem Statement:

Print inorder traversal of the tree without using extra space (Morris Traversal)

Study two websites:

1. http://codingrecipies.blogspot.ca/2013/11/morris-inorder-traversal.ht ml

Implement the code using C #, and use the same example in the above website.

2. http://www.geeksforgeeks.org/inorder-tree-traversal-without-recursion- and-without-stack/

Here is the short description to understand the algorithm:

Concept Involved:

Morris traversal is an implementation of in-order traversal that uses threading:

1.

Create links to the in-order successor

2.

Print the data using these links

3.

Revert the changes to restore original tree.

Here is the C # code:

https://github.com/jianminchen/MorrisInOrderTraverse/blob/master/Program.cs

read the blog (July 9, 2015):

http://www.cnblogs.com/AnnieKim/archive/2013/06/15/MorrisTraversal.html #3171669

1.1.39 Binary tree maximum distance, diameter of binary tree (2015-06-09 19:19)

May 29, 2015

Diameter of binary tree: The diameter of a tree (sometimes called the width) is the number of nodes on the longest path between two leaves in the tree.

Read two websites:

http://www.cnblogs.com/miloyip/archive/2010/02/25/1673114.html

http://www.geeksforgeeks.org/diameter-of-a-binary-tree/

copy Java code solution from the website:

http://blog.csdn.net/fightforyourdream/article/details/16843303

```
1. /**
2. * ?????????????? ??????????????????-?????? (distance / diameter)
3.  * ??????
4. * ?1?????????????0???????-þ?????????????0
5.  * ?2?????????????????????????????-?ƨ?????????????ƨ?
6. * ??????????-Ц9?????????+???????????????
7. * ?Ќ?????????????????????????
8. *
9.  * http://www.cnblogs.com/miloyip/archive/2010/02/25/1673114.html
10. *
11. * ???????????????-ъÅ?:
12.
13. ??A: ???????????????????????????????
14. ??B: ???????????-ч???????????????????????
15. ??????H?????????????????????????????-?ƨ
16. */
17. public
    static Result getMaxDistanceRec(TreeNode root) {
18. if (root == null ) {
19. Result empty = new Result( 0 , - 1 ); // ???????? +1 ?Ў?????????? (NULL) ?????????? 0
20. return empty;
21. }
```

```
22.
23. // ??????????-
24. Result lmd = getMaxDistanceRec(root.left);
25. Result rmd = getMaxDistanceRec(root.right);
26.
27. Result res = new Result();
28. res.maxDepth = Math.max(lmd.maxDepth, rmd.maxDepth) + 1 ; // ???????
29. // ???A???B????
30.     res.maxDistance = Math.max( lmd.maxDepth+rmd.maxDepth, Math.max(lmd.m axDistance,-
    rmd.maxDistance) );
31. return res;
32.  }
33.
34. private
    static
    class Result {
35. int maxDistance;
36. int maxDepth;
37. public Result() {
38. }
39.
40. public Result( int maxDistance, int maxDepth) {
41. this .maxDistance = maxDistance;
42. this .maxDepth = maxDepth;
43.  }
44. }
```

1.1.40 binary tree algorithms (2015-06-09 19:19)

□□□□, □□□□□□; □□□, □□□, □□□□. □□□□□□□□□□□□□□□, □□□□□□□□□□, □□□□□□□□□□. □□□□□, □□, □□□□□□□□, □□□□□□□□. □□, □□□□, □□□□□□□□□□□□□□□м, □□. □□□, □□C #□□□□□□, □□Github, □□□□□□.

http://blog.csdn.net/luckyxiaoqiang/article/details/7518888

□□□□□□□:

http://poj.org/problemlist

□□

http://blog.csdn.net/luckyxiaoqiang/article/details/7393134

□□□□□□□□□□□□□□

http://blog.csdn.net/luckyxiaoqiang/article/details/8937978

1.1.41 Leetcode: scramble string (2015-06-09 19:20)

June 4, 2015

Leetcode problem:

https://leetcode.com/problems/scramble-string/

https://github.com/soulmachine/lintcode

https://helloworld-chasecoder.rhcloud.com/?p=21

http://blog.unieagle.net/2012/10/23/leetcode %E9 %A2 %98 %E7 %9B %AE %EF %BC %9Ascramble-string %EF %BC %8C %E4 %B8 %89 %E7 %BB %B4 %E5 %8A %A8 %E6 %80 %81 %E8 %A7 %84 %E5 %88 %92/

□□□□□□□□, □□□□□□□□□□□□□□□□y□□□□□; □□□□, □□, □□; □C # □□□□□. □□□□□□□□□□□□□□□□□-□, □N□□□□.

C # function:

https://github.com/jianminchen/scramble-string/blob/master/Program.cs

July 9, 2015 (2nd round study)
Totally forget what the problem is about. And by mistake I guessed that scramble strings is all permutations of string (n!), actually it is O(3^n).

Read the blog:
http://www.cnblogs.com/yuzhangcmu/p/4189152.html

plan to add the recursive solution plus memorization using C #.

1.1.42 Leetcode: Divide Two Integers (2015-06-09 19:20)

June 4, 2015

this website provided a very clear explanation using math formula. Great explanation:

http://www.acmerblog.com/leetcode-divide-two-integers-5977.html

http://blog.csdn.net/linhuanmars/article/details/20024907

C # code:

```
/*
first time using uint in C # - June 4, 2015
The sbyte data type is an 8-bit signed integer.
The byte data type is an 8-bit unsigned integer.
The short data type is a 16-bit signed integer.
The ushort is a 16-bit unsigned integer.
The int data type is a 32-bit signed integer.
The uint is a 32-bit unsigned integer.
The long data type is a 64-bit signed integer.
The ulong is a 64-bit unsigned integer.
The char data type is a Unicode character (16 bits).
The float data type is a single-precision floating point.
The double data type is a double-precision floating point.
The bool data type is a Boolean (true or false).
The decimal data type is a decimal type with 28 significant digits (typically used for financial purposes).

*/
static int divide3( int dividend, int divisor) {
int result = 0;
bool sign = (dividend > 0 & & divisor < 0) || (dividend < 0 & & divisor > 0);

uint a = (uint)(dividend >= 0 ? dividend : -dividend);
uint b = (uint)(divisor >= 0 ? divisor : -divisor);

/*
Try to figure out what we are doing in loops:

/
while (a >= b) {
int multi = 1;
uint bb = b;

while (a >= bb) {
a -= bb;
result += multi;
if (bb < int.MaxValue >> 1)
{
bb += bb;
multi += multi;
}
}
}

if (sign) return -result;
else return result;
}
```

1.1.43 Algorithm study (2015-06-09 19:20)

May 31, 2015

Spent two hours to read the webpage:

http://blog.csdn.net/luckyxiaoqiang/article/details/7393134

watch the video:

https://www.youtube.com/watch?v=EjD5PJJoeLU

Start to practice coding and share it on github, first thing to write is about binary tree:

https://github.com/jianminchen/leetcode-tree/blob/master/TreeDemo.cs

1.1.44 Leetcode: blogs reading time (2015-06-09 19:21)

on June 5, 2015

Read most popular leetcode blogs:

http://www.cnblogs.com/lichen782/p/leetcode _maximal _rectangle.html

http://www.cnblogs.com/lichen782/p/leetcode _minimum _window _substring _3.html

http://www.cnblogs.com/lichen782/p/leetcode _Largest _Rectangle _in _Histogram.html

http://www.cnblogs.com/lichen782/p/leetcode _interleaving _string.html

http://www.cnblogs.com/lichen782/category/494719.html

1.1.45 Leetcode: Interleave string (2015-06-09 19:21)

on June 5 - 6 , 2015

In my preference order, read those websites, and then, figure out ways to improve my understanding the problem.

http://blog.sina.com.cn/s/blog _eb52001d0102v2h1.html

http://www.cnblogs.com/lichen782/p/leetcode _interleaving _string.html

http://blog.unieagle.net/2012/09/29/leetcode %E9 %A2 %98 %E7 %9B %AE %EF %BC %9Ainterleaving-string %EF %BC %8C %E4 %BA %8C %E7 %BB %B4 %E5 %8A %A8 %E6 %80 %81 %E8 %A7 %84 %E5 %88 %92/

http://sbzhouhao.net/2014/08/27/LeetCode-Interleaving-String-in-Recursion/

http://fmarss.blogspot.ca/2014/08/leetcode-solution-interleaving-string.htm l

http://shanjiaxin.blogspot.ca/2014/03/interleaving-string-leetcode.html

http://rleetcode.blogspot.ca/2014/01/interleaving-string-java.html

1.1.46 Leetcode: review time (2015-06-09 19:22)

June 8, 2015

Learning to write code is kind of tough process. Mental part, I am still very weak in the challenges. So, train myself to be a talent programmer, take leetcode questions as my projects to stay focus. The benefit of practising leetcode

question is to allow myself to expose top talent programmers in China. So, reading time again for those leetcode questions, 5 -10 questions to review, and then, learn. ??????????, ???????????????. ?????, ???????.

https://www.gitbook.com/book/siddontang/leetcode-solution/details

http://siddontang.gitbooks.io/leetcode-solution/content/array/maximal _rectangle.html

http://siddontang.gitbooks.io/leetcode-solution/content/array/searc h _a _2d _matrix.html

http://siddontang.gitbooks.io/leetcode-solution/content/array/largest _rectangle _in _histogram.html

http://siddontang.gitbooks.io/leetcode-solution/content/dynamic _programming/best _time _to _buy _and _sell _stock.html

1.1.47 Leetcode: Candy (2015-06-09 19:22)

Problem statement:

There are N children standing in a line. Each child is assigned a rating value. You are giving candies to these children subjected to the following requirements:

1. Each child must have at least one candy.
2. Children with a higher rating get more candies than their neighbors.

What is the minimum candies you must give?

June 8, 2015

Read the websites in my favorite orders:

1. http://fisherlei.blogspot.ca/2013/11/leetcode-candy-solution.html
2. http://www.programcreek.com/2014/03/leetcode-candy-java/
3. http://blog.csdn.net/linhuanmars/article/details/21424783
4. http://www.cnblogs.com/lichen782/p/4300325.html
5. http://siddontang.gitbooks.io/leetcode-solution/content/greedy/candy.htm l

Analysis from website 2

This problem can be solved in O(n) time.

We can always assign a neighbor with 1 more if the neighbor has higher a rating value. However, to get the minimum total number, we should always start adding 1s in the ascending order. We can solve this problem by scanning the array from both sides. First, scan the array from left to right, and assign values for all the ascending pairs. Then scan from right to left and assign values to descending pairs.

This problem is similar to Trapping Rain Water.

Most important, play with code; Spent 30 minutes to work on the coding using C # (change from Java code on webpage 2), and also, the code passes leetcode online judge.

https://github.com/jianminchen/candy/blob/master/Program.cs

1.1.48 Binary Tree: Write a function to return count of nodes in binary tree which has only one child. (2015-06-09 19:23)

June 8, 2015

□□□□□□□□□□□, □□□□□□.□□□□□□□?□

□□□□□□□□□□□□□□□□□. 2014□□□, □□□□□□□□□□□□□□□□□□□, □□□□□□□□□□, argue, change, using logic reasoning and try to make it minimal. □□□□bug, □□□; □□, □□□□□□□, □□□□□□.

□□□□□:

1. Local variable in each recursive function
2. Redundancy code
3. If statement, how many if statement in the function
4. How to argue that the code will not have bugs? Can you simplify the code?
5. Make the code simple as possible, no way to hide any bugs
6. Base case discussion
7. Discussion if null checking is needed to call the function.
8. Use a global variable to store the count.
9. Function arguments - what arguments are needed.

2015□□, □□□Java Script □□, □□□□, C #, Leetcode; □□□□□□□□□□.

Github for source code:

https://github.com/jianminchen/BinaryTreeOneChild/blob/master/Program.cs

□□□□□□□, □□□□□□□□□:□ (node.left!=null)!=(node.right!=null)

```
public static int countOneChildNode(Node node)
{
if(node==null) return 0;
return (((node.left!=null)!=(node.right!=null)? 1:0) +countOneChildNode(node.left)+countO-
neChildNode(node.right));
}
```

1.1.49 Leetcode: Largest rectangle in histogram (2015-06-09 19:23)

Given *n* non-negative integers representing the histogram's bar height where the width of each bar is 1, find the area of largest rectangle in the histogram.

June 8, 2015

My favorite websites about this problem:

http://fisherlei.blogspot.ca/2012/12/leetcode-largest-rectangle-in-histog ram.html

□□□□□□□, □□□□□□□□□□□□, □□□□□. □□C #□□□□□□.

http://siddontang.gitbooks.io/leetcode-solution/content/array/largest _rectangle _in _histogram.html

Here is my practice:

https://github.com/jianminchen/largestRectangleInHistogram/blob/master/P rogram.cs

June 30, 2015
Totally forget the problem, and then, need second round study.

Read the blog:
http://segmentfault.com/a/1190000002673098

1.1.50 Leetcode: strstr function (2015-06-10 19:24)

June 10, 2015

http://blog.csdn.net/linhuanmars/article/details/20276833

http://zjalgorithm.blogspot.ca/2014/12/leetcode-in-java-implement-strstr.ht ml

Understand the Boyer-Moore Algorithm using test cases

Try to pass online judge

C # code:

https://github.com/jianminchen/stringDemo/blob/master/Program.cs

1.1.51 Leetcode: Power(x,n) (2015-06-10 19:25)

June 10, 2015

This question is most popular question in terms of my personal experience.

http://fisherlei.blogspot.ca/2012/12/leetcode-powx-n.html

http://fisherlei.blogspot.ca/2012/12/leetcode-powx-n.html

http://yucoding.blogspot.ca/2013/03/leetcode-question-74-powxn.html

best blog I have seen:

power(x,n)

http://rleetcode.blogspot.ca/2014/02/powx-n-java.html

http://www.lilongdream.com/

http://www.lilongdream.com/2014/04/10/94.html

http://www.shuatiblog.com/blog/categories/leetcode/

check this blog and ensure that it makes good sense to read it.

http://www.shuatiblog.com/blog/2014/05/28/Best-Time-to-Buy-and-Sell-Stock-III /

http://codingtnnd.blogspot.ca/2014/09/powx-n-leetcode.html

Share C # code:

https://github.com/jianminchen/powerOfN/blob/master/Program.cs

1.1.52 Read blogs and study (2015-06-10 19:25)

June 10 2015

Came cross this blog, and chose to read 10 leetcode solutions to expedite my practice on leetcode questions. Took more than 6 months, a lot of questions are new to me. So, jump into 10-20 questions new first time:

Iterative tree travesal

Palindrome number

gas station

1.1.53 Leetcode: Sprial Array printout (2015-06-13 08:23)

June 11, 2015

A few of website to read

one recursive solution:

http://gongxuns.blogspot.ca/2012/12/leetcode-spiral-matrix.html

http://fisherlei.blogspot.ca/2013/01/leetcode-spiral-matrix.html

one iterative solution:

http://siyang2leetcode.blogspot.ca/2015/03/spiral-matrix.html
http://blog.csdn.net/linhuanmars/article/details/21667181

C # code to pass Leetcode online judge:

https://github.com/jianminchen/spiralArrayPrintout/blob/master/Program.cs

1.1.54 Leetcode 5: longest palindromic substring (2015-06-15 22:54)

Problem statement:

Given a string S, find the longest palindromic substring in S.

June 15, 2015

Websites to read:

http://www.programcreek.com/2013/12/leetcode-solution-of-longest-palindromic -substring-java/

https://yanbxice.wordpress.com/2015/05/31/leetcode-longest-palindr omic-substring/

□□□□□□□□, □□□□!

https://leetcodenotes.wordpress.com/tag/palindrome/

C # code on github:

https://github.com/jianminchen/palindrome/blob/master/Program.cs

https://github.com/jianminchen/Leetcode _C-/blob/master/5LongestPalindromicSubstring.cs

https://github.com/jianminchen/Leetcode _C-/blob/master/5LongestPalindromic _B.cs

January 2, 2016

Review the algorithm.

Leetcode question:

5. Longest Palindromic Substring

https://github.com/jianminchen/Leetcode _C-/blob/master/5LongestPalindromicSubstring.cs

January 13, 2016

Read the blog:

http://blog.csdn.net/linhuanmars/article/details/20888595

http://blog.csdn.net/linhuanmars/article/details/22777711

Please read 5-6 solution about this leetcode question, and then, collect all the wisdom. Try to have nice memory about the solution, some fun experience; therefore, it will be a quick and fun time to solve the similar problems in the future.

Do not rush to finish more leetcode questions. Try to focus on simple problems.

There are 5 solutions discussed in the following blog, most important is to give overview of 5 analysis, what is brute force solution - time, space complexity.

One way to make review more fun is to read more than 10 - 20 solutions, and know all the ideas out there, and then, write some code; Reading is most important to help understand algorithms, through a simple problem, common interview question.

http://articles.leetcode.com/2011/11/longest-palindromic-substring-part- i.html

4 solutions in the above blog

one optimal solution - linear time solution in the following blog:
http://articles.leetcode.com/2011/11/longest-palindromic-substring-part-ii.h tml

This blog is in Chinese. Excellent! Try to summary a few tips to come out linear time optimal solution! Do something to get more involved.

http://www.felix021.com/blog/read.php?2040

spent 10 minutes to read the article,
https://www.akalin.com/longest-palindrome-linear-time

Read the blog:
http://www.acmerblog.com/leetcode-longest-palindromic-substring-5356.html

6th solution, a suffix tree solution:
http://www.allisons.org/ll/AlgDS/Tree/Suffix/

1.1.55 Leetcode: invert a binary tree (2015-06-15 23:15)

Invert a binary tree

http://www.wengweitao.com/leetcode-invert-binary-tree.html

http://mojijs.com/2015/06/196603/index.html

Binary tree upside down

http://bangbingsyb.blogspot.ca/2014/11/leetcode-binary-tree-upside-down.html

Github source code:

https://github.com/jianminchen/leetcode-tree/

June 15, 2015

Write C # code

```
/**
* Latest update: on June 16, 2015
* Leetcode:
* Invert a binary tree
* Reference:
* http://www.zhihu.com/question/31202353
*
* 7 lines of code - using recursion
*/
public static Node invertBinaryTree(Node root)
{
if (root == null)
return null;
Node temp = root.left;
root.left = root.right;
root.right = temp;
invertBinaryTree(root.left);
invertBinaryTree(root.right);
return root;
}
/**
* Latest update: on June 16, 2015
* Leetcode: Invert a binary tree
* using iterative solution
*/
public static Node invertBinaryTreeIterative(Node root)
{
if (root == null)
return null;
```

```
Queue q = new Queue();
q.Enqueue(root);
/*
* consider the queue:
*/
while (q.Count > 0)
{
Node nd = (Node)q.Dequeue();
Node tmp = nd.left;
nd.left = nd.right;
nd.right = tmp;
if (nd.left != null)
q.Enqueue(nd.left);
if (nd.right != null)
q.Enqueue(nd.right);
}
return root;
}
```

1.1.56 Leetcode: edit distance (2015-06-17 20:03)

June 16, 2015

http://blog.csdn.net/fightforyourdream/article/details/13169573

http://web.stanford.edu/class/cs124/lec/med.pdf

□□□-□æ□

2 dimensional dynamic programming.

□□□□□□□

,

□□□□□

,

□□□□□

;

□□□□□□□□□

.

□□□□□□□□□

,

Share the C # code:

https://github.com/jianminchen/edit-distance/blob/master/Program.cs

1.1.57 Leetcode: string function atoi (2015-06-19 23:58)

June 18, 2015

Figure out how people are smart to get organized. The following blog is well organized.

http://pisxw.com/tag/ #blog

atoi function blog:

http://blog.csdn.net/ljiabin/article/details/40508889

http://blog.csdn.net/ithomer/article/details/8800530

http://blog.csdn.net/column/details/leetcode-solutions.html

interesting stuff:

http://hihocoder.com/services/campus-recruitment

Share C # practice:

https://github.com/jianminchen/atoi/blob/master/Program.cs

1.1.58 Leetcode: word ladder (2015-06-20 00:00)

June 19, 2015

Problem statement:

Word Ladder I

Given two words (*start* and *end*), and a dictionary, find the length of shortest transformation sequence from *start* to *end*, such that:

1. Only one letter can be changed at a time
2. Each intermediate word must exist in the dictionary

For example,

Given:

start = "hit"

end = "cog"

dict = ["hot","dot","dog","lot","log"]

As one shortest transformation is "hit" -> "hot" -> "dot" -> "dog" -> "cog",

return its length 5.

Note:

- Return 0 if there is no such transformation sequence.
- All words have the same length.
- All words contain only lowercase alphabetic characters.

http://bangbingsyb.blogspot.ca/2014/11/leetcode-word-ladder-i-ii.html

http://shanjiaxin.blogspot.ca/2014/04/word-ladder-leetcode.html

□□□□□□□-ʞʌu

,

□□□□□□□□□□

.

□□□□□□□

,

□□□□□□

.

□□□□□

,

□□□□□□□

.

□□□□□□□□□□□□

,

□□□

,

□□□□□

.

□□□□□

,

□□□□

,

□□□□□□

.

□□□□□□□

:

□□□

LeetCode

□□□□□□□□-□□□□□□□□□□□□□□□□□□□□□□□□

“shortest transformation sequence from start to end”

□□□□□□□□□□□□□□□□□□□□□□□□□□□□

1.

□□□□□□□□□□□□□□

2.

□□□

s1

□□□□□□□□□□-X□□□□□□□□

s2

□□□

s1

□

s2

□□□□□□

3.

□□

s1

□

s2

□□□

I

□□□□□□-ʒ□□□

s1->s2

□□□□□□□□□□

□□

II

□□□□□□□□

s1->s2

□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□-À

BFS

□□

BFS

□□□□□□□□□□ƁB□□

:

1.

□□□□□□□□□□□□□□□□□□

□□□□□□□□□□

(1)

□□□□□□□□□□□□□□Ú□□□□□□□□□□□□□□□□□□□□□□□□□

n*w

□

n

□□□□□□□□□□□

w

□□□□-□□□

(2)

□□□□□□□□□□□

x

□□□□□□□

a z

□□

x

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

26*w

□□

w

□□□□□□□

□□□□□□□□-□Θ□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

100

□□□□□□□□□□□-

2

□□□□□

2.

□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□

3.

□□

BFS

□□□□□□□□□□

backtracking

□□□□□

□□□□□□□

:

/**

* Solution: Graph BFS

*

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

BFS

□□□□□□□□□□□

Graph

□□□□□□□

*

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

*

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

25 * L.

□□□□□□□

Graph,

□□□□

start - end

□□□□□□□

.

*

□□□□□□□

start

□□□□

adjacent string,

□□

enqueue,

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

end

□□□□

shortest = length + 1.

*

*

□□□□□□

:

* 1. dict

□□□

BFS

□

visited

□□□□□

remove from dict

□□□□□□□

□□

□□□

* 2.

□□□□□

,

□□□□□□□

.

□□□□□

(

□□□□□□□□□□□□□□

char

□□□□□□□□□□□□□□□□□□□□□□□□□

)

*

* Time Complexity.

□□□□□□

*

□□□□

:

□□□□□□

,

□□□□□

26*wordLength.

□□□□□□□□□

dict

□□□□□

. O(dict.size * 26*wordLength)

* Space

□□□□□

Queue

□□□□□□□□□□

dict

□

size,

□□

dict

□□□□□□□□□□□□

O(1)

*/

□□□□□□□□□□□□-ä□□□□□

Share C # code on github:

https://github.com/jianminchen/word-ladder-I/blob/master/Program.cs

1.1.59 Leetcode question 66: plus one (2015-06-22 18:45)

June 22, 2015

Given a number represented as an array of digits, plus one to the number.

Leetcode: plus one, □□□□□, □□□□□□□□; □□□□□□□ (

6□□□

).
Read the web blogs, and then, try different solutions (Six implementations

).

practice using C #, the source code on github:

https://github.com/jianminchen/math-plus-one/

https://github.com/jianminchen/Leetcode _C-/blob/master/66PlusOne.cs

Try different solutions through blogs, and then, catch up something interesting; basic programming styles, for loop, while loop, and different ways to check carry, using %, /, ==10, ==9; one problem can be interpreted with different solutions. Fun time to play with source code, and get familiar with basic C # stuff, array, initialization.

1.1.60 Leetcode: Count Primes (2015-06-22 18:53)

On June 22, 2015

Problem statement:

Count the number of prime numbers less than a non-negative number, n

Hint: The number n could be in the order of 100,000 to 5,000,000.

Read the article to talk about how to improve skills as a programmer, one comment I like most:

http://blog.csdn.net/gsky1986/article/details/46556933

"□□□□□□□□□□□□□□□□□□□□□□-ğ¥□□□□□□□□□□□□□□□□□□□□", □□□□□□□□□, □□□Leetcode, □□□C # □□; □□□□□□□□C #□□□□.

Share the C # code in github:

https://github.com/jianminchen/count-primes/blob/master/Program.cs

1.1.61 Cache design, dynamic programming - matrix region sum (2015-06-25 21:37)

June 25, 2015

Problem statement:

Given the integer matrix, top left and bottom right coordinates of the rectangular region, calculate the region sum.

Read the blog:

http://www.ardendertat.com/2011/09/20/programming-interview-questions-2-matr ix-region-sum/

Naive solution, O(n x m) calculation of addition to get the sum. How to get it as O(1) using cache, how big the cache space?

Naive way to design the cache, it takes O(n^2 x m^2) space. The efficient way is O(n x m), using dynamic programming.

Share the practice C # code:

https://github.com/jianminchen/matrixRegionSum/blob/master/Program.cs

1.1.62 Leetcode: Jump Game (2015-06-29 22:26)

June 29, 2015

Problem statement:

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Your goal is to reach the last index in the minimum number of jumps.

For example:
Given array A = [2,3,1,1,4]

The minimum number of jumps to reach the last index is 2. (Jump 1 step from index 0 to 1, then 3 steps to the last index.)

Spent some time to go over the blogs, and found this one with good illustration. So, write some c # code and then get first hand experience. □□□□□□□□, □□□□□□□. □□C #□□. □C++, Java□□□□□C #. □□□□□□□□□.

http://www.cnblogs.com/lichen782/p/leetcode _Jump _Game _II.html

Go through the code practice using C #:
https://github.com/jianminchen/jumpGame/blob/master/Program.cs

1.2 July

1.2.1 Leetcode: Add Binary (2015-07-03 22:41)

July 3, 2015

Given two binary strings, return their sum (also a binary string).

For example,
a = "11"
b = "1"
Return "100".

work on programming using C #; find great code to convert it from C++ to C #, know the difference.

https://github.com/jianminchen/addBinary/blob/master/Program.cs

1.2.2 ITint5 - questions (2015-07-03 22:44)

July 3, 2015

Went through the first 10 questions from the blogs:

Stumble on most questions:

https://github.com/AnnieKim/ITint5/blob/master/004 _ %E7 %BB %9F %E8 %AE %A1 %E5 %AE %8C %E5 %85 %A8 %E4 %BA %8C %E5 %8F %89 %E6 %A0 %91 %E7 %BB %93 %E7 %82 %B9 %E6 %95 %B0.cpp

Need to practice this question using C #, excellent question. Do not think clearly about height calculation, left side height vs right side height, which is the complete the tree.

Another one,

https://github.com/AnnieKim/ITint5/blob/master/008 _ %E6 %9C %80 %E5 %A4 %A7 %E8 %BF %9E %E7 %BB %AD %E5 %AD %90 %E6 %AE %B5 %E5 %92 %8C.cpp

how to come out the best solution? Naive solution is O(N^2), start point and end point, each has n choices; so, let us try O(n) solution. Go through one loop, maybe multiple times. How many variables, …stumble on easy question.

And then, how to extend the problem: if the start point can connect end point, like a ring. Stumble on this problem, so, take easy step, first, ask myself, what is the case no ring? And then, what is connection between two of them. Clever solution.

https://github.com/AnnieKim/ITint5/blob/master/009 _ %E7 %8E %AF %E5 %BD %A2 %E6 %9C %80 %E5 %A4 %A7 %E8 %BF %9E %E7 %BB %AD %E5 %AD %90 %E6 %AE %B5 %E5 %92 %8C _ %E8 %A7 %A3 %E6 %B3 %952.cpp

Topological order learning:

First time read it about this ordering - topological order

https://github.com/AnnieKim/ITint5/blob/master/010 _ %E4 %BB %BB %E5 %8A %A1 %E8 %B0 %83 %E5 %BA %A6.cpp

http://www.cnblogs.com/skywang12345/p/3711494.html

One comment is that she needs to go through quick learning through problems, 10 problems a time; pick up something, like a hint, and then, know how to expand her knowledge.

1.2.3 Leetcode 124: Maximum binary tree path sum (2015-07-04 14:25)

July 4, 2015

problem statement:

Given a binary tree, find the maximum path sum.

The path may start and end at any node in the tree.

For example:

Given the below binary tree,

1

/ \

2 3

Return 6.

Blogs to read:

http://blog.unieagle.net/2012/12/09/leetcode %E9 %A2 %98 %E7 %9B %AE %EF %BC %9Abinary-tree-maximum-path-sum/

http://shanjiaxin.blogspot.ca/2014/02/binary-tree-maximum-path-sum-leetcode .html

http://bangbingsyb.blogspot.ca/2014/11/leetcode-binary-tree-maximum-path-su m.html

http://jane4532.blogspot.ca/2013/09/binary-tree-maximum-path-sumleetcode.ht ml

□□

□□□□-À□□

:

http://www.cnblogs.com/yuzhangcmu/p/4172855.html

The best readable code, and perfect to follow for this problem:

https://github.com/xiaoxq/leetcode-cpp/blob/master/src/BinaryTreeMaximumP athSum.cpp

Julia's C # practice.

https://github.com/jianminchen/BinaryTreeMaxPathSum/blob/master/Program. cs

https://github.com/jianminchen/Leetcode _C-/blob/master/124BinaryTreeMaximumPathSum.cs

Extension problem:

https://github.com/AnnieKim/ITint5/blob/master/013 _ %E6 %A0 %91 %E4 %B8 %AD %E6 %9C %80 %E5 %A4 %A7 %E8 %B7 %AF %E5 %BE %84 %E5 %92 %8C.cpp

February 3, 2016

Review the algorithm, need to figure out how to make this algorithm easy to remember, recall, write without a bug.

Good analysis from the blog (http://www.cnblogs.com/yuzhangcmu/p/4172855.html):

□□□□□□

path

□

2

□□Å□□

1.

□

□□□

path.

(1)

□□□□

□□□□□□□□□□□□

Node

□

path

□□□□□□

□□

root

□

(2)

□□□□

□□□□□□□□□□□□

Node

□

path

□□□□□□

□□

root

□

2.

□□

□□□

path.

□□□□□□□□□□□□□

path

□□□

□

□

□□

□□□□

inner class:

□□

2

□

□

□

1.

□

□□□□

path

□

2.

□

□□□□□□□□□□□□□ş□□□□

path.

□□□□

root == null,

□□

2

□

□□□□□

Integer _MIN _VALUE;

□

□□□□□□□□□□□□□□□□□

solution

□□□□□

60

□□□□□

□

http://www.cnblogs.com/yuzhangcmu/p/4172855.html

Practice using this class as return type:

```
private class ResultType {

int singlePath;

int maxPath;

ResultType(int singlePath, int maxPath) {

this.singlePath = singlePath;

this.maxPath = maxPath;

}
```

Read more blogs:

http://buttercola.blogspot.ca/2014/08/leetcode-binary-tree-maximum-path-sum.html

```
Discussion of global variable - max value - Julia likes the discussion


https://leetcode.com/discuss/14190/accepted-short-solution-in-java
```

1.2.4 Leetcode: good blogs to follow (2015-07-05 11:40)

July 5, 2015,

Went through 6 months training on C # using leetcode, learned a lot. So, next two blogs to read:

https://github.com/yinlinglin/LeetCode

https://github.com/xiaoxq/cracking-the-coding-interview-v4-cpp

Read the blog, and transfer C++ to C # code. So, she can learn C #, and also know the difference between C++ and C #.

https://github.com/xiaoxq/leetcode-cpp

https://github.com/Sayericplz/myleetcode

Another good blog to read:

https://github.com/zwxxx/LeetCode

Code in java:
https://github.com/patrickyao1988/LeetCode-Java

https://github.com/wuhanyu/leetC2013

1.2.5 Leetcode: word search (2015-07-06 21:23)

July 6, 2015

Problem statement:

Word Search

Given a 2D board and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once.

For example,
Given **board** =

```
[
  ["ABCE"],
  ["SFCS"],
  ["ADEE"]
]

The problem can be solved using backtracking, recursive solution.
```

Share C # implementation:

https://github.com/jianminchen/wordSearch-/blob/master/Program.cs

1.2.6 ITINT5: tree maximum path sum (I) (2015-07-06 21:25)

July 6, 2015

Problem statement:

□□

: http://www.itint5.com/oj/ #13

□□

:

□□□□□□□□□□□□□□□□□□□□□□□□

val

□

□□□□□□□□

2

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□-□

□□□□□□□□□□□□□□□□□□□

□□□

-10

/ | \

2 3 4

/ \

5 -1

/

6

/

-1

□□□□□□□

13

□□□□□□□□

5

□

6

□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□

“

□□□□

“-□

□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□

val

□□

1

□□□□□□□□□□□□□□□□□□□□□

Solution: LeetCode

□

Binary Tree Maximum Path Sum

□□□□□□□□□□□□□ω□□

Share C # code:

https://github.com/jianminchen/TreeMaxPathSum/blob/master/Program.cs

February 3, 2016

work on Leetcode question 124: Binary Tree Maximum Path Sum

1.2.7 ITInt5: maximum boxes (2015-07-06 21:26)

July 6, 2015

Problem statement:

Link: http://www.itint5.com/oj/ #34

□□

64

:

□

n

□□□□□□□□□□□

vol

□□□

weight

□□□□□□-Ƶ□□□

(vol, weight)

□□□

□□□□□□□□□□□□□□□□□□□□□□-ч□□□□□□□□□□□□□□□□□□□□□□□□□

□□□

□

7

□□□

boxes:

[(65, 100), (70, 150), (56, 90), (75, 190), (60, 95), (68, 110), (80, 12)]

□□□□□

6

□□□□□□□-Φ□□□□

(56, 90), (60, 95), (65, 100), (68, 110), (70, 150), (75, 190)

□□□□□□□□

6

□

□□□-₸□□

CRACKING THE CODING INTERVIEW 9.7

Solution:

□□□

vol

□□□□□□□□□□Ř□□□

weight

□□□□□□□□□□□

Share C # practice:

https://github.com/jianminchen/MaxBoxes/blob/master/Program.cs

1.2.8 Leetcode Question No. 53: Maximum subarray sum (2015-07-06 21:32)

July 6, 2015

problem statement:

Find the contiguous subarray within an array (containing at least one number) which has the largest sum.

For example, given the array [−2,1,−3,4,−1,2,1,−5,4],
the contiguous subarray [4,−1,2,1] has the largest sum = 6.

Study the code:

http://joycelearning.blogspot.ca/2013/10/leetcode-maximum-subarray.html

C # code:

https://github.com/jianminchen/MaximumSubArray/blob/master/Program.cs

https://github.com/jianminchen/MaximumSubArray _Space/blob/master/Program.cs

More reading:

1. Keyword lookup:

DP problem: Kadane's algorithm, maximum subarray sum

2. Write a program to find the sum of contiguous subarray within a one-dimensional array of numbers which has the largest sum.

http://www.geeksforgeeks.org/largest-sum-contiguous-subarray/

January 3, 2015

Review the question.

Here is the link of C # solution:

https://github.com/jianminchen/Leetcode _C-/blob/master/53MaximumSubArray1.cs

https://github.com/jianminchen/Leetcode _C-/blob/master/53MaximumSubArray2.cs

study code:

http://joycelearning.blogspot.ca/2013/10/leetcode-maximum-subarray.html

https://github.com/xiaoxq/leetcode-cpp/blob/master/src/MaximumSubarray.cpp

1.2.9 ITInt5: Arithmetic Expression Evaluation (2015-07-09 22:29)

July 9, 2015

Read blogs:

```
7+3*4*5+2+4-3-1
```

https://github.com/AnnieKim/ITint5/blob/master/026 _ %E8 %A1 %A8 %E8 %BE %BE %E5 %BC %8F %E6 %B1 %82 %E5 %80 %BC.cpp

https://github.com/jordandong/myITint5/blob/master/26 _Evaluation.cpp

1.2.10 ITInt5: Excel columns to integer (2015-07-09 22:36)

July 8, 2015

Problem statement:

□□: http://www.itint5.com/oj/ #23
□□:
Excel□□□□□□A Z 26□□□□□□A, B, C, D, ..., Z, AA, AB, ..., AZ, BA, BB, ...
□□□□10□□□1, 2, 3, 4, ..., 26, 27, 28, ..., 52, 53, 54...□
□□□2□□□decToExcel□excelToDec□□10□□□□□□Excel□□□□□Excel□□□□10□□□□□

Share C # code for the practice:

https://github.com/jianminchen/ExcelColumnStrToInt/blob/master/Program.c s

1.2.11 Leetcode: study time (2015-07-13 20:08)

July 13, 2015

Choose to read blogs with code using Java programming language. □□□□□□□□□□.

http://yyeclipse.blogspot.ca/2012/11/leetcode-populating-next-right-pointer s.html

http://yyeclipse.blogspot.ca/2012/11/leetcode-set-matrix-zeroes.html

http://yyeclipse.blogspot.ca/2012/11/solving-maximal-rectangle-probl em-based.html

http://yyeclipse.blogspot.ca/2013/01/leetcode-generate-parentheses.ht ml

The majority vote (more than n/3 element in an array, go through the array once or twice, get the number)

http://www.cs.utexas.edu/ moore/best-ideas/mjrty/index.html

The maximum rectangle problem:

http://www.drdobbs.com/database/the-maximal-rectangle-problem/184410529?p gno=1

The longest increasing subsequence:

http://www.felix021.com/blog/read.php?1587

http://yyeclipse.blogspot.ca/2013/02/notes-on-lcs-lis-and-related-pr oblems.html

Microsoft interview question:

http://yyeclipse.blogspot.ca/2013/02/two-equal-sized-subsets-partition.h tml

Review strstr leetcode question: (KMP, Boyer-Moore, Rabin-Karp algorithm)

http://yyeclipse.blogspot.ca/2013/02/leetcode-implement-strstr-string.h tml

DP problem: leetcode question

http://yyeclipse.blogspot.ca/2013/03/leetcode-palindrome-partition-ii.htm l

http://www.geeksforgeeks.org/dynamic-programming-set-17-palindrome-partitioni ng/

http://www.geeksforgeeks.org/dynamic-programming-set-8-matrix-chain-multipl ication/

1.2.12 Leetcode: study time (2015-07-13 22:38)

July 14, 2015

□Leetcode□□□□□□□□□□-ɧ□□□□□; □□, □□□C #□□□, □C++□□□, □□C #; □Java □□□□□C #. □□, □□□□□□□□□□, □□□□□, □□□□. □□, □□□□□□□□□□, □□□□□□□□□□□□, □□□□□□□□□□.

□□□□□□□□□:

http://blog.csdn.net/fightforyourdream/article/category/1350225

http://www.lifeincode.net/category/programming/

1.2.13 leetcode Question No 132: palindrome 2 (2015-07-16 21:51)

July 16, 2015

Problem statement:

Given a string *s*, partition *s* such that every substring of the partition is a palindrome.

Return the minimum cuts needed for a palindrome partitioning of *s*.

For example, given *s* = "aab",
Return 1 since the palindrome partitioning ["aa","b"] could be produced using 1 cut.

Read the blogs:

http://blog.csdn.net/fightforyourdream/article/details/17730939

Share C # code:

https://github.com/jianminchen/palindromeII/blob/master/Program.cs

January 2, 2016
Review the source code, and try to figure out the algorithm.

1.2.14 leetcode: longest substring without repeating characters (2015-07-16 21:58)

July 16, 2015

Problem statement:

Longest Substring Without Repeating Characters

Read the blog:

http://blog.csdn.net/fightforyourdream/article/details/17860983

□□□□□□□□□□, □□□□, □□□, □□□□□□□□□□, □□□□, □□□□, □□□□□□□; □□, □□□□□□, □□□□□, □□□□□□□□□□, □□□□□□, □□. □□□, □□□□, □□□□□□□, □□□□□□□; □□□□□□□□□□□□□□, □□□□, □□□□□□□□□□.

□□□□□□□, □□□□□□□, □Java□□□, □□C #; □□, □□□□□□□□□□, □□□□□□□□□□□□□□□, □□, □□□; □□□□□□□□□□□□□□□□□□. □□□□□□□□, □□□□□□C #□□□□.

□□□□□□□□□□, □□□□□□□□□□□, □□□□□□□□□□□□□□□□. □□□□□□□□□□, □□, □□□□□□□□□□□□□, □□, □□□□□□□□□□□□□□□; □□□□□□(256□□)□□□□□□-,□□□□□□□□□□□□□, □□□□, □□□□□□, □□□□□□□□□□□□□□□; □□□□□, □□□□□□□□□□□□□□□□□□□□□□□□.

□□□□□□□□, □□□□□□□□□□□, □□□□□□□□, □□□□□□□□□□□□□□.

Julia□□□□□□□□, □□□□□□□□C #, □□□□□□□□□□□□□□□□, □□C #□□□; □□, □□□□□□□□□□□□□, □□□□□; □□□□□, □□□□□, □□□□□□□□□, □□□□□□□□□, □□□□□□□□, □□□□□, □□□Leetcode□□, □□□□□.

Share C # code:

https://github.com/jianminchen/BinaryTreeMaxPathSum/blob/master/longestSubst ringWithoutRepeatingChars.cs

First blog about the same problem:

http://juliachencoding.blogspot.ca/2015/06/longest-substring-without-repe ating.html

"□□□□□□,□□□□□□-ļ ", □□□□□□□□□, □□, □□□□□, □□□□□□□□□□□, Hashset, □□□□□□256□□□□□□□□□, □□□□□10 - 15□□□□□□□□. □□Julia□□□□□□, □□□□□□□□□□□□□□□.

https://github.com/jianminchen/Leetcode _C-/blob/master/3LongestSubstringWithoutRepeating.cs

1.2.15 Leetcode: maximal rectangle (2015-07-19 12:23)

July 19, 2015

Given a 2D binary matrix filled with 0's and 1's, find the largest rectangle containing all ones and return its area.

http://www.aiuxian.com/article/p-389634.html

http://www.cnblogs.com/lichen782/p/leetcode _maximal _rectangle.html

Fully understand the problem and solution, and then, spend some time to convert java code to C # code; and then, play with code, and see if I can improve the code.

1.2.16 Leetcode: Sudoku Solver (2015-07-19 23:13)

July 19, 2015

Problem statement:

Write a program to solve a Sudoku puzzle by filling the empty cells.

Empty cells are indicated by the character '.'.

You may assume that there will be only one unique solution.

Solution 1:

Great blog to read:

http://blog.csdn.net/fightforyourdream/article/details/16916985

And then, start to implement the solution using C # code:

https://github.com/jianminchen/sudokuSolver/blob/master/Program.cs

Solution 2:

Read the blog,
http://blog.csdn.net/linhuanmars/article/details/20748761

and then, implement the solution using c # code:

https://github.com/jianminchen/sudokuSolver/blob/master/Program2.cs

Solution 3: (good workout on C # KeyValuePair class)

and then, convert C++ code to C # code from the blog:
https://github.com/yinlinglin/LeetCode/blob/master/SudokuSolver.h

Excellent code in C++, using class for node on the board. Learn a few things, fun to play with the code

C # code:

https://github.com/jianminchen/sudokuSolver/blob/master/Program3.cs

solution 4:
https://github.com/jianminchen/sudokuSolver/blob/master/Program4.cs
source code from the blog:
https://github.com/xiaoxq/leetcode-cpp/blob/master/src/SudokuSolver.cpp

Solution 5:
read the blog: (Good coding! practice more based on this blog)
https://github.com/zwxxx/LeetCode/blob/master/Sudoku _Solver.cpp

and convert the C++ code to C # code, (great workout on C # LinkedList for blank nodes)

https://github.com/jianminchen/sudokuSolver/blob/master/Program5.cs

Solution 6:
read the blog:
http://shanjiaxin.blogspot.ca/2014/04/sudoku-solver-leetcode.html

and convert Java code to C # code, great workout on C # and logic checking " return false "

https://github.com/jianminchen/sudokuSolver/blob/master/Program6.cs

Solution 7:
blog:
http://www.jiuzhang.com/solutions/sudoku-solver/
C # code:
https://github.com/jianminchen/sudokuSolver/blob/master/Program7.cs

Solution 8:
Thanks for the blog's highlight line of code on back tracking; finally, I got it! My logic thinking has flaws on back tracking; extra backtracking is not a good. Minimize the back tracking , only do it when "return false". It makes sense to do that.

blog:
http://bangbingsyb.blogspot.ca/2014/11/leetcode-valid-sudoku-sudoku-sol ver.html
C # code:
https://github.com/jianminchen/sudokuSolver/blob/master/Program8.cs

Solution 9:
blog:
http://www.cnblogs.com/TenosDoIt/p/3800485.html

C # code:
https://github.com/jianminchen/sudokuSolver/blob/master/Program9.cs

Solution 10:
blog: (Excellent implementation! no extra line or number in the code! Best for memorization! Go through other solutions later.)
https://github.com/rfffffffff007/leetcode/blob/master/Sudoku %20Solver.java

C # code:
https://github.com/jianminchen/sudokuSolver/blob/master/Program10.cs

□□□-ƨ□□, □□□□□□□□; □□□□□□□□□, □□□□□□□□□; □□□□□□□□□, □□□□! □□□□, □□□□□□□□□□, □□□□; □□□□□□□□□□. □□□□□□□□□□□□□□, □□□□□□□; □□□□□□□□□□□□□□□, □□□□□□□□□, □□□□□□; □□□□C #□□□□□□.

Also, the code written has been work on readability, learned through my favorite book reading:
http://shop.oreilly.com/product/9780596802301.do

Those favorite rules I like to learn, pick up and follow:

Big fan of DRY (Do not repeat yourself) principle, do one thing a time, break giant expression, using explaining variable or summary variable, and abstract the thing to a function, extract a subproblem to a function. The code is also modified to fit into short memory, less mental baggage to read through.

Read solutions:

https://github.com/jordandong/myleetcodes/blob/master/SudokuSolver.cpp

https://github.com/Sayericplz/myleetcode/blob/master/isValidSudoku.cpp

BFS, using queue - try to convert it to C #
http://yucoding.blogspot.ca/2013/12/leetcode-question-sudoku-solver.html

1.2.17 Leetcode 102: Binary tree level order traversal (2015-07-23 22:01)

July 23, 2015

Problem statement:

Given a binary tree, return the *level order* traversal of its nodes' values. (ie, from left to right, level by level).

For example:
Given binary tree {3,9,20, #, #,15,7 },

```
    3
   / \
  9  20
    /  \
   15   7
```

return its level order traversal as:

```
[
  [3],
  [9,20],
  [15,7]
]
```

confused what " {1, #,2,3 }" means? > read more on how binary tree is serialized on OJ.

□□□□□□□□□□. □□□□□, □□. □□□□□□□□□□□□□. □C++□□, □□□C #, □□□□□. □□□□□□□□C #□□, □□□□! □□6□□□.

1. Solution 1: (push extra null node in the queue to divide level)

Read the blog:

https://github.com/xiaoxq/leetcode-cpp/blob/master/src/BinaryTreeLevelOrderT raversal.cpp

convert it to C # code:

https://github.com/jianminchen/BTreeLevelOrderTraversal/blob/master/BTreeLev elOrderTraversal.cs

C # code passing leetcode online judge:

https://github.com/jianminchen/BTreeLevelOrderTraversal/blob/master/BTreeLe velOrderTraversal2.cs

Solution 2: (using 3 variables to help queue to do BFS algorithm)

blog:

http://fisherlei.blogspot.ca/2013/01/leetcode-binary-tree-level-or der.html

C # code:

https://github.com/jianminchen/BTreeLevelOrderTraversal/blob/master/BTreeLev elOrderTraversal3.cs

Solution 3: DFS algorithm:

blog:

http://fisherlei.blogspot.ca/2013/01/leetcode-binary-tree-level-or der.html

C # code:

https://github.com/jianminchen/BTreeLevelOrderTraversal/blob/master/BTreeLev elOrderTraversal4.cs

Solution 4: using two containers for current level, next level

blog:

http://bangbingsyb.blogspot.ca/2014/11/leetcode-binary-tree-level-orde r.html

C # code:

https://github.com/jianminchen/BTreeLevelOrderTraversal/blob/master/BTreeLev elOrderTraversal5.cs

Solution 5: one queue and iteratively solution (**□□queue□□□□□)**

blog:

http://bangbingsyb.blogspot.ca/2014/11/leetcode-binary-tree-level-order.htm l

C # code:

https://github.com/jianminchen/BTreeLevelOrderTraversal/blob/master/BTreeLev elOrderTraversal6.cs

Solution 6: one queue and

extra node null

into queue to mark end of level

blog:

http://bangbingsyb.blogspot.ca/2014/11/leetcode-binary-tree-level-order .html

C # code:

https://github.com/jianminchen/BTreeLevelOrderTraversal/blob/master/BTreeLev elOrderTraversal7.cs

Blogs to read:

1. http://www.jiuzhang.com/solutions/binary-tree-level-order-traversal-ii/
2. http://siddontang.gitbooks.io/leetcode-solution/content/tree/binary _tree _level _order _traversal.html
3. http://blog.csdn.net/ljphhj/article/details/22428939

1.2.18 Leetcode 238: Product of Array except itself (2015-07-25 02:05)

July 24, 2015

Read the blog and see if the blog should be on my reading list in next week:

http://noalgo.info/tag/leetcode/page/3

and then, found a link on the blog, and enjoyed reading the blog to the end.

http://www.guokr.com/article/61878/

?Leetcode????, ??????????, ???, ????; ???????????. ??, ?Leetcode????, ??????????, ??, ?????.

??????????.

????????“??”???????????????google????

??-)??????????? N ????N ????????? N ????????????? N ????????? k ????????????????? ??????????????????-??????????????????-?

??????????? A [1 .. n] ????? O (n) ????????????????? B [1 .. n] ???? B [i] = A [1] * A [2] * … * A [n]?A [i] ???????????????

???S?????????????????????????????????????-?†???????

????????6?????????????????????????????????????-ũS?????????????????????? k ????????????????? Ń?????????????????????

??????????? A [1 .. n] ?????????????????æ????????? B [1 .. n] ???? B [i] = A [1] * A [2] * … * A [n]?A [i] ? n???????????? O (n) ??????????????????Б?????????n??????-ĦÅ?????????????????-ɟ?????? n ????????C?????????????????? O (n) ??????????????????;????????? n 2 ?????C??????????-?æ?????? O (n 2) ??

??????????????????N???????????? k?N?????????????-ɤ?????????????????????????????? ??????????????????-Ȧz?????????????????Ў???-Ч?????? 0 ? 1 ??? k ?????????

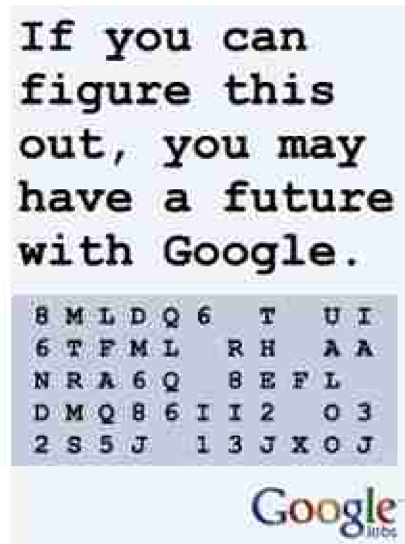
???????????????????? B [i] ????????????? B [i] = A [1] * … * A [i - 1] * A [i + 1] * … * A [n] ??????????????????-?????? B[i] ????????n - 1????????????????? B [1 .. n] ???? O (n 2) ?????????????????Ă???????????

□□□ B [i] □□□□□□□□□□□□□□ A [1] * … * A [i - 1] □ A [i + 1] * … * A [n] □□□ B [i + 1] □□ A [1] * … * A [i - 1] * A [i] □ A [i + 2] * … * A [n] □□□□□ B [i] □□□□□□□□□□ B [i + 1] □□

S [i] = A [1] * … * A [i – 1]
T [i] = A [i + 1] * … * A [n]

□□□□□□□ S [1 .. n] □ T [1 .. n] □□□ O (n) □□□□□□□□□□□□ B [i] = S [i] * T [i] □□□□□□□□□□ B [1 .. n] □□□□□ O (n) □□□□□□□

> □□□□□□3□□□□□□□□□□□□□□MIT□□□□□□□□□□□□□□-□□□□□□□□□□□□□Google Jobs□□□□□□□□□□□□□ - □□□□□□□□□□□□□□□□-□□-□□□



1.2.19 Algorithm reading time (2015-07-26 23:15)

July 26, 2015

So lucky to come cross the webpages and then start to read, great reading time:

http://www3.cs.stonybrook.edu/ rezaul/Spring-2015/CSE548/Yonghui-Wu/lecture- 2-1.pdf

Competitive Programming for Solving Algorithmic Problems:
http://www3.cs.stonybrook.edu/ rezaul/CSE548-S15.html

1.2.20 Leetcode Question No 70: climbing stairs (2015-07-28 19:46)

July 28, 2015

Problem statement:

You are climbing a stair case. It takes *n* steps to reach to the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

The problem is most popular question in the algorithm, so I do like to spend time to find out all sorts of solution, and get myself comfortable to all kinds of ideas, and figure out which one is best, and all concerns we can have in the discussion of climbing stairs:

1. Recursion solution vs. DP problem solution (Dynamic Programming solution)
2. Time complexity solution: O(2^n) vs O(n) solution
3. The space O(N) vs O(1), in other words: array of N or 2 variable, and another tmp variable
4. The base case discussion: f(0) = 1 or f(0) =1, math question?
5. Math formula - closed form solution vs DP problem solution
6. Use Memoization DP vs. no memoization DP
7. Programming skills, how to make code easy to follow, more readable, more abstract.

The investment of time on the problem is well done. Go over 16 implementation one by one using C # programming language.

C # code:

https://github.com/jianminchen/climbingStairs/blob/master/climbingStairs.cs

□Ŷ□, □□□□□□□□□, □□□□□□□□□□□; □□□□□□□□□□□□□, □□□□□; □□□□□□□□□□□□□, □□□□□□; □□□□□□, □□, □□□□□, □□□□□□□□□□□□, □□□□□□□□□□□, □□, □□□□□□□□□, □□□□□□□□□□. □□, □□□□□□.

□□□□□□□, C #□□□□□□; □□□□□□□□□□□; □□□□□Leetcode□□□C #□□, □□□□□□□□□□□□□□□□□-□□, □□□□□□□□□□□.

January 3, 2016

Review the leetcode question 70, climbing stairs.

Read the blog:

http://blog.csdn.net/kenden23/article/details/17377869
http://yucoding.blogspot.ca/2012/12/leetcode-question-15-climbing-stairs.ht ml

http://www.cnblogs.com/springfor/p/3886576.html

http://www.cnblogs.com/springfor/p/3886576.html

http://siddontang.gitbooks.io/leetcode-solution/content/dynamic _programming/climbing _stairs.html

https://github.com/zwxxx/LeetCode/blob/master/Climbing _Stairs.cpp

1.2.21 SQL Server DBA, database design, and development (2015-07-28 19:50)

July 28, 2015

SQL server DBA most popular guidelines for self-learner:

1.

Database may be set up simple transaction log, so the database can be fixed size; never worry about disk space;

2.

Back up database every day; no pressure to work on live database update; certainly, reduce risk to manual update record one by one instead of update statement.

Also, the website to find the SQL performance script to do daily monitor using database mail setup, so database disk space, backup log, auto growth log, and all other activity can be helpful for every morning.

Here is her favorite site to get the SQL script:

https://ola.hallengren.com/

Never too later to share her thoughts and great ideas with outside world, also her first blog about SQL server DBA experience.

1.2.22 CSS, JavaScript, Jquery learning (2015-07-28 19:53)

July 27, 2015

Basic training code for CSS, html, Jquery, Javascript from books, public internet, and other people's blog. Training basics, know concepts.

Learning a new language takes time, know syntax, memorize CSS selectors, and then, read "Head first CSS" and understand basic box model, rules, cascading rules, specificity in CSS etc.; she should have stayed more organized.

She tries to learn more about CSS, html, Java Script, practice more code.

https://github.com/jianminchen/JavaScriptAndPHP

https://github.com/jianminchen/JQueryExamples

https://github.com/jianminchen/CSSExamples

https://github.com/jianminchen/JavaScriptCNBlogs _DolphinX

https://github.com/jianminchen/JQueryExamples

https://github.com/jianminchen/JavaScriptPattern

My favorite thing to do is to challenge myself, try to memorize something over 10 times and see if it will help me learn a new language, improve programming performance.

Java Script cheat sheet from book Pocket Guide To JavaScript is my favorite one to read, and practice to memorize.

Here is another one:

http://www.cheatography.com/davechild/cheat-sheets/javascript/

1.2.23 Leetcode: lowest common ancestor in binary tree (2015-07-30 22:30)

July 30, 2015

problem statement:

lowest common ancestor in binary tree, question No: 236

Try 4 implementations, 2 passes, but 2 failed.

read blogs:

http://articles.leetcode.com/2011/07/lowest-common-ancestor-of-a-binary-tr ee-part-i.html

http://huangyawu.com/leetcode-lowest-common-ancestor-of-a-binary-tre e/

http://www.cnblogs.com/chkkch/archive/2012/11/26/2788795.html

write C # code:

https://github.com/jianminchen/LowestCommonAncestorInBinaryTree/blob/master /LowestCommonAncestorB.cs

1.3 August

1.3.1 Leetcode study time (2015-08-02 15:12)

August 2, 2015

□http://zzk.cnblogs.com/□□"□□□",□□□□□□□,□□□□-□□□□Leetcode□□□

Read the leetcode question and solutions:

1. Find lowest common ancestor in binary tree:

blog of a facebook programmer working on leetcode questions:

□□□□□□□□O(1)□□-□O(n)

http://www.cnblogs.com/chkkch/archive/2012/11/26/2788795.html

Try it myself and compare to solutions in my blog:
http://juliachencoding.blogspot.ca/2015/07/leetcode-lowest-common-ancestor-i n.html

2. Reverse nodes in k group

http://www.cnblogs.com/lichen782/p/leetcode _Reverse _Nodes _in _kGroup.html

http://www.cnblogs.com/grandyang/p/4606710.html

3. binary tree

http://www.cnblogs.com/wwwjieo0/p/3891037.html

4. All 158 leetcode algorithm implementation in git hub:

http://coolshell.cn/articles/1870.html

Leetcode solution on git hub (use this one more often, best answer, author profile: http://coolshell.cn/haoel)

https://github.com/haoel/leetcode/tree/master/algorithms

5. Leetcode using C #

https://github.com/xisuogu/LeetSharp/tree/master/LeetSharp

6. One more to read

http://blog.csdn.net/u014674776/article/category/2296455

□□□□□□□□,□□□□□□□□□□□□□□□Γ□□□□□□□□□!

1.3.2 Leetcode questions and web link (2015-08-05 20:21)

August 5, 2015

Here is the table about leetcode questions:

1

Two Sum

2

Add Two Numbers

3

Longest Substring Without Repeating Characters

4

Median of Two Sorted Arrays

5

Longest Palindromic Substring

6

ZigZag Conversion

7

Reverse Integer

80

8

String to Integer (atoi)

9

Palindrome Number

10

Regular Expression Matching

11

Container With Most Water

12

Integer to Roman

13

19

Remove Nth Node From End of List

20

Valid Parentheses

21

Merge Two Sorted Lists

22

Generate Parentheses

23

Merge k Sorted Lists

24

30

Substring with Concatenation of All Words

31

Next Permutation

32

Longest Valid Parentheses

33

Search in Rotated Sorted Array

34

Search for a Range

35

Search Insert Position

36

Valid Sudoku

37

Sudoku Solver

38

Count and Say

39

Combination Sum

40

Combination Sum II

41

First Missing Positive

42

Trapping Rain Water

43

Multiply Strings

44

Wildcard Matching

45

Jump Game II

46

52

N-Queens II

53

Maximum Subarray

54

Spiral Matrix

55

Jump Game

56

Merge Intervals

57

Insert Interval

58

Length of Last Word

59

Spiral Matrix II

60

Permutation Sequence

61

Rotate List

62

Unique Paths

90

63

Unique Paths II

64

Minimum Path Sum

65

Valid Number

66

Plus One

67

Add Binary

68

74

Search a 2D Matrix

75

Sort Colors

76

Minimum Window Substring

77

Combinations

78

Subsets

79

Word Search

80

Remove Duplicates from Sorted Array II

81

Search in Rotated Sorted Array II

82

Remove Duplicates from Sorted List II

83

Remove Duplicates from Sorted List

84

Largest Rectangle in Histogram

85

Maximal Rectangle

86

Partition List

87

Scramble String

88

Merge Sorted Array

89

Gray Code

90

Subsets II

91

Decode Ways

92

Reverse Linked List II

93

Restore IP Addresses

94

Binary Tree Inorder Traversal

95

Unique Binary Search Trees II

96

96

Unique Binary Search Trees

97

Interleaving String

98

Validate Binary Search Tree

99

Recover Binary Search Tree

100

Same Tree

101

Symmetric Tree

102

Binary Tree Level Order Traversal

103

Binary Tree Zigzag Level Order Traversal

104

Maximum Depth of Binary Tree

105

Construct Binary Tree from Preorder and Inorder Traversal

106

Construct Binary Tree from Inorder and Postorder Traversal

107

Binary Tree Level Order Traversal II

108

Convert Sorted Array to Binary Search Tree

109

Convert Sorted List to Binary Search Tree

110

Balanced Binary Tree

111

Minimum Depth of Binary Tree

112

Path Sum

113

Path Sum II

114

Flatten Binary Tree to Linked List

115

Distinct Subsequences

116

Populating Next Right Pointers in Each Node

117

Populating Next Right Pointers in Each Node II

118

Pascal's Triangle

119

Pascal's Triangle II

120

Triangle

121

Best Time to Buy and Sell Stock

122

Best Time to Buy and Sell Stock II

123

Best Time to Buy and Sell Stock III

124

Binary Tree Maximum Path Sum

125

Valid Palindrome

126

Word Ladder II

127

Word Ladder

128

Longest Consecutive Sequence

129

Sum Root to Leaf Numbers

130

Surrounded Regions

131

Palindrome Partitioning

132

Palindrome Partitioning II

133

Clone Graph

134

Gas Station

135

Candy

136

Single Number

137

Single Number II

138

Copy List with Random Pointer

139

Word Break

140

Word Break II

141

Linked List Cycle

142

Linked List Cycle II

143

Reorder List

144

Binary Tree Preorder Traversal

145

Binary Tree Postorder Traversal

146

LRU Cache

147

Insertion Sort List

148

Sort List

149

Max Points on a Line

150

Evaluate Reverse Polish Notation

151

Reverse Words in a String

152

Maximum Product Subarray

153

Find Minimum in Rotated Sorted Array

154

Find Minimum in Rotated Sorted Array II

155

Min Stack

156

Binary Tree Upside Down

157

Read N Characters Given Read4

158

Read N Characters Given Read4 II - Call multiple times

159

Longest Substring with At Most Two Distinct Characters

160

Intersection of Two Linked Lists

161

One Edit Distance

162

Find Peak Element

163

Missing Ranges

164

Maximum Gap

165

Compare Version Numbers

166

Fraction to Recurring Decimal

167

Two Sum II - Input array is sorted

168

Excel Sheet Column Title

169

Majority Element

170

Two Sum III - Data structure design

171

Excel Sheet Column Number

172

188

Best Time to Buy and Sell Stock IV

189

Rotate Array

190

Reverse Bits

191

Number of 1 Bits

198

House Robber

199

Binary Tree Right Side View

200

Number of Islands

201

Bitwise AND of Numbers Range

202

Happy Number

203

Remove Linked List Elements

204

Count Primes

205

Isomorphic Strings

206

Reverse Linked List

207

Course Schedule

208

Implement Trie (Prefix Tree)

209

Minimum Size Subarray Sum

210

Course Schedule II

211

Add and Search Word - Data structure design

212

Word Search II

213

House Robber II

214

Shortest Palindrome

215

Kth Largest Element in an Array

Maximal Square

222

Count Complete Tree Nodes

223

Rectangle Area

224

Basic Calculator

225

Implement Stack using Queues

226

Invert Binary Tree

227

Basic Calculator II

228

Summary Ranges

229

Majority Element II

230

Kth Smallest Element in a BST

231

Power of Two

232

Implement Queue using Stacks

233

Number of Digit One

234

Palindrome Linked List

235

Lowest Common Ancestor of a Binary Search Tree

236

Lowest Common Ancestor of a Binary Tree

237

Delete Node in a Linked List

238

Product of Array Except Self

239

Sliding Window Maximum

240

Search a 2D Matrix II

241

Different Ways to Add Parentheses

242

Valid Anagram

243

Shortest Word Distance

244

Shortest Word Distance II

245

Shortest Word Distance III

246

Strobogrammatic Number

247

Strobogrammatic Number II

248

Strobogrammatic Number III

249

Group Shifted Strings

250

Count Univalue Subtrees

251

Flatten 2D Vector

1.3.3 Leetcode 239: sliding window maximum (2015-08-08 20:33)

August 7, 2015

Spent 20-30 minutes to think about solution first, and then, read the blogs, and understand the best solution, and then try different solutions. Try to get some workout on the C # programming practice.
□□□□□□□□□□□□Б, □□□□□□□□□, □□□, □□□□□□□□□□□□□□□□□□, □□□, □□□□.

1. Blog to read:

http://www.shuatiblog.com/blog/2014/07/27/Sliding-Window-Maximum/

C # code implementation (**double ended queue is implemented using C # LinkedList class**):

https://github.com/jianminchen/slidingWindowMaximum/blob/master/slidingWin dowMaximum1.cs

2. Blog to read

http://bookshadow.com/weblog/2015/07/18/leetcode-sliding-window-maximum/

C # code implementation (C # does not have deque class, so using C # List<int>,

convert Python code implementation to C #, time limit exceeded)

Good workout on C # List<int>, and also, experience different style on removing head element if out of sliding window.

https://github.com/jianminchen/slidingWindowMaximum/blob/master/slidingWi ndowMaximu2.cs

3. Blog to read

http://www.geeksforgeeks.org/maximum-of-all-subarrays-of-size-k/

Method A: naive solution, time complexity O(nw)

https://github.com/jianminchen/slidingWindowMaximum/blob/master/slidingWind owMaximum3.cs

Method B: using Self-Balancing Tree (Time complexity: O(nk), need to write c # code)

https://github.com/jianminchen/slidingWindowMaximum/blob/master/slidingWindo wMaximu4.cs

4. blog:

http://yuanhsh.iteye.com/blog/2190852

http://n00tc0d3r.blogspot.ca/2013/04/sliding-window-maximum.html

Good comment about Deque:

We can use a Deque which allow insertions/deletions on both ends. For a Deque implemented by Circular Array/Buffer or Double Linked List , the basic insert/delete operations run in constant time.

discussion of using heap:

The first thought might be heap.

By maintaining a heap for all numbers in the window can give us a O(*nlogw*)-time solution, where

- building up a heap for initial window takes time O(*wlogw*)
- when window moves to the next number, each insertion and deletion take time O(*logw*) and there are *n*-*w* moves in total.
- after updating the heap, findMax only takes time O(1) since we know the top of heap is the largest.

So, if *w* << *n*, the performance of this solution is good, close to O(*n*); but if w is not that small, say *w = n/3* or *n/4*, the running time goes up to O(*nlogn*).

5. blog:

https://github.com/haoel/leetcode/commit/74d83796aa48cc55d7995b1f9e61db40759 204bb

using C++ multiset in the above solution, so try to convert it to C # class using SortedList

https://github.com/jianminchen/slidingWindowMaximum/blob/master/sliding WindowMaximu5.cs

6. blog:

http://www.mamicode.com/info-detail-927510.html

convert Java Script code to C #; I spent over 12 months to try to be expert on Java Script, so much fun to read the Java Script code again, and enjoyed the blog about the analysis.

Others:

https://github.com/jianminchen/slidingWindowMaximum/blob/master/slidingWind owMaximu5.cs

http://www.cnblogs.com/grandyang/p/4656517.html

http://techinpad.blogspot.ca/2015/05/lintcodeleetcode-sliding-window-maximu m.html

1.3.4 Leetcode: Maximum product subarray (2015-08-10 20:16)

August 10, 2015

Find the contiguous subarray within an array (containing at least one number)
which has the largest product.

For example, given the array [2,3,-2,4],
the contiguous subarray [2,3] has the largest product = 6.

Two approaches, one is dynamic programming DP implementation, another is the greedy method.

DP solution:

https://github.com/jianminchen/MaximumProductSubarray/blob/master/Maximu mProductSubarray1.cs

Greedy algorithm:
read the blog first,

http://fmarss.blogspot.ca/2014/10/leetcode-solution-maximum-product.html

and then, convert it to C # programming language,

https://github.com/jianminchen/MaximumProductSubarray/blob/master/Maxim umProductSubarray2.cs

1.3.5 Java Script book reading and videos (2015-08-10 20:25)

August 10, 2015

Spent 12 months starting from January 2014 to study Java Script, it is most challenging language I learned and finally get used to this functional language, dynamic, loosely type language. Enjoy reading the Java Script code and play with the code.

Here are some resources most helpful to my personal experience:

Books I read, some of them only a few pages here and there:

1. [O'Reilly] - JavaScript. The Definitive Guide, 6th ed. - [Flanagan]

Only read 200 pages of the book (total pages: around 1000 pages), spent 2-3 months to read, practice

1. JavaScript for PHP Developers

Tried examples in the book, and practiced some. Focused on first 3 chapters. Great book!

1. JavaScript Patterns
2. Javascript cookbook
3. [Manning] - Secrets of the JavaScript Ninja - [Resig]
4. O'reilly - Head First Javascript
5. Maintainable JavaScript

6. Java Script good parts
7. Testable JavaScript (one of my favorite books)

Videos I watched:

1. Douglas Crokford: The Javascript Programming language and most of the videos from him on Youtube.com

https://www.youtube.com/watch?v=v2ifWcnQs6M

1. Learning to love Javascript and a lot of other google videos about Javascript

https://www.youtube.com/watch?v=seX7jYI96GE

https://www.youtube.com/watch?v=hQVTIJBZook

1.3.6 Julia's reading list of CSS, html, Java Script (2015-08-12 19:04)

CSS/ Jquery/ Html reading list

Julia read several articles to expand knowledge about CSS, html, Java Script, and here is her reading list compiled on August 12, 2015:

1. August 28, 2013

Read the article web standards curriculum

http://www.w3.org/wiki/Web _Standards _Curriculum

1. May 9, 2013, read the article, and then memorize the selectors.

http://code.tutsplus.com/tutorials/the-30-css-selectors-you-must-memori ze–net-16048

1. Visual formatting model (August 26, 2013):

http://www.w3.org/TR/CSS21/visuren.html

1. CSS 2.1 primer

http://www.w3c.it/education/2012/upra/documents/css.pdf

1. Cascading Style Sheets articles and tutorials (August 28, 2013)

http://www.w3.org/Style/CSS/learning

1. Inheritance and cascade (Nov. 18, 2013): (learning specificity calculation in detail)

http://www.w3.org/wiki/Inheritance _and _cascade

1. Basic Structures of a web page (Nov. 27, 2013):

http://www.sitepoint.com/web-foundations/basic-structure-of-a-web-page/

1. CSS absolute and fixed positioning (Nov. 20, 2013)

http://www.w3.org/wiki/CSS _absolute _and _fixed _positioning

1. Javascript Guide (Mozzila Foundation US)

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide

1. CSS pocket book: (March 24, 2015)

http://www.amazon.ca/CSS-Pocket-Reference-Eric-Meyer/dp/1449399037

1.3.7 Blogging benefits (2015-08-12 20:36)

Julia likes to keep her own research on writing benefits as a software programmer. So, she keeps the log of her research.

August 12, 2015

Read two articles about benefits about blogging as a software programmer. Great ideas in the blog.

http://kb.cnblogs.com/page/517038/

http://kb.cnblogs.com/page/526625/

Dec. 17, 2015
Take some notes from above blogs:

□□□□□□□□□□□□

□□□□□□□□□□Bloom□□□□□□□□□□□□□□□□□□□□□□□□□□□□□6□□□, □□□□□□□□□□□□□□□æ□□□□□□□□□□□□□□ □□□□□□□□□□□□□□□-□□□□□

□□□□□□□□□□G

□□□□□□□□□□

□□□

January 3, 2015

Thomas Friedman: Lessons Learned After 20 Years of Writing Columns

https://www.youtube.com/watch?v=dxXx3XvNVWA &list=PLxq _lXOUlvQAwaY _9K4ZFH9Xdar9WzCaL &index=19

Note:
1. Learn to listen; person can sense if you listen, respect him/her; wait until he/she finishes the talking.
2. Always be a big tipper. Sign of respect. Always be nice to helper.
3. Always call your mother. I wish I can call mine.

January 12, 2015

Hiring Rockstars by Roger Philby

https://www.youtube.com/watch?v=R22dJ7bn-pU &list=PLgYNPs-V9YFPqcnEvbIy5hFE40BjxMbjw &index=2

A person loves to write, then the person must love to read, and then, the person must be intellect curious. That is one of 5 things in the talk:

1. intellect value
- how we judge people - how we collaborate - highly collaborate - not collaborate - collaboration - need to know that, how to measure

fast/decisive - consensus - relationship - seek colloaborate

value in your business - big company

3. conformity -

4. Motivation -

5. Experience -

High performance -

March 23, 2016

http://onstartups.com/tabid/3339/bid/14208/Why-Every-Entrepreneur-Should-Writ e-and-9-Tips-To-Get-Started.aspx

http://www.c-sharpcorner.com/article/why-every-developer-should-write/

1.3.8 Leetcode: questions and favorite blogs (2015-08-13 23:28)

August 13, 2015,

Go over the leetcode questions one by one, find one blog/ more to help me fully understand the problem. □□□□□□, □□□□, □□□, □□□□□□□, □□□□□□□; □□□□, □□□□□□; □□□□□□□□□□□□□.

23 Merge k Sorted Lists hard

(just do it) http://www.cnblogs.com/TenosDoIt/p/3673188.html

22 Generate Parentheses (fight for dream) http://blog.csdn.net/fightforyourdream/article/details/14159435

http://blog.csdn.net/linhuanmars/article/details/19873463 (more easy to understand) (one line error)

http://yucoding.blogspot.ca/2013/01/leetcode-question-31-generate.html

21 Merge two sorted lists

http://fisherlei.blogspot.ca/2013/03/leetcode-merge-two-sorted-lists-solution .html

http://www.geeksforgeeks.org/merge-two-sorted-linked-lists/

http://www.cnblogs.com/springfor/p/3862040.html (show how to optimize the code! Very good.)

22 valid parentheses easy

http://fisherlei.blogspot.ca/2013/01/leetcode-valid-parentheses.html

http://yucoding.blogspot.ca/2013/02/leetcode-question-121-valid-parentheses .html

19 remove Nth node from End of List

http://harrifeng.github.io/algo/leetcode/remove-nth-node-from-end-of-list.htm l (Excellent!)

http://www.cnblogs.com/springfor/p/3862219.html(very clever tip for faster pointer)

Here are C # code folder:
https://github.com/jianminchen/Leetcode _C-

1.3.9 Leetcode questions 1 - 10 (2015-08-16 22:16)

August 15

1. Two sum
C #
https://github.com/jianminchen/Leetcode _C-/blob/master/1TwoSum.cs

2. Add two numbers
C #
https://github.com/jianminchen/Leetcode _C-/blob/master/2AddTwoNumbers.cs

3. Longest substring without repeating characters

blog:

128

http://blog.csdn.net/linhuanmars/article/details/19949159

read all the comments, and play with code using C #, at least 20 minutes; get more understanding this time.

https://github.com/jianminchen/Leetcode _C-/blob/master/3LongestSubstringWithoutRepeating.cs

4. Median of two sorted array (8/17/2015)

C # implementation:
https://github.com/jianminchen/Leetcode _C-/blob/master/4MedianOfTwoSortedArrays.cs

5 longest palindromic substring

C #
https://github.com/jianminchen/Leetcode _C-/blob/master/5LongestPalindromicSubstring.cs

9 Palindrome number
blog:
http://codeganker.blogspot.ca/2014/02/palindrome-number-leetcode.html

C # code:
https://github.com/jianminchen/Leetcode _C-/blob/master/9palindromeNumber.cs

10 Regular Expression Matching
blog:

http://bangbingsyb.blogspot.ca/2014/11/leetcode-regular-expression-m atching.html

C # code:
https://github.com/jianminchen/Leetcode _C-/blob/master/10RegularExpressionMatching.cs

1.3.10 Leetcode question 20 - 30 (2015-08-18 23:27)

August 18, 2015

Work on those ten questions quickly, spend 20 minutes on each question; Do not over analyze the problem, try to get basic idea and implementation tips through blogs first.

22 Generate Parentheses

https://github.com/jianminchen/Leetcode _C-/blob/master/GenerateParentheses _No22.cs

23 Merge K Sorted Lists

https://github.com/jianminchen/Leetcode _C-/blob/master/MergeKSortedLists _A _No23.cs

https://github.com/jianminchen/Leetcode _C-/blob/master/MargeKSortedLists _B _No23.cs

24 Swap nodes in pairs

https://github.com/jianminchen/Leetcode _C-/blob/master/24SwapNodesInPairs.cs

C # implementation:
https://github.com/jianminchen/Leetcode _C-/blob/master/24SwapNodesInPairs.cs

25 Reverse Nodes in k-Group

26 Remove Duplicates from Sorted Array

27 Remove Element

28 Implement strStr()

29 Divide Two Integers

30 Substring with Concatenation of All Words

1.3.11 Leetcode questions quick review from No. 30 - 40 (2015-08-20 23:13)

August 19, 2015

30 Substring with Concatenation of All Words

http://segmentfault.com/a/1190000002625580

http://bangbingsyb.blogspot.ca/2014/11/leetcode-search-for-range.html

31 Next Permutation

32 Longest Valid Parentheses

http://codeganker.blogspot.ca/2014/03/longest-valid-parentheses-leetco de.html

http://bangbingsyb.blogspot.ca/2014/11/leetcode-longest-valid-parentheses.h tml

http://blog.csdn.net/worldwindjp/article/details/39460161

http://shanjiaxin.blogspot.ca/2014/04/longest-valid-parentheses-leetcode.ht ml

C # code:
https://github.com/jianminchen/Leetcode _C-/blob/master/32LongestValidParentheses.cs

33 Search in Rotated Sorted Array

34 search for range

http://bangbingsyb.blogspot.ca/2014/11/leetcode-search-for-range.html

35 Search Insert Position

36 Valid Sudoku

37 Sudoku Slover

Try 10 various implementations:

http://juliachencoding.blogspot.ca/2015/07/leetcode-sudoku-solver.html

38 count and say

http://www.cnblogs.com/springfor/p/3889221.html

39 Combination Sum

40 Combination Sum II

1.3.12 Leetcode questions from 61 - 69 (2015-08-21 22:14)

August 21, 2015

Speed up study leetcode questions, 10 question a time, in 2-3 hours. And see if I can learn something quickly.

61 Rotate List

http://bangbingsyb.blogspot.ca/2014/11/leetcode-rotate-list.html

julia's C # implementation practice:
https://github.com/jianminchen/Leetcode _C-/blob/master/61RotateList.cs

62 Unique Paths

http://bangbingsyb.blogspot.ca/2014/11/leetcode-unique-paths-i-ii.html

http://siddontang.gitbooks.io/leetcode-solution/content/dynamic _programming/unique _paths.html

http://yucoding.blogspot.ca/2013/04/leetcode-question-116-unique-path-i .html

DP algorithm in detail, time complexity and space complexity analysis is great in the following blog:

http://codeganker.blogspot.ca/2014/03/unique-paths-leetcode.html

63 Unique Paths II

64 Minimum Path Sum

discussion of using 2 dimension array or using one dimension array, still not clear.

http://fisherlei.blogspot.ca/2012/12/leetcode-minimum-path-sum.html

understand this blog on DP solutions, one is space O(n^2), one is space O(n).

http://joycelearning.blogspot.ca/2013/10/leetcode-minimum-path-sum.html

http://shanjiaxin.blogspot.ca/2014/03/minimum-path-sum-leetcode.html

65 Valid number

ideas for the solution from the following blog:

3 flags are set: num, exp, dot, and then,

1. if there is e flag, then there is a digit before e flag, cannot have any e.

Also, there is a digit followed.

2. if . shows up, then it is a small fraction number, so neither . nor e is before .

3. if +, -, then it must be first one, or the previous on is e,

for example: "005047e+6".

The above summary is from the following blog:

http://www.yanyulin.info/pages/2014/10/38958011441714.html

code is here:

http://www.jiuzhang.com/solutions/valid-number/

Julia's C # code practice:

https://github.com/jianminchen/Leetcode _C-/blob/master/65ValidNumber.cs

blogs:

https://leetcodenotes.wordpress.com/2013/11/23/leetcode-valid-number/

http://www.cnblogs.com/TenosDoIt/p/3475305.html
http://www.cnblogs.com/chasuner/p/validNumber.html
https://github.com/fuwutu/LeetCode/blob/master/Valid %20Number.cpp

http://rleetcode.blogspot.ca/2014/01/valid-number-java.html

66 Plus number

67 Add Binary

Totally forget I did this implementation. So, I have to learn everything about this problem again, read more blogs this time.

https://github.com/jianminchen/addBinary/blob/master/Program.cs

my favorite solution is the following blog:
http://fisherlei.blogspot.ca/2013/01/leetcode-add-binary.html

julia's C # code:
https://github.com/jianminchen/Leetcode _C-/blob/master/67AddBinary2.cs

http://bangbingsyb.blogspot.ca/2014/11/leetcode-add-binary.html

the code should be more short, but the discussion and idea in the code is very clear.
http://siddontang.gitbooks.io/leetcode-solution/content/string/add _binary.html

Good quality code, try it myself later.
http://www.jiuzhang.com/solutions/add-binary/

julia's c # implementation:
https://github.com/jianminchen/Leetcode _C-/blob/master/67AddBinary.cs

68 Text Justification

69 sqrt(x)

http://yucoding.blogspot.ca/2013/03/leetcode-question-102-sqrtx.html

http://codeganker.blogspot.ca/2014/02/sqrtx-leetcode.html

C # implementation:

https://github.com/jianminchen/Leetcode _C-/blob/master/69Sqrt(x).cs

70 Climbing Stairs

http://juliachencoding.blogspot.ca/2015/07/leetcode-climbing-stairs.html

1.3.13 Leetcode questions 11 - 20 (2015-08-22 13:09)

August 22, 2015

11 Container with Most water
blog:
http://www.cnblogs.com/TenosDoIt/p/3812880.html

C #:
https://github.com/jianminchen/Leetcode _C-/blob/master/11ContainerWithMostWater.cs

12. Integer to Roman

13 Roman to Integer

https://github.com/jianminchen/Leetcode _C-/blob/master/13RomanToInteger.cs

14. LOngest Common Prefix

https://github.com/jianminchen/Leetcode _C-/blob/master/LongestCommonPrefix _No14.cs

15. 3Sum

16 3 Sum Closest

https://github.com/jianminchen/Leetcode _C-/blob/master/3sumCloset.cs

17 Letter Combinations of a phone number (DFS)

http://www.cnblogs.com/grandyang/p/4452220.html
Analysis from the above blog:

□□□□□□□□□□□□-ĥ'□□□□□□□□2□9□□□□-X'□□□□□□□□□□□□□□?□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Path Sum II □□□□□□□□□□ □ Subsets II □□□□□ □ Permutations □□□ □ Permutations II □□□□□ □ Combinations □□□ □ Combination Sum □□□□ □ Combination Sum II □□□□□□ □□□□□□□□□□Recursion□□□□□□□□□□□□□□□□□□□□□□□□□□□-èh□□□□□□□□□□□□□□□□□level□□□□□□□□□ □'□□□□□□□□□□□□□□□□□□□□□□□□□□□□

https://github.com/jianminchen/Leetcode _C-/blob/master/LetterCombinationOfAPhoneNumber.cs

18 4 sum

19 Remove Nth Node From End of List

20 **Valid Parentheses**

http://blog.csdn.net/fightforyourdream/article/details/13011825

C # code:
https://github.com/jianminchen/Leetcode _C-/blob/master/20ValidParentheses.cs
January 20, 2016
https://github.com/jianminchen/Leetcode _C-/blob/master/20ValidParentheses _B.cs

favorite blogs to read: January 20, 2016
http://bangbingsyb.blogspot.ca/2014/11/leetcode-valid-parentheses.html

http://www.acmerblog.com/leetcode-solution-valid-parentheses-6316.html

http://blog.csdn.net/foreverling/article/details/49685177?hmsr=toutiao.io &utm _medium=toutiao.io &utm _source=toutiao.io

http://segmentfault.com/a/1190000003481208

http://harrifeng.github.io/algo/leetcode/valid-parentheses.html

http://www.jiuzhang.com/solutions/valid-parentheses/

So, ready to work on the next algorithm about parentheses:
Leetcode 32: longest valid parentheses -
http://codeganker.blogspot.ca/2014/03/longest-valid-parentheses-leetcode.html

1.3.14 Leetcode questions from 41 - 50 (2015-08-22 13:25)

August 22, 2015

41 Frist Missing Positive

42 Trapping Rain Water

43 Multiply Strings

44 Wildcard Matching

45 Jump Game II

46 Permutations

47 Permutations II

48 Rotate Image

49 Anagrams

50 Pow(x, n)

http://juliachencoding.blogspot.ca/2015/06/leetcode-powerxn.html

C # implementation:

https://github.com/jianminchen/powerOfN/blob/master/Program.cs

1.3.15 Leetcode questions: 70 - 80 (2015-08-22 14:12)

August 22, 2015

70 climbing stairs

71 Simplify Path

72 Edit Distance

http://juliachencoding.blogspot.ca/2015/06/leetcode-edit-distance.html

73 Set Matrix Zeroes

74 Search a 2D Matrix

75 Sort Colors

76 Minimum Window Substring

77 Combinations

78 Subsets

79 Word Search

http://juliachencoding.blogspot.ca/2015/07/leetcode-word-search.html

1.3.16 Leetcode question 80-90 (2015-08-22 14:20)

August 22, 2015

80 Remove duplicates from Sorted Array II

81 Search in Rotated Sorted Array II

82 Remove Duplicates from Sorted List II

83 Remove Duplicates from Sorted List

84 Largest Rectangle in Histogram

http://juliachencoding.blogspot.ca/2015/06/leetcode-largest-rectangle-in -histogram.html

85 Maximal Rectangle

86 Partition List

87 Scramble String

http://juliachencoding.blogspot.ca/2015/06/leetcode-scramble-string.html

need to write first C # implementation for the problem.

88 Merge Sorted Array

89 Gray Code

90 Subsets II

1.3.17 Leetcode questions 90 - 100 (2015-08-22 15:36)

August 22, 2015

90 subsets II

91 Decode ways

92 Reverse Linked List II

93 Restore IP Addresses

94 Binary Tree Inorder Traversal

95 Unique Binary Search Trees II

96 Unique Binary Search Trees

97 Interleaving String

98 Validate Binary Search Trees

99 Recover Binary Search Tree

100 Same Tree

– Julia's practice:

94 Binary Tree Inorder Traversal

Julia's C # implementation:
https://github.com/jianminchen/leetcode-tree/blob/master/TreeDemo. cs

95 Unique Binary Search Trees II

https://github.com/jianminchen/Leetcode _C-/blob/master/95UniqueBinarySearchTreeII.cs

97 Interleaving String
the blog containing reading list:
http://juliachencoding.blogspot.ca/2015/06/leetcode-interleave-string.html

Need to write first C # implementation.

99 Recover Binary Search Tree

blogs:

http://www.lifeincode.net/programming/leetcode-recover-binary-search-tr ee-java/
http://www.cnblogs.com/AnnieKim/archive/2013/06/15/MorrisTraversal.html

C # implementations:

https://github.com/jianminchen/Leetcode _C-/blob/master/99RecoverBinarySearchTree.cs

https://github.com/jianminchen/Leetcode _C-/blob/master/99RecoverBinarySearchTreeB.cs

work on extracting small functions, and then, understand the algorithm better.
https://github.com/jianminchen/Leetcode _C-/blob/master/99RecoveryBinarySearchTree _C.cs

100 same tree
Two implementations,one recursive, one iterative solution.
https://github.com/jianminchen/Leetcode _C-/blob/master/100SameTree.cs

1.3.18 Leetcode questions 100 - 110 (2015-08-22 15:49)

August 22, 2015

100 Same Tree

101 Symmetric Tree

102 Binary Tree Level Order Traversal

103 Binary Tree Zigzag Level Order Traversal

104 Maximum Depth of Binary Tree

105 Construct Binary Tree from Preorder and Inorder Traversal

106 Constructu Binary Tree from Inorder and Postorder Traversal

107 Binary Tree level Order Traversal II

108 Convert Sorted Array To Binary Search Tree

109 Convert Sorted List to Binary Search Tree

110 Balanced Binary Tree

_

101 Symmetric Tree (Sept. 8, 2015)

https://github.com/jianminchen/Leetcode _C-/blob/master/101SymmetricTreeA.cs

102 Binary Tree Level Order Traversal

6 different solutions:

http://juliachencoding.blogspot.ca/2015/07/leetcode-binary-tree-level-o rder.html

103 Binary Tree Zigzag Level Order Traversal

http://juliachencoding.blogspot.ca/2015/06/leetcode-zigzag-order-travers al-of.html

104 Maximum Depth of Binary Tree

https://github.com/jianminchen/Leetcode _C-/blob/master/104MaximumDepthOfBinaryTree.cs

106 Constructu Binary Tree from Inorder and Postorder Traversal

https://github.com/jianminchen/Leetcode _C-/blob/master/106ConstructuBTreeFromInorderPostOrderTraversal.cs

second implementation using C #:

https://github.com/jianminchen/Leetcode _C-/blob/master/106ConstructBTreeFromInOrderPostOrderTraversal _B.cs

Leetcode 106: January 2, 2015
Further reading: (1 hour)
Serialize and Deserialize a Binary Tree
http://www.geeksforgeeks.org/serialize-deserialize-binary-tree/

http://www.cs.usfca.edu/ brooks/S04classes/cs245/lectures/lecture11.pdf

108 Convert sorted array to binary search tree (No. 108)

https://github.com/jianminchen/Leetcode _C-/blob/master/108ConvertSortedArrayToBinarySearchTree.cs

109 Convert sorted list to binary search tree (No. 109)

8/25/2015

Read the following blogs:

http://articles.leetcode.com/2010/11/convert-sorted-list-to-balanced-binary .html

C #, bottom up, time O(n), space O(log n) solution - best solution:

https://github.com/jianminchen/Leetcode _C-/blob/master/109ConvertSortedListToBinarySearchTreeB.cs

C #, top down, time O(n^2), space O(long n) solution - naive solution:

worked on code 2 times, first time, the calculation is kind of messy, then, worked on Leetcode question 108, get the idea to make it more simple; tips like len/2 only shows once, afterwards, use m instead. Just need to improve coding, think to make it more abstract, simple.

https://github.com/jianminchen/Leetcode _C-/blob/master/109ConvertSortedListToBinarySearchTreeC.cs

9/21/2015
Review the best solution, and then, totally forgot the bottom up solution idea. So, update the code with more comment.

Need to review more about bottom up/ top down solution in tree problems. Get more experience on bottom-up solution, read some articles about it.

1.3.19 Leetcode question 111 - 120 (2015-08-22 16:18)

August 22, 2015

111 Minimum Depth of Binary Tree

112 Path Sum

113 Path Sum II

114 Flatten Binary Tree to Linked List

115 Distinct Subsequences

116 Populating Next Right Pointers in Each Node

117 Populating Next Right Pointers in Each Node II

118 Pascal's Triangle

119 Pascal's Triangle II

120 Triangle

1.3.20 Leetcode questions 120 - 130 (2015-08-22 16:20)

August 22, 2015

120 Triangle

write a blog, need to write C # code.
http://juliachencoding.blogspot.ca/2015/06/leetcode-triangle.html

121 Best Time to Buy and Sell Stock

122 Best Time to Buy and Sell Stock II

123 Best Time to Buy and Sell Stock II I

124 Binary Tree Maximum Path Sum

http://juliachencoding.blogspot.ca/2015/07/leetcode-maximum-binary-tree-path- sum.html

Tree Maximum Path Sum
http://juliachencoding.blogspot.ca/2015/07/itint5-tree-maximum-path-sum.html

Tree maximum path sum:

125 Valid Palindrome

126 Word Ladder II

127 Word Ladder

http://juliachencoding.blogspot.ca/2015/06/leetcode-word-ladder.html

128 Longest Consecutive Sequence

129 Sum Root to Leaf Numbers

130 Surrounded Regions

_
Question No: 124

C # implementation:
https://github.com/jianminchen/Leetcode _C-/blob/master/124BinaryTreeMaximumPathSum.cs

Maybe, similar problem: ITINT5 Tree Maximum Path Sum

http://juliachencoding.blogspot.ca/2015/07/itint5-tree-maximum-path-sum.html

1.3.21 Leetcode question 130 - 140 (2015-08-22 16:46)

August 22, 2015

130 Surrounded Regions

131 Palindrome Partitioning

132 Palindrome Partitioning II

133 Clone Graph

134 Gas Station

135 Candy

http://juliachencoding.blogspot.ca/2015/06/leetcode-candy _9.html

136 Single Number

137 Single Number II

138 Copy List with Random Pointer

139 Word Break

140 Word Break II

1.3.22 Leetcode question 140 - 150 (2015-08-22 16:52)

August 22, 2015

140 Word Break II

141 Linked List Cycle

142 Lined List Cycle II

143 Reorder List

144 Binary Tree Preorder Traversal

http://juliachencoding.blogspot.ca/2015/06/leetcode-binary-tree-preorder -traversal.html

145 Binary Tree Postorder Traversal

http://juliachencoding.blogspot.ca/2015/06/leetcode-post-order-binary-tr ee.html

146 LRU Cache

147 Insertion Sort List

148 Sort List

149 Max Points on a Line

150 Evaluate Reverse Polish Notation

1.3.23 Leetcode question 151 - 160 (2015-08-22 17:06)

August 22, 2015

151 Reverse words in a string

152 Maximum Product Subarray

http://juliachencoding.blogspot.ca/2015/08/leetcode-maximum-product-subar ray.html

153 Find Minimum in Rotated Sorted Array

154 Find Minimum in Rotated Sorted Array II

155 Min Stack

156 Binary Tree Upside Down

157 Read N Characters Given Read 4

158 Read N Characters Given Read 4 II - Call multiple times

159 Longest Substring with At Most Two Distinct Characters

160 Intersection of Two Linked List

1.3.24 Leetcode questions 161 - 170 (2015-08-22 17:38)

August 22, 2015

161 One Edit distance

162 Find Peak Element

http://juliachencoding.blogspot.ca/2015/06/leetcode-question-find-peak-ele ment.html

need to write first C # implementation for this problem.

163 Missing Ranges

164 Maximum Gap

165 Compare Version Numbers

166 Fraction to Recurring Decimal

167 Two Sum II - Input array is sorted

168 Excel Sheet Column Title

169 Majority Element

170 Two Sum III - Data Structure Design

1.3.25 Leetcode question 51 - 60 (2015-08-22 17:48)

August 22, 2015

51 N-Queens

https://github.com/jianminchen/AlgorithmsPractice/blob/master/EightPuzzel Queen.cs
https://github.com/jianminchen/Leetcode _C-/blob/master/51NQueenProblems.cs

http://juliachencoding.blogspot.ca/2015/06/algorithm-eight-puzzle-queen.htm l

52 N-Queens II

53 Maximum Subarray

54 Spiral Matrix

55 Jump Game

56 Merge Intervals

57 Insert Intervals

C # implementation:

https://github.com/jianminchen/Leetcode _C-/blob/master/57InsertIntervals.cs

58 Length of Last Word

https://github.com/jianminchen/Leetcode _C-/blob/master/58LengthOfLastWord.cs

59 Spiral Matrix II

60 Permutataion Sequence

1.3.26 Tree algorithms review (2015-08-23 13:29)

August 23, 2015

Here is the list of tree questions worked on:

1. Lowest common ancestor in Binary Tree (January 6, 2016 - Leetcode question: 236 medium)

https://github.com/jianminchen/LowestCommonAncestorInBinaryTree/blob/maste r/LowestCommonAncestorB.cs

Lowest common ancestor in Binary **Search** Tree (January 6, 2016 - Leetcode question: 235 easy)
https://github.com/mengli/leetcode/blob/master/LowestCommonAncestorOfaBinary SearchTree.java

https://github.com/jianminchen/Leetcode _C-/blob/master/235LowestCommonAncestorBSearchTree.cs

https://github.com/jianminchen/Leetcode _C-/blob/master/235LowestCommonAncestorBinarySearchTree.cs

2. Leetcode 102: Binary tree level order traversal

a few of implementations:

https://github.com/jianminchen/BTreeLevelOrderTraversal

January 7, 2016 review

https://github.com/jianminchen/BTreeLevelOrderTraversal/blob/master/BTreeLe velOrderTraversal4.cs

3. C # files:

TreeDemo.cpp

Pre Order Traversal

In Order Traversal

Post Order Traversal

Breadth First Traversal

Breadth First Traversal Iterative

Pre Order Traversal Iterative

Post Order Traversal Iterative*

In Order Traversal Iterative (A, B two versions)

Count One Child Node

Recover Binary Tree

Invert Binary Tree

Invert Binary Tree Iterative

–

Tree Post Order Iterative

Tree In order Iterative

https://github.com/jianminchen/leetcode-tree

4. Tree Max Path Sum

https://github.com/jianminchen/TreeMaxPathSum/blob/master/Program.cs

5. Linked List To Binary Search Tree

https://github.com/jianminchen/LinkedListToBST/blob/master/Program.cs

6. Zigzag order traversal of a binary Tree

http://juliachencoding.blogspot.ca/2015/06/leetcode-zigzag-order-traversal-of .html

7. Maximum Binary Tree Path Sum

How to come out the idea using max value cross root? Cannot recall the idea.

Read the blog again (8/25/2015):

http://blog.unieagle.net/2012/12/09/leetcode %E9 %A2 %98 %E7 %9B %AE %EF %BC %9Abinary-tree-maximum-path-sum/

https://github.com/jianminchen/Leetcode _C-/blob/master/124BinaryTreeMaximumPathSum.cs

The problem can be extended to a tree, not just a binary tree. The solution is here:
http://juliachencoding.blogspot.ca/2015/07/itint5-tree-maximum-path-sum.html

8. ININT5: tree maximum path sum

http://juliachencoding.blogspot.ca/2015/07/itint5-tree-maximum-path-sum.html

9. Binary Tree Maximum Distance, diameter of binary

http://juliachencoding.blogspot.ca/2015/06/binary-tree-maximum-distance-dia meter.html

10. Morris Inorder Traversal

http://juliachencoding.blogspot.ca/2015/06/morris-inorder-traversal-bina ry-tree.html

11. Convert sorted List to binary search tree (I) (Leetcode question: No. 109)

http://juliachencoding.blogspot.ca/2015/06/leetcode-convert-sorted-list-to.ht ml

12. Convert sorted list to binary search tree (II) (over 3 hours debugging to find a bug)

(Leetcode question: No. 109)

http://juliachencoding.blogspot.ca/2015/06/leetcode-sorted-list-to-binary-sea rch.html

August 23, 2015

It is so good to have chance to review post order traversal iterative solution. So, Julia had chance to read more about the problem, understand another solution using space O(log(n)),

use a *prev* variable to keep track of the previously-traversed node.

Also, she had chance to work out using two stacks solution for post order traversal iterative solution.

Like this, First stack, root -> left child -> right child, the order of getting into first stack,

however, second stack, root-> right child -> left child, the order of getting into second stack.

since left, right child both get into the first stack, then, pop out two of them in reverse order.

* Read the following blog to help understand post order traversal iterative

http://articles.leetcode.com/2010/10/binary-tree-post-order-traversal.ht ml

C # implementation of post order traversal iterative solution, using prev variable to help:

https://github.com/jianminchen/leetcode-tree/blob/master/TreePostOrderIterat ive _PrevVarirable.cs

August 24, 2015

13. Convert sorted list to binary search tree (No. 109)
Read the following blogs:
http://articles.leetcode.com/2010/11/convert-sorted-list-to-balanced-binary .html

C #, bottom up, time O(n), space O(log n) solution:
https://github.com/jianminchen/Leetcode _C-/blob/master/109ConvertSortedListToBinarySearchTreeB.cs

C #, top down, time O(n^2), space O(long n) solution:

worked on code 2 times, first time, the calculation is kind of messy, then, worked on Leetcode question 108, get the idea to make it more simple; tips like len/2 only shows once, afterwards, use m instead. Just need to improve coding, think to make it more abstract, simple.

https://github.com/jianminchen/Leetcode _C-/blob/master/109ConvertSortedListToBinarySearchTreeC.cs

14. Convert sorted array to binary search tree (No. 108)

https://github.com/jianminchen/Leetcode _C-/blob/master/108ConvertSortedArrayToBinarySearchTree.cs

August 27, 2015
Leetcode Tree questions (29 questions):

Binary Tree Inorder Traversal

95

Unique Binary Search Trees II

96

Unique Binary Search Trees

98

Validate Binary Search Tree

99

Recover Binary Search Tree

100

Same Tree

101

Symmetric Tree

102

Binary Tree Level Order Traversal

103

Binary Tree Zigzag Level Order Traversal

104

Maximum Depth of Binary Tree

https://github.com/jianminchen/Leetcode_C-/blob/master/104MaximumDepthOfBinaryTree.cs

105

Construct Binary Tree from Preorder and Inorder Traversal

106

Construct Binary Tree from Inorder and Postorder Traversal

107

Binary Tree Level Order Traversal II

108

Convert Sorted Array to Binary Search Tree

109

Convert Sorted List to Binary Search Tree

110

Balanced Binary Tree

111

Minimum Depth of Binary Tree

114

Flatten Binary Tree to Linked List

173

Binary Search Tree Iterator

199

Binary Tree Right Side View

208

Implement Trie (Prefix Tree)

222

Count Complete Tree Nodes

226

Invert Binary Tree

235

Lowest Common Ancestor of a Binary Search Tree

236

Lowest Common Ancestor of a Binary Tree

156

250

Count Univalue Subtrees

More practice using C#:
August 28, 2015

104

Maximum Depth of Binary Tree

https://github.com/jianminchen/Leetcode_C-/blob/master/104MaximumDepthOfBinaryTree.c

95

Unique Binary Search Trees II

https://github.com/jianminchen/Leetcode_C-/blob/master/95UniqueBinarySearchTreeII.cs

98

Validate Binary Search Tree

Blogs to read:

http://blog.csdn.net/likecool21/article/details/23271621
https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

favorite blog about this problem:
http://huntfor.iteye.com/blog/2070278

solutions included in C# practice:

1. recursive, time O(n), using beta-alpha pruning?

2. recursive, use long Max value / min Value instead of int's to avoid the bug

3. use inorder traversal output to help checking BST

4. same as 3, value TreeNode variable

5. sames as 3,

6. same as 3,

7. same as 3, but use iterative solution

8. Great analysis, brute force solution, time complexity O(n^2) vs beta-alpha pruning time O(n)

https://github.com/jianminchen/Leetcode_C-/blob/master/98ValidateBinarySearchTree.cs

99 Recover Binary Search Tree - August 30, 2015

read blogs:

http://www.lifeincode.net/programming/leetcode-recover-binary-search-tree-java/
http://www.cnblogs.com/AnnieKim/archive/2013/06/15/MorrisTraversal.html

C# implementations:

https://github.com/jianminchen/Leetcode_C-/blob/master/99RecoverBinarySearchTree.cs

https://github.com/jianminchen/Leetcode_C-/blob/master/99RecoverBinarySearchTreeB.cs

work on extracting small functions, and then, understand the algorithm better.
https://github.com/jianminchen/Leetcode_C-/blob/master/99RecoveryBinarySearchTree_C.cs

100 same tree

blogs to read:

http://blog.csdn.net/linhuanmars/article/details/22839819

http://www.cnblogs.com/lautsie/p/3247097.html

http://www.cnblogs.com/TenosDoIt/p/3440753.html

julia's implementation in C#
https://github.com/jianminchen/Leetcode_C-/blob/master/100SameTree.cs

Sept. 5, 2015

Morris order post order traversal

blogs to read:

http://www.cnblogs.com/AnnieKim/archive/2013/06/15/MorrisTraversal.html

C# implementation:
https://github.com/jianminchen/MorrisOrder/blob/master/MorrisPostOrder.cs

Morris order in order traversal

https://github.com/jianminchen/MorrisInOrderTraverse/blob/master/Program.cs

Sept. 7, 2015

101 Symmetric Tree

this one is good to follow

use queue

http://www.cnblogs.com/TenosDoIt/p/3440729.html

LinkedList - as queue - Java

http://www.cnblogs.com/springfor/p/3879595.html

julia's practice: (Sept. 8, 2015)
https://github.com/jianminchen/Leetcode_C-/blob/master/101SymmetricTreeA.cs

105 Construct binary tree from preorder and inorder traversal

106 Construct binary tree from inorder and postorder traversal

http://siddontang.gitbooks.io/leetcode-solution/content/tree/construct_binary_t
ree.html

C# practice:
https://github.com/jianminchen/Leetcode_C-/blob/master/106ConstructuBTreeFromInorderPostOrderTraversal.cs

second implementation using C#:
https://github.com/jianminchen/Leetcode_C-/blob/master/106ConstructBTreeFromInOrderPostOrderTraversal_B.cs
Sept. 13, 2015

208 Implement Trie (prefix tree)

blogs:
http://pisxw.com/algorithm/Implement-Trie-(Prefix%20Tree).html
http://www.jyuan92.com/blog/leetcode-implement-trie-prefix-tree/
August 25, 2015
January 5, 2015
Preorder traversal of ternary tree
https://github.com/jianminchen/TreeAlgorithms/blob/master/ternaryTreeTraversal.cs

More reading about ternary tree:
https://en.wikipedia.org/wiki/Ternary _search _tree

1.3.27 String functions review (2015-08-23 14:22)

August 23, 2015

1. stringDemo.cpp

Including

atoi 5 versions of implementation

https://github.com/jianminchen/stringDemo/blob/master/Program.cs

2. Scramble string:

https://github.com/jianminchen/scramble-string

3. strstr

Boyer-Moore algorithm

https://github.com/jianminchen/stringDemo/blob/master/Program.cs

Read the string function website and get ideas:

http://www-igm.univ-mlv.fr/ lecroq/string/

http://algs4.cs.princeton.edu/53substring/

http://zjalgorithm.blogspot.ca/2014/12/leetcode-in-java-implement-strstr. html

Need a test case to help me figure out Boyer-Moore algorithm again on August 23, 2015.

Here is a short one for me to memorize the idea:
http://www.personal.kent.edu/ rmuhamma/Algorithms/MyAlgorithms/StringMatch/b oyerMoore.htm

read the article quickly in 20 minutes on August 23, 2015
http://web.cs.ucdavis.edu/ gusfield/cs224f11/bnotes.pdf

Dec. 12, 2015 video watch:
https://www.youtube.com/watch?v=fHNmRkzxHWs
one of examples the presenter gave in his Cpp conference video.
Know that there is a definitely better algorithm than O(n^2), but also, need to know what the ideas are to beat the naive solution.

Dec. 11, 2015
Need to work on a small test case, therefore, the algorithm can be easily recalled, and ideas of algorithms can be demoed clearly in the example. Go to find my favorite string, substring. (January 5, 2015, read the wiki page, https://en.wikipedia.org/wiki/Boyer %E2 %80 %93Moore _string _search _algorithm, read 'The bad character rule' and 'The good suffix rule')

4. longest palindromic string

5. Look up standard string function implementation, quick review and learn:

January 5, 2016
Read the Java code on the following website:
http://algs4.cs.princeton.edu/53substring/BoyerMoore.java.html

Write a C # version, and check in github, and see if it will help to memorize the algorithm.

Read the webpage: (well written! now Julia knows two rules: bad character rule, the good suffix rule)
https://en.wikipedia.org/wiki/Boyer %E2 %80 %93Moore _string _search _algorithm

http://www.cs.tufts.edu/comp/150GEN/classpages/BoyerMoore.html

1.3.28 10 communication tips from bible (2015-08-23 15:15)

August 23, 2015

Try to improve my communication skills. Here are 10 favorite tips from bible teaching.

1. Listen without interrupting

2. Speak without Accusing

3. Give without Sparing

4. Pray without Ceasing

5. Answer without Arguing
6. Share without pretending
7. Enjoy without Complaint
8. Trust without Wavering
9. Forgive without Punishing
10. Promise without Forgetting

1.3.29 Reading time for algorithm, web programming (2015-08-29 17:08)

August 29, 2015

□□□□□□□□□□, □□, □□, □□□□□□□□□□. □□□□□, □□□□□□□□□, □□□□, □□□□□□□□.

http://justjavac.iteye.com/blog/2091899

http://www.sitepoint.com/store/jquery-novice-to-ninja-new-kicks-and-tricks/

http://eng.wealthfront.com/2014/09/from-wall-street-to-silicon-valley.html

http://blog.csdn.net/xudli/article/category/1357583

Sept. 4, 2015

http://blog.csdn.net/sunbaigui/article/details/6335138

1.3.30 Leetcode: Recover binary search tree (2015-08-31 20:32)

August 31, 2015

□□□□□, □□□□□□□□□□□□□□, □□□□, □□□□□□□□□□□□□□, □□□□□□□□□.

read blogs:

http://www.lifeincode.net/programming/leetcode-recover-binary-search-tree-java/
http://www.cnblogs.com/AnnieKim/archive/2013/06/15/MorrisTraversal.html

C# implementations:

, C# , , .

https://github.com/jianminchen/Leetcode_C-/blob/master/99RecoverBinarySearchTre
e.cs

, , .

https://github.com/jianminchen/Leetcode_C-/blob/master/99RecoverBinarySearchTre
eB.cs

, , , , . . , , .

, , .

work on extracting small functions, and then, understand the algorithm better.
No place for a bug.

https://github.com/jianminchen/Leetcode_C-/blob/master/99RecoveryBinarySearchTr
ee_C.cs

,

,

,

.

1.4 September

1.4.1 Morris post order traversal algorithm (2015-09-06 00:14)

Sept. 5, 2015

□

□□□□□□□□□

,

□□□

□□

.

□□□□□

,

□□

□

,

□□□□□□□□□

. □□□□□□□, □□□□.

□□□□□□□□□□-¡A

,

4

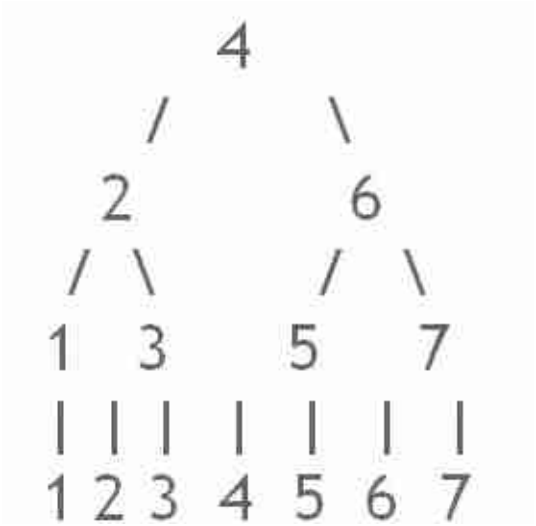
/ \

2 6

/ \ / \

1 3 5 7

easy way to travel is to remember the order of its position in the horizontal way



□□

,

□□

Morris

□□□□

□

,

□□□□□

,

□□□□□

□□

C #

□

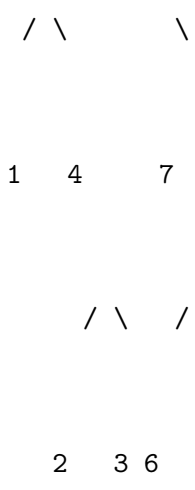
□□□□□

,

9

/ \

5 8



,

,

1,

2,

3, 4, 5

```
 6,

 7, 8, 9,

[MorrisOrder_PostOrder.jpg]

 1 2 3 4 5 6 7 8 9

dummy node, with left child is the root node.
```

Morris post order traversal

blogs to read:

http://www.cnblogs.com/AnnieKim/archive/2013/06/15/MorrisTraversal.html

C # implementation:

https://github.com/jianminchen/MorrisOrder/blob/master/MorrisPostO rder.cs

Morris order in order traversal

https://github.com/jianminchen/MorrisInOrderTraverse/blob/master/Program .cs

1.4.2 Draw a circle algorithm **(2015-09-10 22:11)**

August 18, 2015

Interesting problem – draw a circle,

blogs to read:

1. http://petercai.com/the-microsoft-sde-interview/
2. http://www.startuplessonslearned.com/2008/11/abcdefs-of-conduct ing-technical.html
3. https://en.wikipedia.org/wiki/Midpoint _circle _algorithm

C # code:

https://github.com/jianminchen/AlgorithmsProblems/blob/master/DrawACircle.cs

1.4.3 Backtracking algorithm: rat in maze **(2015-09-10 22:21)**

Sept. 10, 2015

Study again the back tracking algorithm using recursive solution, rat in maze, a classical problem. Made a few of mistakes through the practice, one is how to use two dimension array, another one is that "not all return path returns value", not so confident that "return false" at the end of function.

□□□□□□□□□□□, Rat in Maze, □□□□□□! □□□□□□□, □□□□□□□□□□□□□, □□□□□□□□, □□□□□. □□□□□□□□□, □□□□□, □□□□□□□□□□□□□□. □□□□□□□□, □□□□□□□. □□□□□□□□, □□□□□□□□; □□, □□, "return false" □□□□□□□□□□□, □□□□□. □□, □□□□□□□, □□□□□□□□; □□□□□□□, □□□□□□□, □□□□□□□.

Here are my favorite blogs about this problem:

1. http://www.geeksforgeeks.org/backttracking-set-2-rat-in-a-maze/
2. http://algorithms.tutorialhorizon.com/backtracking-rat-in-a-maze-puzzle/
3. https://www.cs.bu.edu/teaching/alg/maze/

Julia’s C # pratice:

https://github.com/jianminchen/AlgorithmsPractice/blob/master/RatInAMaz e _BackTracking.cs

And then, try to find more discussion about this problem, came cross blogs to challenge my analysis skills. □□□Google, □□□□□□□□□□□□, □□□□, □□□□□□, □□□□□□□□□□□□□□□.

http://blogs.msdn.com/b/mattwar/archive/2005/02/03/366498.aspx

http://blogs.msdn.com/b/mattwar/archive/2005/02/11/371274.aspx

More code to read and then play:

http://www.evercrest.com/ext/CheeseAppropriator.cs

and C # practice:

https://github.com/jianminchen/AlgorithmsPractice/blob/master/MousingAro und.cs

January 10, 2016
Review the algorithm

1.4.4 Leetcode 106: construct binary tree from inorder and post order traversal (2015-09-13 00:29)

Sept. 13, 2015

Spent more than a few hours to work on the leetcode problem, and my favorite blogs about this problems:

1. http://siddontang.gitbooks.io/leetcode-solution/content/tree/constru ct _binary _tree.html

2. http://blog.csdn.net/linhuanmars/article/details/24390157

After reading the above reference 1, Julia spent first few hours to write the C # implementation:

https://github.com/jianminchen/Leetcode _C-/blob/master/106ConstructuBTreeFromInorderPostOrderTraversal.cs

In her coding practice of function build(...), she spent over 20 minutes to figure out the coding task: the code to partition the inorder traversal into two partitions, first is left subtree, second is right subtree. And then, post order traversal also can be partitioned into two intervals, first one is for left subtree, and then, second one is for right subtree.

She took more than 10-15 minutes to understand the solution. That is too long for real problem solving. Figure out that only job is to find the root node, and then, its left child and right child is also the root node of left subtree/ or right subtree. The recursive call, actually two of them, can help to do the task.

So, she needs to cut down the practice time, and make sure the calculation is correct, easy to tell/ maintainable code/ testable code. So, she decided to practice it using class Range instead of two arguements start/ end integers. Hopefully, this practice will enhance her memory about partition the array into two parts, one is dividing in mid point, another one is by length of first interval - left subtree.

```
public class Range {
public int start, end;
...

}
```

Also, she likes to enhance her memory about this solution, so, she writes second implementation using C #, and see if the code can pass online judge. Most important, she tried to use different solution, to improve, challenge

herself. Write the code without any mistake first time, in less than 10 minutes based on previous one.

https://github.com/jianminchen/Leetcode _C-/blob/master/106ConstructBTreeFromInOrderPostOrderTraversal _B.cs

Julia likes to train herself using leetcode questions, and biggest problem is to cut down the time to write a solution. She tries to cut down time from hours to 10-30 minutes.

After the code writing, she thought about more about great ideas out there, she should not miss. So, she reads the second reference, and like the most about the analysis:

"???? Construct Binary Tree from Preorder and Inorder Traversal ??--????????????????-?Z????????????????A????????????????????????????????????-H???????????????? ".????????????????-????????-

Julia ??????????, ??????????, ??????????, ??????, ?10-30 ??, ??????????. ????, ??, ??????????, ???????; ???, ???????????????, ??????, ??????????, ??????????.

??, ??, ??????????, ??????????, ????????.

???, ????????; ?????????, ?????????10-15??????-çn???. ????????, ?????.

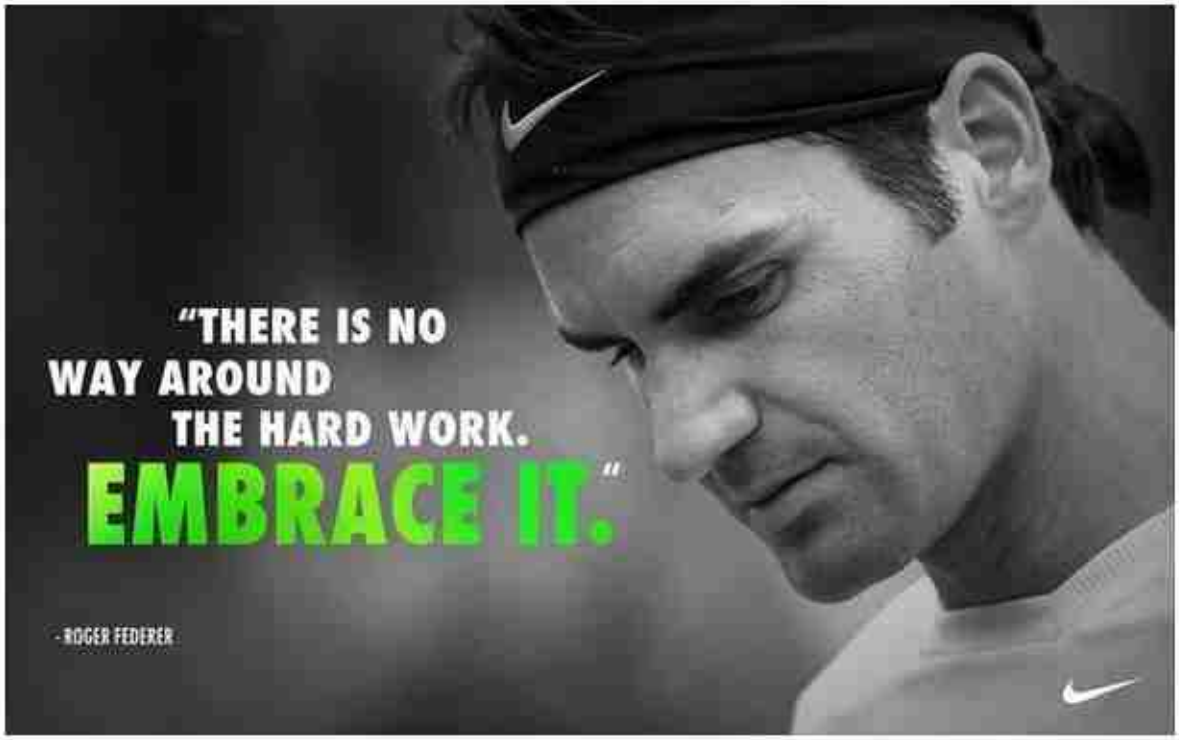
1.4.5 Java Script, C# learning - reading / videos material (2015-09-21 23:02)

September 21, 2015

First, share my favorite two verses to encourage myself to work harder, and continue to pursuit efforts in JavaScript/ C # / CSS/ other technologies learning in my career. I should have more organized, I just copied all my posts from my facebook and documented in a blog in Google+. Review them, and get more reading / study done continuously while practicing Leetcode questions.


Andy Murray
@andy_murray
Follow
WHY DO I SUCCEED?
I AM WILLING TO DO THE THINGS YOU ARE NOT
I WILL FIGHT AGAINST THE ODDS
I WILL SACRIFICE
I AM NOT SHACKLED BY FEAR, INSECURITY OR DOUBT
I AM MOTIVATED BY ACCOMPLISHMENT, NOT PRIDE
IF I FALL -- I WILL GET UP
IF I AM BEATEN -- I WILL RETURN
I WILL NEVER STOP GETTING BETTER
I WILL NEVER GIVE UP -- EVER
THAT IS WHY I SUCCEED
RETWEETS
2,464
FAVORITES
3,150
3:16 PM - 3 Apr 2014

Share the verse I like as well. There is no way around the hard work. Embrace it. Repeat again, there is no way around the hard work. Embrace it. That is easy to remember.



Roger Federer, Fans from all over the world

RT de Roger
Nike Tennis @NikeTennis 19 min

Lessons From The Court 2013: @RogerFederer Share yours and tag it #CourtLessons

2014□□□□□, □□□□□JavaScript□□□. □□□□JavaScript, 6□□□□, □□□□□□□□□□, □□□□□□□, □□□□□. □□□□□□, □□□□□□, □□□□□.

From the beginning of 2014, I spent over 12 months to learn JavaScript; Try to explore all I can learn in JavaScript, expose myself to all the best teachers, reading /videos in the world. Learn how smart people are in this technology world, JavaScript area. First time, I did slow down everything to learn a functional programming language.

After 6 - 8 months, I found out that I start to generate ideas to rewrite the Javascript wrote before.

So, document my learning experience and help myself to move forward when I have time.

June 1, 2015

Share the video. I read chinese blog in wechat, and then, catch up this one in English.

CHM Revolutionaries: " How Google Works" Eric Schmidt & Jonathan Rosenberg

https://www.youtube.com/attribution _link?a=s1oxvH9-h4E &u= %2Fwatch %3Fv %3D3tNpYpcU5s4 %26feature %3Dshare

March 15, 2015

So relax and also learn something on Sunday morning. Ajax and async.

https://sec.ch9.ms/ch9/03b0/3ac5c264-043f-4ffd-b2ee-422f39ba03b0/IntroTojQu eryM06 _mid.mp4

February 22, 2015
Spent half hour on CSS preprocessor. Great topic, and think about benefits to use it.

http://channel9.msdn.com/Series/Adding-Style-with-CSS/06?wt.mc _id=EntriesInArea

Enjoyed one hour video about CSS transition and transformation. Sunday is such great time
to learn something, and then, go out to play sports in Vancouver.

http://channel9.msdn.com/Series/Adding-Style-with-CSS/05

Being a web developer, it is fun to learn something called transition, transformation; Great video.

http://channel9.msdn.com/Series/Adding-Style-with-CSS/05?fb _action _ids=823960460986167 &fb _action _types=og.likes &fb _source=other _multiline &action _object _map= %5B771052266276726 %5D &action _type _map= %5B %22og.likes %22 %5D &action _ref _map= %5B %5D

February 9, 2015

Cooking time - learn something.

http://channel9.msdn.com/Series/Introduction-to-jQuery/02?wt.mc _id=player

February 7, 2015

Will have good time this weekend. Try to watch series video about single page application.

http://channel9.msdn.com/Series/Single-Page-Applications-with-jQuery- or-AngularJS/02?wt.mc _id=EntriesInArea

February 6, 2015

My favorite tip learned today. Learned a few tips, peek definition is also my favorite.

http://channel9.msdn.com/Series/vstips/lazycodesnippets?wt.mc _id=player

February 5, 2015

My Thursday night study time, Java Script is a good choice.

http://channel9.msdn.com/Shows/Defrag-Tools/Defrag-Tools-37-JavaScript-Par t-1?wt.mc _id=EntriesInArea

February 1, 2015

My favorite video this Sunday. Learn LINQ, good framework. Save time to develop.

http://www.youtube.com/watch?v=tGr6xm9nUm4 &feature=share &fb _ref=share

January 31, 2015

Saturday morning video. Nice to learn a few things, portable library is one.

Programming in C # (6 of 8) Splitting Assemblies, WINMD, Diagnostics and Instrumentation

http://youtu.be/ypdnMpR5jZ8

January 26, 2015

Watch the video and learn something new.

Programming in C # (1/8) OOP, Managaged Languages and C #

January 26, 2015

Share the video I like to watch. To test java script is new thing to learn.

http://www.youtube.com/watch?v=yRMsxyGsez4 &list=PLfXiENmg6yyUpIVY9XVOkbdmBPx6PUm9 _ &feature=share &index=3 &fb _ref=share

Netflix: Stop Making Excuses and Start Testing Your JavaScript!

http://goo.gl/QBWjvA

January 25, 2015
Sunday morning video, relax me and then learn something about Javascript.

Netflix JavaScript Talks - Async JavaScript with Reactive Extensions

http://www.youtube.com/attribution _link?a=LtVczHAKqbM &u= %2Fwatch %3Fv %3DXRYN2xt11Ek %26feature %3Dshare &fb _ref=share

January 23, 2015

Study time, good presentation!

Netflix JavaScript Talks - Version 7: The Evolution of JavaScript

http://www.youtube.com/watch?v=DqMFX91ToLw &feature=share &fb _ref=share

January 22, 2015

So easy to find an advance topic about java script through youtube, and then enjoy the video.

Lo-Dash and JavaScript Performance Optimizations

http://www.youtube.com/watch?v=cD9utLH3QOk &feature=share &fb _ref=share

January 21, 2015

Douglas Crockford: Advanced JavaScript

https://www.youtube.com/watch?v=DwYPG6vreJg

January 15, 2015

Share the video I watched tonight. Good video

The Definitive Guide to Object-Oriented JavaScript

http://www.youtube.com/watch?v=PMfcsYzj-9M &feature=share &fb _ref=share

January 1, 2015

Enjoy new year day off in Vancouver. Relax, read java script definitive guide, read again and again 20 pages from 163-180, and then, learn something from the video. My new year resolution is to become a stronger java script programmer in 2015.

What the heck is the event loop anyway? JSConf EU 2014

http://www.youtube.com/attribution _link?a=Cqtja3KchL0 &u= %2Fwatch %3Fv %3D8aGhZQkoFbQ %26feature %3Dshare &fb _ref=share

Dec. 30, 2015

Such a good video for me to watch, and then learn the scope chains and closures.

JavaScript Scope Chains and Closures

http://www.youtube.com/watch?v=zRZNb4GDOPI &feature=share &fb _ref=share

Dec. 27, 2015

Share the video. The CRAP rules - contrast, repeat, alignment, proximity, very great idea for website design.

HTML and CSS Web Development - Footer and Design Theory (CRAP rules) - Part 13

http://www.youtube.com/attribution _link?a=JeHOVs6tcZw &u= %2Fwatch %3Fv %3D9IpeHgl _g90 %26feature %3Dshare &fb _ref=share

Dec. 14, 2014

o2 Creation Patterns Creational Patterns

https://www.youtube.com/watch?v=LjIWRvOL7aM &list=PLrzrNeNx3kNHsaPfrpPo0AlW-MhJE6gOA &index=10

Dec. 27, 2014

Make me smile, the short video, nice teaching.

HTML and CSS Web Development - How to apply transition effects - Part 14

http://www.youtube.com/watch?v=WRADgQVSWG4 &feature=share &fb _ref=share

Dec. 25, 2015

My favorite half hour to watch this Java script video in Christmas day morning.

Exam Application Programming Tutorial JavaScript Quiz Online Test

http://www.youtube.com/attribution _link?a=UiB8WtJBt5A &u= %2Fwatch %3Fv %3Dd _UuOVhuCF8 %26feature %3Dshare &fb _ref=share

Dec. 24, 2015

Just watch, and then, learn something interesting. Good teaching

Image Cycle JavaScript HTML CSS Web Programming Tutorial

http://www.youtube.com/watch?v=3QuSKsr47f0 &feature=share &list=PL00952AC35D0A4701 &index=3 &fb _ref=share

Java Script Tutorial Videos (Object oriented)

https://www.youtube.com/playlist?list=PL00952AC35D0A4701

Dec. 23, 2014

My favorite video this Christmas holiday. Java script, is a fun, challenging language to learn.

Douglas Crockford: Advanced JavaScript

http://www.youtube.com/watch?v=DwYPG6vreJg &feature=share &fb _ref=share

Dec. 21, 2014

Spent time to read the book, this weekend. Just start to read creation pattern. Good book for a programmer to start to love the java script.

Book: JavaScript Patterns

http://www.amazon.ca/JavaScript-Patterns-Stoyan-Stefanov/dp/0596806752

Dec. 15, 2014

Google I/O 2014 - Keynote

https://www.youtube.com/watch?v=wtLJPvx7-ys

Dec. 13, 2014

Such a good teaching. Excellent for a morning hour.

AngularJS Fundamentals In 60-ish Minutes

http://www.youtube.com/attribution _link?a=-U2QuDFQH48 &u= %2Fwatch %3Fv %3Di9MHigUZKEM %26feature %3Dshare &fb _ref=share

http://www.youtube.com/watch?v=DwYPG6vreJg &feature=share &fb _ref=share

Dec. 13, 2015

Like the presentation. Programming can be easy with great training. Still learn java script, having fun.

Structuring JavaScript with the Revealing Module Pattern - Video

http://www.youtube.com/watch?v=V _X86VyrEjc &sns=fb

My favorite learning time is insomnia, reading does help. Learning helps more.

Using the JQuery each() Function - video

http://www.youtube.com/watch?v=Fekw8FwJcOk &sns=fb

Nov. 29, 2014

Being a big fan of the book last 2 months, motivated to rewrite the code make more readable. Coaching is such great experience, easy to read, short book, and training myself with a coach will make life much easy.

The Art of Readable Code (Theory in Practice) - Book

http://www.amazon.com/The-Readable-Code-Theory-Practice/dp/0596802293

November 29, 2014

Great video to watch, still on the way to learn Jquery, Java script.

Fundamentals for Great jQuery Development

http://www.youtube.com/watch?v=YcylSiDoOio &feature=share &fb _ref=share

Nov. 26, 2014

Share the video I watched tonight.

Douglas Crockford: The Better Parts - JSConfUY 2014

http://www.youtube.com/attribution _link?a=v2YJsPYGvZM &u= %2Fwatch %3Fv %3Dbo36MrBfTk4 %26feature %3Dshare &fb _ref=share

November 17, 2014

Learning java is so much fun. Like the video.

Google I/O 2009 - Big Modular Java with Guide

http://www.youtube.com/attribution _link?a=GCVjQsKVR5E &u= %2Fwatch %3Fv %3DhBVJbzAagfs %26feature %3Dshare &fb _ref=share

Nov. 12, 2014

So many questions and answers. Like those answers.

Google I/O 2011: Programming Well with Others: Social Skills for Geeks

http://www.youtube.com/attribution _link?a=NHYbdKud8r0 &u= %2Fwatch %3Fv %3Dq-7l8cnpI4k %26feature %3Dshare &fb _ref=share

Nov. 11, 2014

So many good ideas. Never thought about those ideas before. Be a manager seems to be like a teacher.

Google I/O 2010 - The joys of engineering leadership

http://www.youtube.com/attribution _link?a=7r4t5Axk4pc &u= %2Fwatch %3Fv %3DskD1fjxSRog %26feature %3Dshare &fb _ref=share

Watch the video. Two experts share principles, good points: do not burn the bridge, face time, visible/invisible work etc.

Google I/O 2012 - The Art of Organizational Manipulation

http://www.youtube.com/attribution _link?a=-05Nv51UgJU &u= %2Fwatch %3Fv %3DOTCuYzAw31Y %26feature %3Dshare &fb _ref=share

Nov. 11,2014

Videos I like

Fluent 2013: Nicholas Zakas, "A " Thank You" Can Change Your life"

http://www.youtube.com/attribution _link?a=P6SF3zGOp4U &u= %2Fwatch %3Fv %3D78eexsReL9M %26feature %3Dshare &fb _ref=share

Nov. 9, 2014

Really a struggle to read a few pages Java script definitive guide; and then, find the companion of reading, piano music, players. Learning a programming language is much easy to compare to piano.

Music video

http://www.youtube.com/attribution _link?a=pxI27DzEb5I &u= %2Fwatch %3Fv %3DAkCpX2Pnj0w %26feature %3Dshare &fb _ref=share

Nov. 2, 2014

Reading "definitive Java Script guide" is still slow, 1 hour a time, still on the page 38 out of 1098 pages. Learning a programming language needs me to be more patient.

Oct. 19, 2014

Good idea to learn a few things to write a language. It took 10 days for some one to design Java script.

JavaScript: Choose Your Own Adverture!

http://www.youtube.com/attribution _link?a=uklLhqKvXYA &u= %2Fwatch %3Fv %3D _ _8eIX0QFXU %26feature %3Dshare %26list %3DPLndbWGuLoHeZVgHEBjGGtCqQaXpZJfv51 %26index %3D4 &fb _ref=share

Learn java script programming language, warm up before I read through java script definitive guide book.

Context in JavaScript - 1/4 - Purpose and Problems with JavaScript's "This"

http://www.youtube.com/attribution _link?a=387pTjKZkJg &u= %2Fwatch %3Fv %3Dsu-SdgebJCE %26feature %3Dshare &fb _ref=share

Oct. 16, 2014

Good video to spend 1 hour.

JavaScript Essentials - video
http://www.youtube.com/attribution _link?a=9hzmx-0p57Q &u= %2Fwatch %3Fv %3D03EQu _2K2cs %26feature %3Dshare &fb _ref=share

Oct. 13, 2014

Good topic. Testable design, is a good start to think about testing.

GTAC 2010: Flexible Design? Testable Design? You Don't Have To Choose!

http://www.youtube.com/watch?v=K3q _y8H1ZTo &feature=share &fb _ref=share

Oct. 13, 2014

Watch the video, website administration work is fun and challenge sometimes.
https://www.youtube.com/watch?v=5 _-YukDEDBE &list=UUtXKDgv1AVoG88PLl8nGXmw &index=427

Thanksgiving day is so much fun, and with a lot of talks and laughing and singing around the house.

Game On: 16 Design Patterns for User Engagement - Google Tech Talk

http://www.youtube.com/watch?v=YZCX-izCYMk &list=UUtXKDgv1AVoG88PLl8nGXmw &feature=share &index=21 &fb _ref=share

Oct. 13, 2014

My Canadian thanksgiving morning time video - great time to relax and enjoy some talks

NYC Tech Talk Seris: Javascript Testing at Google Scale

http://www.youtube.com/watch?v=draY63DasmQ

Oct. 12, 2014

watch the video. Testability is a good topic.

Design Tech Talk Series Presents: OO Design for Testability - Google Tech Talk

http://www.youtube.com/watch?v=acjvKJiOvXw &feature=share &fb _ref=share

Oct. 11, 2014

Like this talk. Angular JS, 1 hour learning. Good time to spend at home to learn something other people working on.

Google I/O 2013 - Design Decisions in AngularJS

http://www.youtube.com/watch?v=HCR7i5F5L8c &feature=share &fb _ref=share

Like this talk. It is good for Saturday morning, organizing house while watching the video.

Google I/O 2009 - The Myth of the Genius Programmer

http://www.youtube.com/watch?v=0SARbwvhupQ &list=PLC0C39A4F2D9580F4 &feature=share &fb _ref=share

Oct. 10, 2015

Learning to love Javascript. Who else is learning?

Google I/O 2011: Learning to Love JavaScript

http://www.youtube.com/attribution _link?a=mI4cQMSJT4g &u= %2Fwatch %3Fv %3DseX7jYI96GE %26feature %3Dshare &fb _ref=share

Start to watch more videos at home, learn from experts. Java script, will be my favorite language.

Speed Up Your JavaScript

http://www.youtube.com/attribution _link?a=EqQZpKBee6c &u= %2Fwatch %3Fv %3DmHtdZgou0qU %26feature %3Dshare &fb _ref=share

Oct. 7, 2014

Cannot believe that today I read this book first time. What a great book.

Functional Programming in Java

http://shop.oreilly.com/product/9781937785468.do

Oct. 5, 2014

Spent Sunday morning to watch the video. Learn Java Script language knowledge from the expert. Learning is so much fun, depth of the knowledge is out of my imagination.

Crockford on JavaScript - Chapter 2: And Then There Was JavaScript

http://www.youtube.com/attribution _link?a=AmQCwXmbO60 &u= %2Fwatch %3Fv %3DRO1Wnu-xKoY %26feature %3Dshare &fb _ref=share

Oct. 2, 2014

Enjoy the talks in the evening, Model this 1+2*3, good question.

"The Clean Code Talks – Inheritance, Polymorphism, & Testing"

https://www.youtube.com/watch?v=4F72VULWFvc

Sept. 28, 2014

Learning Java Script is so much fun. Got lost in the video first, and then, read the book to try to understand, and then, play with the sample code.

Douglas Crockford: Advanced JavaScript

http://www.youtube.com/attribution _link?a=m9mNGZ0AGLM &u= %2Fwatch %3Fv %3DDwYPG6vreJg %26feature %3Dshare &fb _ref=share

Sept. 26, 2014

So great to have a video to watch, and then, try to make sense from my own experience.

How to Write Clean, Testable Code

http://www.youtube.com/watch?v=XcT4yYu _TTs &feature=share &fb _ref=share

Sept. 25, 2014

□□□□□, □□□□□□□□. □□□□□□□□, □□□□□□□, □□□□□!

Douglas Crockford: An Inconvenient API - The Theory of the DOM

http://www.youtube.com/watch?v=Y2Y0U-2qJMs &feature=share &fb _ref=share

Like the book "testable java script"; and then, have chance to know the expert, and then, watch videos to learn Java script.

Douglas Crockford: The JavaScript Programming Language

http://www.youtube.com/attribution _link?a=6fbrVqm1kJ0 &u= %2Fwatch %3Fv %3Dv2ifWcnQs6M %26feature %3Dshare &fb _ref=share

Sept. 21, 2014

So great to spend 3-4 hours in the Sunday morning to go through a book, for 30-40 pages, comparing to read all weChat blogs. Be able to read through the book and read other people's source code, really feels good.

Book: High Performance JavaScript

1.4.6 Leetcode 109: convert sorted list to a binary search tree (2015-09-22 21:33)

Sept. 22, 2015

109 Convert sorted list to binary search tree (No. 109)

8/25/2015

Read the following blogs:

http://articles.leetcode.com/2010/11/convert-sorted-list-to-balanced-binary .html

C #, bottom up, time O(n), space O(log n) solution - best solution:

https://github.com/jianminchen/Leetcode _C-/blob/master/109ConvertSortedListToBinarySearchTreeB.cs

C #, top down, time O(n^2), space O(long n) solution - naive solution:

worked on code 2 times, first time, the calculation is kind of messy, then, worked on Leetcode question 108, get the idea to make it more simple; tips like len/2 only shows once, afterwards, use m instead. Just need to improve coding, think to make it more abstract, simple.

https://github.com/jianminchen/Leetcode _C-/blob/master/109ConvertSortedListToBinarySearchTreeC.cs

9/21/2015

Review the best solution, and then, totally forgot the bottom up solution idea. So, update the code with more comment.

Need to review more about bottom up/ top down solution in tree problems. Get more experience on bottom-up solution, read some articles about it.

9/22/2015

Go over one example to build some muscle memory about this bottom up, O(1) solution to find the root node in subtree function.

Sorted List:

1->2->3->4->5->6->7,

How to convert the above sorted list to a binary search tree?

Thought process:

1.

First, get length of the list, which is 7 in the above list;

2.

Secondly, define a recursive function called

constructBST(ref TreeNode head, int start, int end)

the above function definition, 3 arguments:

1.

First one is the reference of head node of sorted list,

2.

Start index of the list,

3. End index of the list

In the function, first define the base case:

head is null, or start<end, return null

start==end, return head

then, call the recursive function for left subtree:

head node is the same, start is the same, but end is len/2-1;

great tip comes in, the head node should move in the function, so that

root node can be accessed in the list using O(1), instead of starting from very beginning.

One more statement:

head = head.next;

TreeNode root = head; // return this node as tree root node

Root.left = left subtree root node

Root.right = right subtree recursive function

constructBST(ref head.next, mid+1, end)

The tips to remember in the design, the recursive function should return the root node of the tree; secondly, input argument of linked list should use reference, and also head node moves in the recursive function, so it is O(1) to find the root node.

Just cannot believe that only call .next function once in the recursive function! How to argue that the move is only once? Therefore, the total calls of .next should be length of list. Total recursive function calls is n, length of list.

Debate why the recursive function has to return root node, and set up root node, connect its left/ right subtree root node.

Debate why the linked list head node is moving.

□□□□□□□□

,

□□□□□□□□□□□□

,

□□□□

?

□□□□□□□□□□

,

□□□□□□□□□□

,

□□□□□□

,

□□□□□

.

□□□□

,

□□□□□□□□

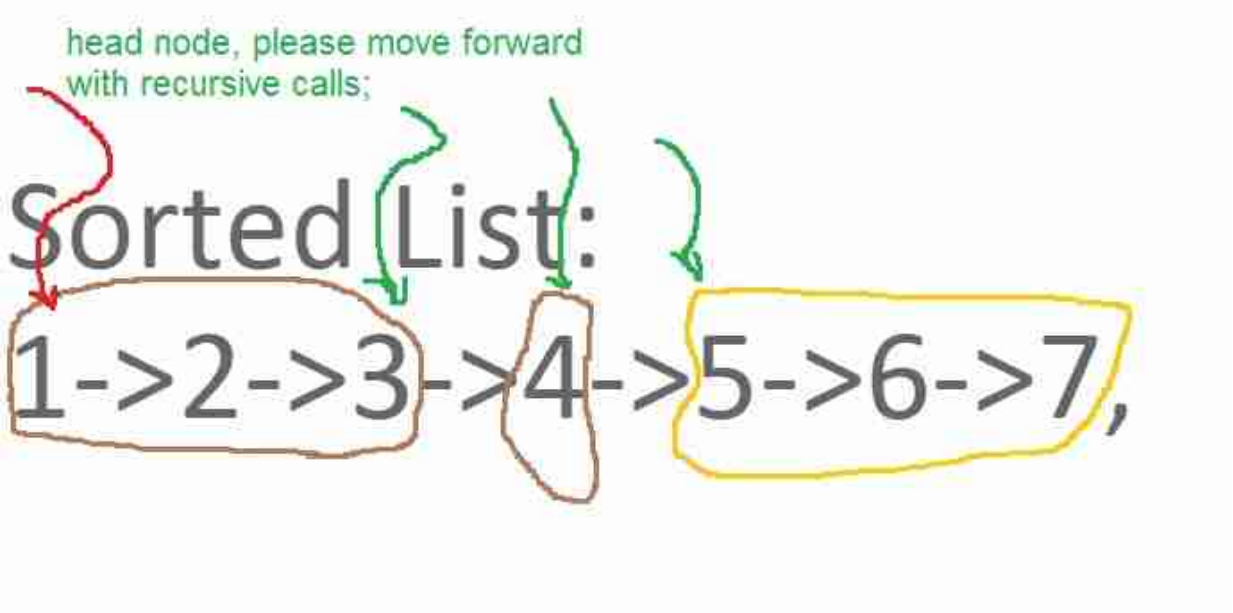
;

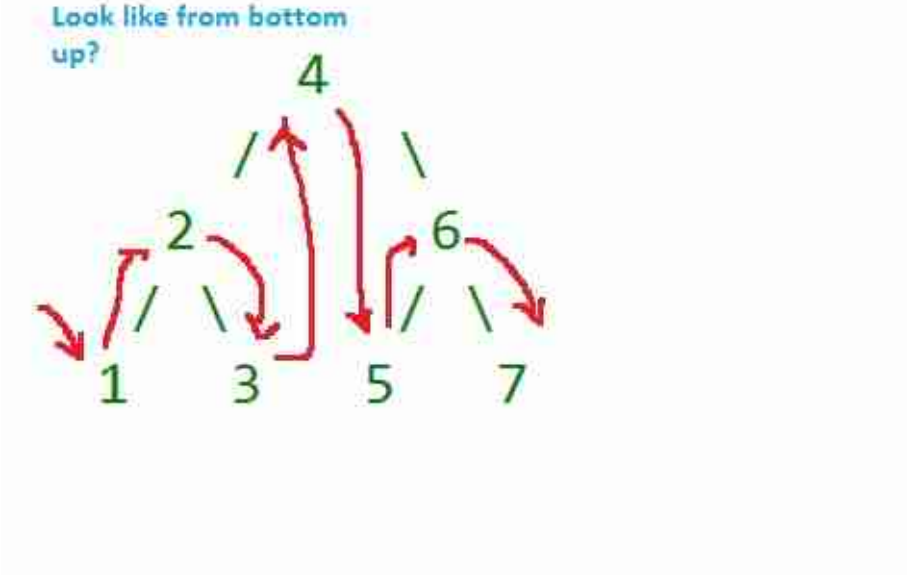
□□□□□

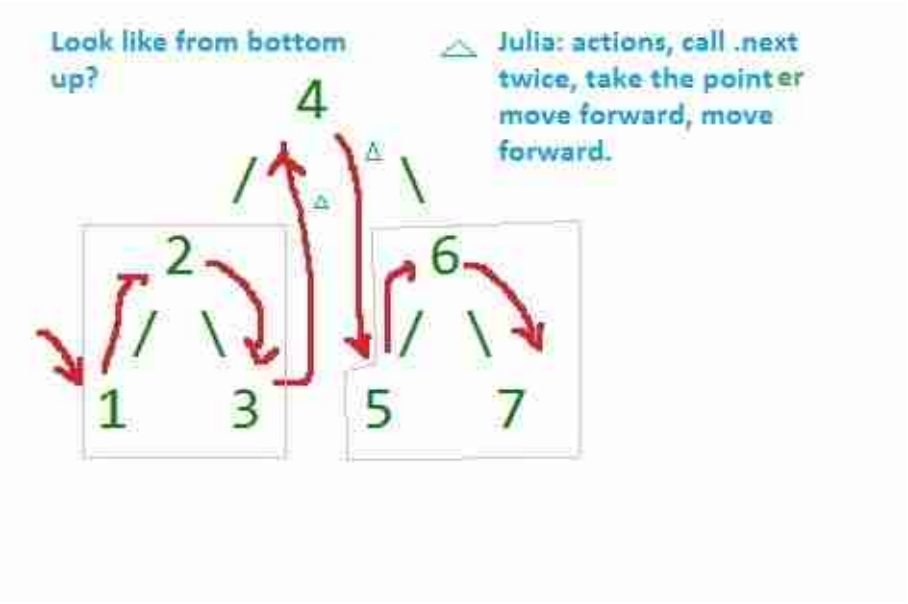
,

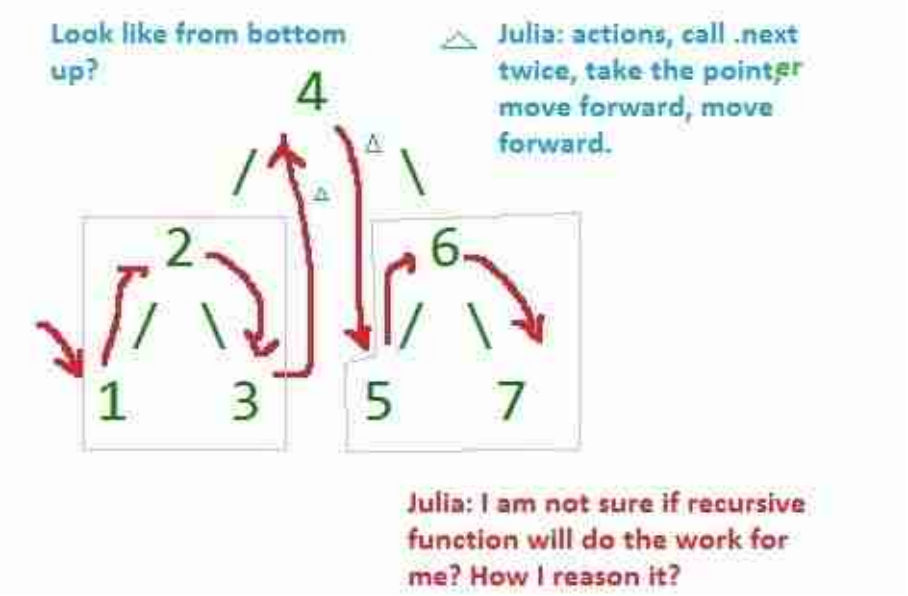
□□□□

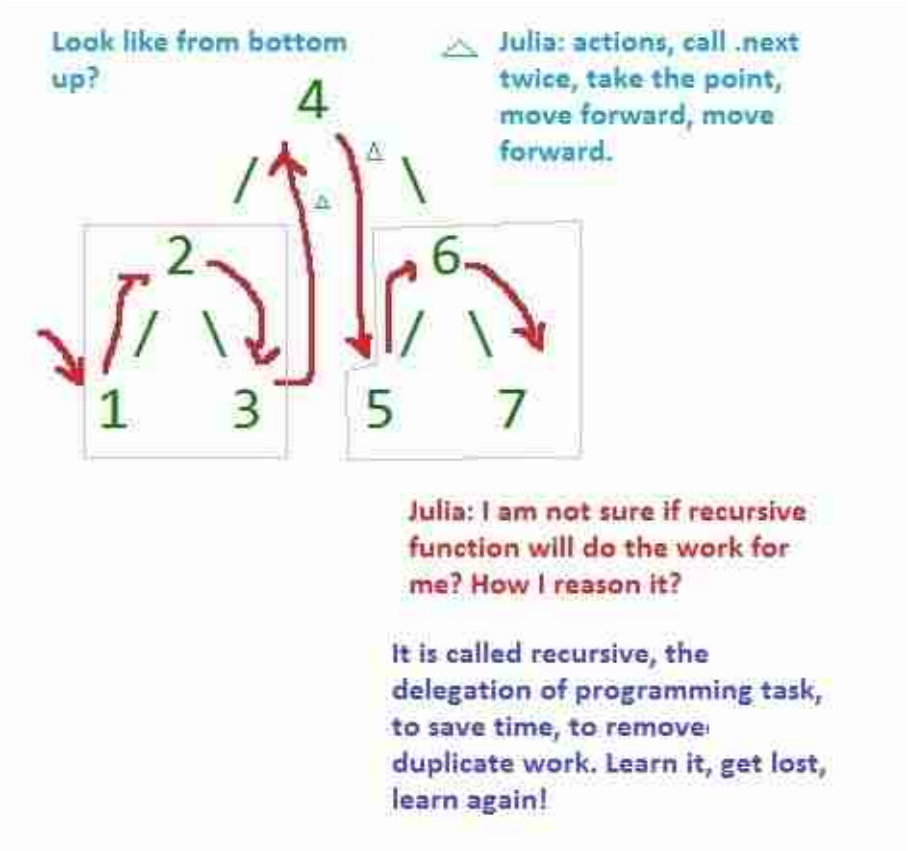
.


head node, please move forward
with recursive calls;
Sorted List:
1->2->3->4->5->6->7,


Look like from bottom
up?
4
2
6
1
3
5
7


Look like from bottom up?
Julia: actions, call .next twice, take the pointer move forward, move forward.
4
2
6
1
3
5
7


Look like from bottom up?
Julia: actions, call .next twice, take the pointer move forward, move forward.
4
2
6
1
3
5
7
Julia: I am not sure if recursive function will do the work for me? How I reason it?



The main point to understand the best solution using O(ln N) space, the left subtree has to be built first, and then, head node can be retrieved as .next method call, root node can be set up, and then, left subtree can be built. So, left subtree, then right subtree, then the tree with root node. Bottom up.

Whereas sorted array to BST, the root node can be find right away, and then, tree can be set up top down. Julia is still confusing this top down / bottom up difference. :-) read more blogs.

Blogs:

1. use global variable to remember the head node of linked list, great idea:

http://www.jiuzhang.com/solutions/convert-sorted-list-to-binary-search-tree/

2. another implementation using two pointers. Try it using C # later.

https://siddontang.gitbooks.io/leetcode-solution/content/tree/convert _sorted _listarray _to _binary _search _tree.html

3. Java implementation, teach me how to use reference in Java or wrapper class. Good point!

http://rleetcode.blogspot.ca/2014/02/convert-sorted-list-to-binary-search.h tml

4. Time and space complexity analysis - think about it - very clear analysis

http://blog.unieagle.net/2012/12/14/leetcode %E9 %A2 %98 %E7 %9B %AE %EF %BC %9Aconvert-sorted-list-to-binary-search-tree/

5. Three implementation discussions

http://www.cnblogs.com/feiling/p/3267917.html

6. Great explanation - bottom up / top down, and in order traversal/ post order traversal

https://leetcodenotes.wordpress.com/2013/11/23/convert-sorted-list-to-bi nary-search-tree/

Implement the above 6 blogs using C #, and then, share the blog. After 1 month, check again and see if I can come out the bottom up solution in 5 minutes. If yes, stop; otherwise review again.

Dec. 24, 2015

Review the solution again.

How to recall the solution step by step?

1. Get list length <- travel the list once

2. Design the recursive function with a list, start and end two position; use the position of start/end to boundary case checking; also the function returns the root node of the tree.

3. Build left subtree

4. Find the middle of list as root <- how to get there?

Cannot traverse the list again to half length if you want optimized solution, since it is in recursive function, O(nlogn)

Every recursive call the list pointer will traverse once

5. Connect root node to left child

6. Move list pointer to next one

7. Build right subtree

connect root node to right child as well.

Recap the importance of function design:

1. list with a pointer moving, starting from head

2. start, end position of linked list

3. in the function build a tree

4. return root node of the tree

1.4.7 Leetcode: Convert sorted array to binary search tree (2015-09-25 20:54)

09/25/2015

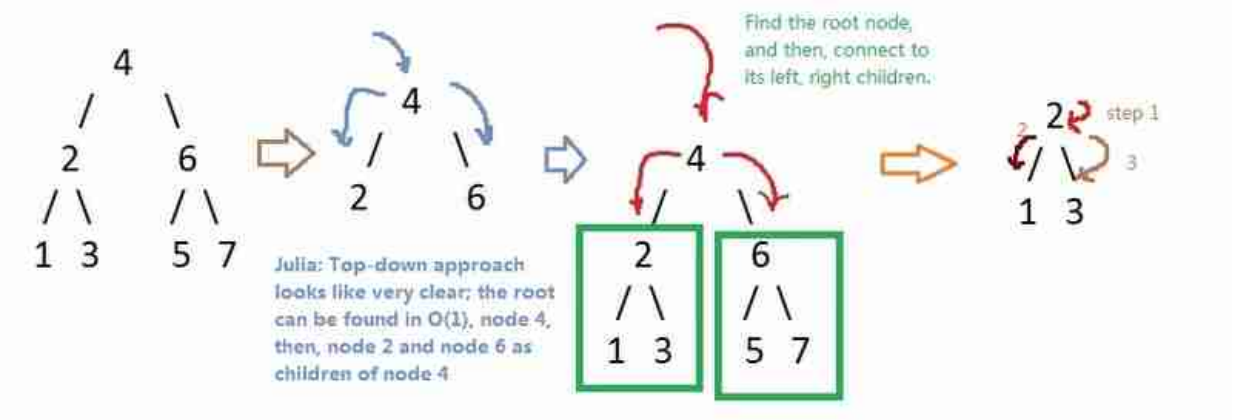
First practice on 08/25/2015

108 Convert sorted array to binary search tree (No. 108)

Julia's C # implementation: top-down approach

https://github.com/jianminchen/Leetcode _C-/blob/master/108ConvertSortedArrayToBinarySearchTree.cs

9/25/2015 Try to add some fun memory about top-down approach for binary search tree construction, then, present some diagrams.



my task is to make the coding practice more fun, and some funny part with naive drawing. Maybe, Julia could use the drawing to bring some comic style to serious coding challenges.

Julia's favorite blogs about this leetcode question:

1.

https://gist.github.com/Jeffwan/8982109

Read this blog (1) and like the analysis, first time see the comment: "Conquer" above the return statement:

return node; Learn Divide and Conquer in this simple illustration.

2. http://blog.csdn.net/linhuanmars/article/details/23904883

Because this blog's popularity, it is one of my favorites. One thing I like to discuss is about the base case, how this base case is making sense, and then mimimum/ no redundancy. Let Julia argue about it.

Here is the code in the above blog:

1. private
   TreeNode helper(
   int
   [] num,
   int
   l,
   int
   r)
2. {
3. if
   (l>r)
4. return
   null
   ;

```
5.  int
    m = (l+r)/
    2
    ;
6.  TreeNode root = new
    TreeNode(num[m]);
7.  root.left = helper(num,l,m- 1
    );
8.  root.right = helper(num,m+ 1
    ,r);
9.  return
    root;
10. }
```

Have some base case debates, and make this algorithm more entertainment:

why base case is:

if(l>r) return null,

if l==r, the above function will perform:

root = new TreeNode(num[l]), and also,

set up

root.left = RecursiveFunctionCall() -> go to base case, return null,

root.right is the same as root.left,

and then return root;

how come it is not:

if(l==r) return new TreeNode(num[l])

then, need to add checking before the helper call:
if(l<=m-1)

root.left = helper(num, l, m-1)

if(m+1<=r)

root.right = helper(num, m+1, r)

So, better to leave this if conditional checking to base case; at least there are 2 if checking, anyone of them implies l<=r, so duplicate checking.

3.

http://www.acmerblog.com/leetcode-solution-convert-sorted-array-to-b inary-search-tree-6247.html

julia's comment:

the base case has redundant line:

Java code:

if(start == end) return

new

TreeNode(num[start]);

Here is the Java code:

```
 public class Solution {

 11     public static TreeNode sortedArrayToBST(int[] num) {

 12         if(num == null || num.length == 0) return null;

 13         return sortedArrayToBST(num, 0, num.length-1);

 14     }

 15

 16     public static TreeNode sortedArrayToBST(int num[],int start, int end){

 17         if(start > end ) return null;

 18         if(start == end) return new TreeNode(num[start]);

 19         int mid = start + (end-start)/2;

 20         TreeNode root = new TreeNode(num[mid]);

 21         root.left = sortedArrayToBST(num, start, mid-1);

 22         root.right = sortedArrayToBST(num, mid+1, end);

 23         return root;

 24     }

 25 }
```

192

4. http://pisxw.com/algorithm/Convert-Sorted-Array-to-Binary-Search-Tree.html

Remind me the BST, the left and right subtree difference is maximum 1.

5.

http://www.kunxi.org/notes/LeetCode/

Look into the leetcode solution later. Should be a great one for me to catch up some programming tips.

1.4.8 Study time - watch videos, and read articles (2015-09-25 23:04)

10/18/2015

REST+JSON API Design - Best Practices for Developers

https://www.youtube.com/watch?v=hdSrT4yjS1g

How To Design A Good API and Why it Matters (Julia's rating: A+)
https://www.youtube.com/watch?v=aAb7hSCtvGw

– more notes: How To Design A Good API and Why it Matters
Watched again on 10/21/2015, notes taken to share:
54:40/1:00:18
Use Appropriate Parameter and Return Types
. Favor interface types over classes for input
- provides flexibility, performance
. Use most specific possible input parameter type
- Moves error from runtime to compile time (Julia's comment: Great tip, reduce time on debugging, fix things in compiling time!)
. Don't use string if a better type exists
- Strings are cumbersome, error-prone, and slow
. Don't use floating point for monetary values
- Binary floating point causes inexact results!
- Use double (64 bits) ranther than float (32bits)
-Precision loss is real, performance loss negligible

watched again on 10/21/2015, notes take to share:
49:19/1:00:18
Don't violate the Principle of Least Astonishment

Read more on this article:
http://programmers.stackexchange.com/questions/187457/what-is-the-princip le-of-least-astonishment

– end of notes for

Josh Bloch, Lord of the APIs - A Brief, Opinionated History of the API
https://www.youtube.com/watch?v=ege-kub1qtk

https://www.youtube.com/watch?v=4YxnxmQS41s

9/27/2015
SQL injection Myths & Fallacies: Best practices of defense

https://www.youtube.com/watch?v=o4dJ7hdA8fs

Quickly go over the book "SQL antipatterns" in short future.

9/24/2015

Effective Java - Still Effective After All These Years

https://www.youtube.com/watch?v=V1vQf4qyMXg

Julia's favorite time: first 15 minutes with two great coding examples; watch again later.

9/25/2015

Back to my life without writing code, enjoy the videos to catch up things about internet and technology, my favorite topic in the video: google CEO comment about competition, company vs. government, what the differences, How CEO thinks.

Eric Schmidt & Jared Cohen: The impact of Internet and Technology

https://www.youtube.com/watch?v=OwJTCHxjcjQ

Get curious about .net framework - work on .NET framework and enjoy benefit last 5 years; should spend 10 hours to watch the .NET framework videos. Will spend more time when I take vacation from Oct. 2 - Oct. 16, 2015

https://www.youtube.com/watch?v=ywN0vTFJNcw &list=PLnrS3jsiKkdKcChakQPJi1XdKyDt _2jPa

9/28/2015
first time using search in github, and seach Sudoku, and find some code to read:

https://github.com/dartist/sudoku _solver/blob/master/benchmark/sudoku.cs

http://norvig.com/sudoku.html

http://aspadvice.com/blogs/rbirkby/archive/2007/08/23/Silverlight-Sudoku-wi th-LINQ.aspx

1.4.9 Object oriented principles - S.O.L.I.D. (2015-09-26 00:10)

August 28, 2015

□□□□□□□□□□□□□□□Ž□□□□□□□□□□

,

□□□ƙ□□

,

□□□

!

□□□□□□□-ĥ□

,

□□□□□□□□□□

.

□□□

,

□□□

,

□□□□□□

,

□□□□□

.

□□□□□□

,

□□□□□□□□□

?

□□□□□□ω□□

,

□□□□□□□□□□□□□□□

.

SOLID principles:

· S – Single-responsiblity principle

· O – Open-closed principle

· L – Liskov substitution principle

· I – Interface segregation principle

· D – Dependency Inversion Principle

□□□□□□□□□□□

:

https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-orie nted-design

Challenge myself, learn some OO design. Here are my focus from Jan. 2014 - Sept. 2015, Java Script -> Leetcode -> .NET framework, OO design. Julia learns to change her focus and become a good learning "wild animal" in IT industry.

Nov. 11, 2015

Need to work on the code seriously, start to apply those five principles: SOLID.

Here is the video studying on this remembrance day of Canada holiday,

Applying S.O.L.I.D. Principles in .NET/C #

https://www.youtube.com/watch?v=Whhi1C2PpaA

Spent 60 minutes to watch this video:

Bob Martin SOLID Principles of Object Oriented and Agile Design
https://www.youtube.com/watch?v=TMuno5RZNeE

video 1:12/1:23

Introduction to Dependency injection

https://msdn.microsoft.com/library/hh323705(v=vs.100).aspx

Nov. 12, 2015

book chapter from uncle bob:

http://objectmentor.com/resources/articles/Principles _and _Patterns.pdf

Lecture notes to read

http://www.it.uu.se/edu/course/homepage/ood/ht12/overview/principles/Lecture 4.pdf

Videos (Julia's favorite)

https://www.youtube.com/watch?v=5lqB0AwrMxg &index=13 &list=PL8m4NUhTQU48oiGCSgCP1FiJEcg _xJzyQ

http://code.tutsplus.com/series/the-solid-principles–cms-634

http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

1.4.10 ASP.NET framework (2015-09-27 22:03)

August 28, 2015

□□□□

,

□□□□□□

ASP

□

ASP.NET

□□

, .NET framework

□□□□□□□

?

□□□-ɾκ□□

, □□□□□□ . □

http://www.indiabix.com/technical/dotnet/asp-dot-net/

Read the webpage: (Sept. 2, 2015)

https://www.microsoft.com/learning/en-ca/exam-70-486.aspx

Watch MVC videos if I have a few of hours.

https://www.microsoftvirtualacademy.com/en-US/training-courses/developing-a sp-net-mvc-4-web-applications-jump-start-8239

1.5 October

1.5.1 Good API video watching and notes taken (2015-10-22 00:04)

Oct. 22, 2015

Spent time to look into the note through the video, and then study more later:

How To Design A Good API and Why it Matters (Julia's rating: A+)

https://www.youtube.com/watch?v=aAb7hSCtvGw

1. **Watched again on 10/21/2015, notes taken to share**:

54:40/1:00:18

Use Appropriate Parameter and Return Types

. Favor interface types over classes for input

- provides flexibility, performance

. Use most specific possible input parameter type

- Moves error from runtime to compile time (Julia's comment: Great tip, reduce time on debugging, fix things in compiling time!)

. Don't use string if a better type exists

- Strings are cumbersome, error-prone, and slow

. Don't use floating point for monetary values

- Binary floating point causes inexact results!

- Use double (64 bits) rather than float (32bits)

-Precision loss is real, performance loss negligible

2. **watched again on 10/21/2015, notes taken to share**:

49:19/1:00:18

Don’t violate the Principle of Least Astonishment

Read more on this article:
http://programmers.stackexchange.com/questions/187457/what-is-the-princip le-of-least-astonishment

3. watched again on 10/21/2015, notes taken to share:

42:31/1:00:18

Subclass only where it Makes Sense

. Subclassing implies substitutability (Liskov)

– Subclass only when is-a relationship exists

– Otherwise, use composition

. Public classes should not subclass other public classes for ease of implementation

Bad:

Properties extends Hashtable

Stack extends Vector

Good:

Set extends Collection

1.5.2 The clean code talks - unit testing (2015-10-25 12:50)

10/25/2015

Watch the google talk "The clean code talks - unit testing", write down notes and then learn better for next time.

video link:
https://www.youtube.com/watch?v=wEhu57pih5w

Video time:
02:02 / 32:07
Hard to test code

Most People say:
Make things private
Using final keyword
Long methods

Real issues: (interview questions)
Mixing new with logic
Work in constructor
Global state
Singleton
Static Methods (procedure programming) (leaf method vs way-up method)
Deep inheritance (Divorce myself from inheritance at run time.)
Too many conditionals ()

Procedure code - Seam - Polymorphism - Julia's comment: look into it, read more about it. 10/25/2015

Video time:
14:35/32:07
Progression of Testing:

1. Scenario / Large Tests
Test whole applicatio by prentending to be a user
slow / Flaky / Mostly happy paths

2. Functional / Medium Tests
External dependencies simulated
Test class interaction

3. Unit / Small Tests
Focus on application logic
Very Fast / No I/O / No Need for a debugger

4. Think of it more as a continuum than discrete

5. All tests levels important
Not all levels have the same probability of a bug
Because smaller tests cover less need more of them

6. Testability more important as you get more focused

Video time:
17:05 /32:07
Different kinds of Tests
Execution time :
unit test -> Functional Tests -> Scenario

Test individual ->
Test collections of classes as subsytems ->

Test the whole system pretending to be a user

Video time:
27:06/32:07
How do you write hard to test code?

You mix object creation code with business logic
This will assure that a test case never can construct a graph of objects different from production. Hence nothing can be tested in isolation.

Unit test class -
Seam - friendly - try to understand the dependency injection
class depends on a lot of other classes, mocking, seam, learned through hacking,

class:
object graph
construction & lookup
business logic

27:36/32:07
Take Away
Unit tests are a preferred way of testing

Unit tests require seperation of instantiation of object graph from business logic under test

Questions:
1. Global variable - make test unpredictable, the order of test matters and so on and so forth.

Do not clean up after yourself, another test will fail;
Counter argument: framework than DI framework.

Monkey-patching - evil on the global state - dependency injection

C++ / Java / frameworks - dependency injection difference

And read the slideshow using the following link:

http://www.slideshare.net/avalanche123/clean-code-5609451

http://misko.hevery.com/code-reviewers-guide/

http://misko.hevery.com/presentations/

http://misko.hevery.com/2008/10/21/dependency-injection-myth-reference-pass ing/

1.5.3 Study notes - "The clean code talks - 'Global State and Singletons' " (2015-10-25 14:07)

10/25/2015

Google talks:

The clean code talks - "Global State and Singletons"

https://www.youtube.com/watch?v=-FRm3VPhseI

Global state:

Julia tries to understand global state in this talk:

Run same thing a multiple time, will get different test results;

5:40/54:08
A. Multiple Executions can product different results
1. Flakiness
2. Order of tests matters
3. Can not run test in parallel

B. Unbounded location of state

C. Hidden Global State in JVM
1. System.currentTime()
2. new Date()
3. Math.random()

D. Testing above code is hard
1. Therefore inject in doubles

8:49/54:08
Singleton: Good vs Bad

1. Application Global vs. JVM Global
2. Usually we have one application per JVM
A. Hence we incorreclty assume that:
Application Global state = JVM Global

3. Each test is a different configuration of a portion of our application

19:01/54:08
Deceptive API
1. API that lies about what it needs
2. Spooky action at a distance

Julia's comment, the talk is interesting and worth time to watch again later.

1.5.4 reading time on Sunday - housing issue, health issue, problem solving (2015-10-25 15:51)

10/25/2015

Spent time to watch the video on this Sunday. Learn how billionaire educate their kids, what philosophy they hold, I do not have money but I definitely like the wisdom. Enjoy the perfect English and forget my concerns about money, Vancouver and all other things.

https://www.youtube.com/watch?v=aSL-iIskEFU

Enjoy reading, and also find interesting things to read.

http://www.aol.com/article/2015/10/21/a-23-year-old-google-employee-lives-in -a-truck-in-the-companys/21251909/?ncid=txtlnkusaolp00001363

http://frominsidethebox.com/

10/28/2015

Melinda Gates - talk about puzzle, nonprofit, problem solving in general

Personal story to help parents to do rental business - so encouraging me to work hard as well - start from small business.

https://www.youtube.com/watch?v=i9Ifb3EGl9Q

Melinda French Gates: What nonprofits can learn from Coca-Cola

https://www.youtube.com/watch?v=GlUS6KE67Vs

10/31/2015

https://engineering.pinterest.com/blog/discover-pinterest-search-and-discov ery

read linkedin profile about site reliability engineer work experience

https://www.linkedin.com/profile/view?id=AAEAAAAxOf0BLtv8Mfz6NXAfym0v5a dk35-sCV4 &authType=name &authToken=EWTR &trk=prof-sb-browse _map-name

Building Pinterest's Mobile Apps ()
http://www.infoq.com/presentations/pinterest-app

https://engineering.pinterest.com/tech-talks

How we've scaled dropbox
https://www.youtube.com/watch?v=PE4gwstWhmc

Discover Pinterest: Mobile Engineering and Design
https://www.youtube.com/watch?v=wzRyTmBxs7Y #t=63m24s

1:10/1:35
Nags - Nag placement, Home view placement, Pin Tutorial Card

Further reading Nov. 3, 2015

Responsive Web Design: Relying Too Much on Screen Size

http://www.lukew.com/ff/entry.asp?1816

http://www.lukew.com/presos/

Nov. 6, 2015
Discover Pinterest: Search and Discovery
https://www.youtube.com/watch?v=f _JeAEBxpUs

Read the article

Venture Beat article.

http://venturebeat.com/2015/05/28/pinterest-improves-related-pins-with-d eep-learning-plans-product-recommendations-using-object-recognition/

Nov. 7, 2015

Google I/O 2012 - SQL vs NoSQL: Battle of the Backends

https://www.youtube.com/watch?v=rRoy6I4gKWU

Julia spent time to learn NoSQL, and got some basic ideas using NoSQL.

12:46pm
Google I/O 2012 - Building Mobile App Engine Backends for Android, IOS and the Web
https://www.youtube.com/watch?v=NU _wNR _UUn4

Google I/O 2011: HTML5 versus Android: Apps or Web for Mobile Development?

https://www.youtube.com/watch?v=4f2Zky _YyyQ

O'Reilly Webcast: Building Mobile HTML5 Apps in Hours, Not Days

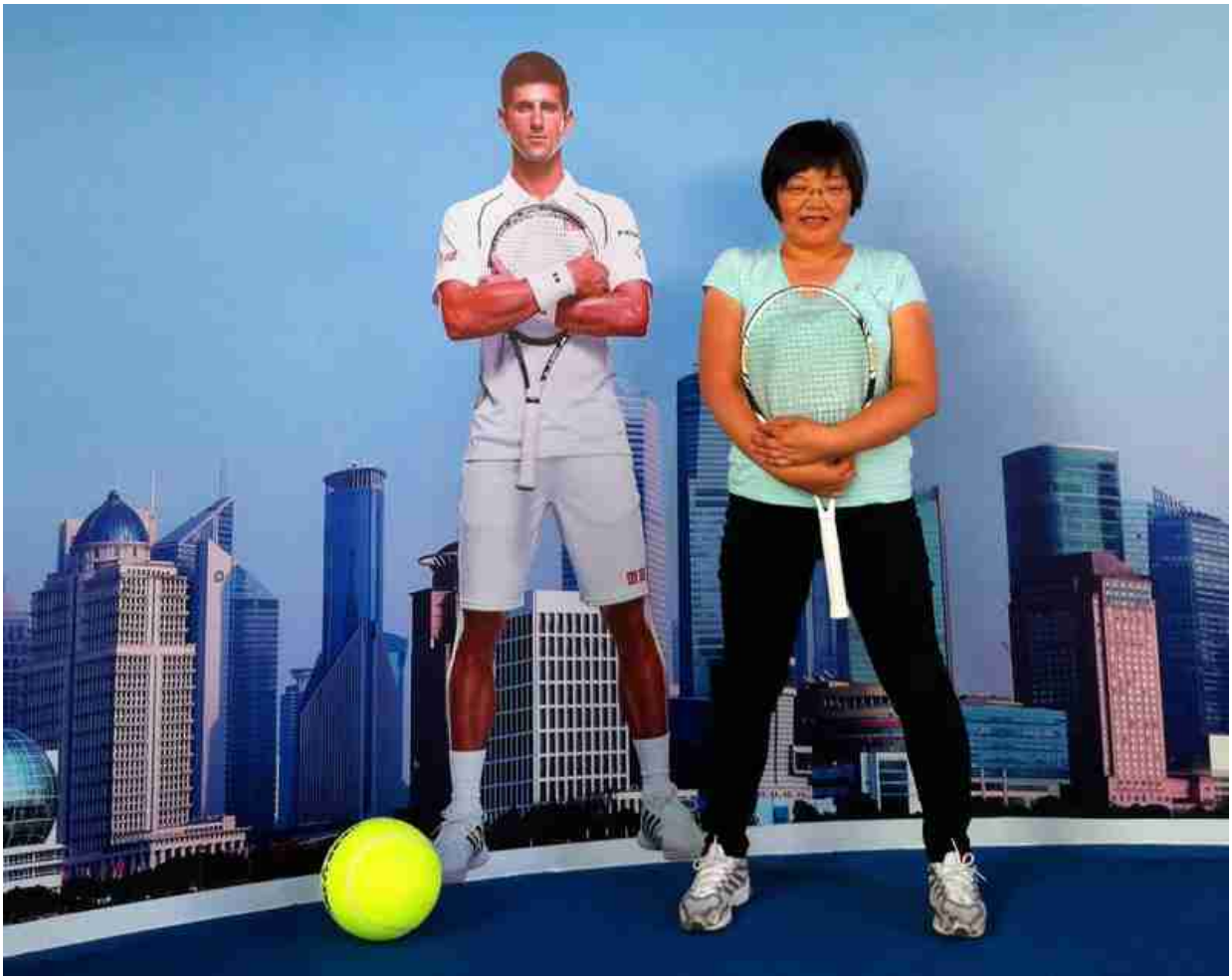
https://www.youtube.com/watch?v=G0UC-C2Y7ME

1.5.5 Back to leetcode code algorithms (2015-10-26 22:44)

10/26/2015

Spent 2 weeks vacation in China from Oct. 2 - Oct. 16.

Enjoyed the tennis practice in the city of yichun, jiangxi province, and also the Rolex tennis Master, 2015 in Shanghai.



Start to read blogs about leetcode algorithms in the evening, back to training and practice. My favorite blogs to read tonight.

http://fisherlei.blogspot.ca/2015/10/leetcode-product-of-array-except-s elf.html

http://fisherlei.blogspot.ca/2015/10/leetcode-different-ways-to-add.html

http://fisherlei.blogspot.ca/2015/10/leetcode-valid-anagram-solution.html

Read 6 more blogs from the above blog.

1.5.6 Leetcode question 241: Different ways to add parentheses (2015-10-28 21:40)

10/28/2015

Read the blog,

http://fisherlei.blogspot.ca/2015/10/leetcode-different-ways-to-add.html

http://blog.csdn.net/guanzhongshan/article/details/48086695

and write C # code, compile it, build it, pass online judge, check in Github, and then, write a new code using memorization.

1. First step, write down C # code, compile ok.

https://github.com/jianminchen/Leetcode _C-/blob/master/241DifferentWaysToAddParentheses.cs

2. Read the most popular blog about this leetcode question through Google,
http://blog.csdn.net/sbitswc/article/details/48546421

and then, take Java code as a sample, write C # version of implementation.

https://github.com/jianminchen/Leetcode _C-/blob/master/241DifferentWaysToAddParentheses _B.cs

read the leetcode string algorithms blog (A plus):
http://blog.csdn.net/sbitswc/article/details/20429853

The leetcode question 241 is just a medium question in difficulties.

Dec. 17, 2015
Review the solution, and then, need to write down the solution - a script to help solve the problem:
1. 1+2*3
-> read a substring from leftmost to convert it to number, for the example, read '1', integer 1, and then, visit '+' stop,
-> now, let us use recursive function call to get the first part of substring before '+', and get second part of substring after '+', and then, two lists, each one of list1 will operate '+' with each one of list2, add into return list.

Not convincing, try again:
use binary tree to model this problem:
http://juliachencoding.blogspot.ca/2015/12/oo-principle-solid-open-close- principle.html

So, get the root node - operator, and then, deal with left child, right child, and root node does the evaluate function.

1.6 November

1.6.1 testing, automation and testing patterns (2015-11-08 12:51)

Nov. 8, 2015

Watch the video:

Automated Testing Patterns and Smells

https://www.youtube.com/watch?v=Pq6LHFM4JvE

Julia is a developer in the city of Vancouver last 5 years, she enjoyed the understanding how to minimize time 70 % by following one simple principle: DRY - Do not repeat yourself.

For instance, to calculate something, the same code repeats twice; Just a simple example, Julia does not like to see the code repetition in the C # code, so she gives out a try to use base class, and then, merge to one function applying to the base class. But, all other repetitions are everywhere, so she cuts other places similar to the one, 2 to 1. So, do a math, the code needs change for cases 2 x 2 x 2, 3 place duplication, becoming 1 x 1 x 1; Done, reduce the 7/8 work to work with design, unit test, and bugs related to design. (similar ideas seen video, watched on Nov. 13, 2015 - https://www.youtube.com/watch?v=bxQK7hcVyGs &list=PLD4GnSXHpkQ2 _eKG5fKzszXs5hYcEsft7)

Julia tries to contribute the ideas to show how important it is to following DRY principle, and then, reduce test cases, improve code quality, and then, shorten time to work on a project.

Back to the video, Julia likes to write down the notes from the video, she does not have time to read 600 pages book, but she knows the tip: get a good video, google talk, and then, invest 2-3 hours:

1. video time 16:07/59.33
What is a " Test smell"
1. duplicate code, hard coded values, etc.

A set pf symptoms of an underlying problem in test code. Like duplicate code, breaking DRY principle.

In the talk, using changing diaper as an example, how to know when to change diaper? When diaper stinks.

code smells - visible problems in test code
behavior smells - test behaving badly
project smells - testing-related problems visible to a project manager

2. video time 19:27 / 59:33
What’s a "Test Pattern"?

A "test pattern" is a recurring solution to a test automation problem
- E.g. A "Mock Object" solves the problem of verifying the behavior of an object that should delegate behavior to other objects

Test Patterns occur at many levels:
- Test automation strategy pattern
recorded test vs scripted test
- Test design Patterns
implicit setup vs delegated setup
- Test coding patterns
assertion method, creation method
- Language-specific test coding idioms
expected exception test, constructor test

3. video time 20:28/59:33
Common code smells

- conditional test logic
- hard to test code
- obscure test
- test code duplication
- test logic in production

4. Example to explain, best part
video time 22:16/59:33

1. Better assertion
2. Hard-wired test data - what is the relationship between those data? What is math? Do we need math? 30, why it is not 1, 19.99, why it is not 2, 69.96, why it is not 3? 6 months later, what is business logic?
Hard-wired test data - will lead to fragile test

3. Introduce custom assert

Avoid any conditional test logic in the test method - make less test code.
Problem of conditional test logic is not sure what you are testing. <- Julia likes this argument
use customer assert, make a compaction of the code.

video time 28:58 / 59:33
4. Automated fixture teardown

tear down logic is a smell, such as, to separate business logic with presence logic

explicitly create deleteAllTestObjects instead of those complicated logic checking - nested try methods.

5. video time 32:07 / 59:33

Obscure test - irrelevant information

create address
create .. vs createAnnoymousCustomer vs createAnnoymounsInvoice
create customer createAnnoymounsInvoice

Hide detail in creation method - encapsulation, what is level of info - different levels info stays in one function - create more functions - each function much easy to be understandable, easy for unit test, variable scope less than a few lines

6. Test coverage, rapid test writing

Work outside in, create functions not existed yet, type test what you like to look

- Julia's comment:
If the code she wrote, it seems that the task is not showed in Leetcode questions, that means the code is not abstracted, not using object-oriented programming, cannot be unit test easily, people cannot read it without too much memorization, one function does all kinds of tasks; results? just too much time wasted for development, maintenance, testing, and it is bad behavior, not a pro.

7. video time
44:08/ 59:38
Reducing Erratic Tests - Shared Fixture

Build new Shared Fixture for each run
- Avoids Unrepeatable Tests
- When:
>> Lazy Setup
>> Setup Decorator
>> SuiteFixture Setup

Fragile Tests:
Causes:
Interface Sensitivity
- Every time you can change the SUT (S- , U- , T- tear down), tests won't compile or start failing
- You need to modify lots of tests to get things "Green" again
- Greatly increases the cost of maintaining the system

Behavior Sensitivity
- Behavior of the SUT changes but it should not affect test outcome
- Caused by being dependent on too much of the SUT's behavior.

Data sensitivity

Context sensitivity
- something outside the SUT changes
e.g. system time/date, contents of another application

46:19/ 59:33

Hard to test code
. code can be hard to test for a number of reasons:
- too closely coupled to other software
- No interface provided to set state, observe state
- Only asynchronous interfaces provided

. Root cause is lack of design for testability
- comes naturally with Test-Driven development
- Must be retrofitted to legacy (test-less) software

. Temporary workaround is Test Hook
- Becomes test logic in production (code smell) not removed.

50:17/59:33
What *Does* it take to be successful?
Programming Experience
+ xUnit Experience
+ Testing Experience
+ Design for Testability
- Test smells
+ Test Automation Patterns
+ Fanatical Attention to Test Maintainability

read the website:
http://xunitpatterns.com/

Julia's comment:
Source code should be only written for machine to understand - 20 %
Source code should be readable for herself after 6 months / years - 20 %
Source code should be minimal - DRY principle - All other principle for easy to understand,
relax for herself - any time - 30 %
Source code should be fun to read, with documented unit test cases, log history etc. - ?

1.6.2 OO design principles - DRY - Don't repeat yourself principle (2015-11-14 15:34)

Nov. 14, 2015

It is so great to find something to work on this weekend of November. My favorite time to study the video:

The Don't repeat yourself princple, part 2

https://www.youtube.com/watch?v=wweNdRuM64g &list=PLD4GnSXHpkQ2 _eKG5fKzszXs5hYcEsft7 &index=2

Video time: 30:15/31:07
Summary
. Repetition breed errors and waste

. Abstract repetitive logic in code
. Related fundamentals:
. template method pattern
. command pattern
. dependency inversion principle
Recommended reading:
. The pragmatic programming: From Journeyman to Master
. 97 Things Every Programmer Should Know.

So, Julia found one book to read this weekend, 97 Things Every Programmer Should Know.

Further reading: (Nov. 18, 2015)
Template method pattern:
https://en.wikipedia.org/wiki/Template _method _pattern

1.6.3 book reading: 97 Things Every Programmer Should Know (I) (2015-11-15 13:05)

Nov. 15, 2015.

I like to tell how I get here to read this book - 97 things every programmer should know. It is a long story, but it can be described as this flow:

Julia works on leetcode questions (January 2015) ->
Challenged on August, 2015 about SOLID OO principles (Only know / remember S, basic decoupling, work on interface)
-> procedural code's concern, try to follow DRY principles, Oct. 2015
-> Learn SOLID priciples through videos in Nov., 2015:
-> Recommended reading: the book

Spent 3 hours to read the book on Sunday morning, take some notes about my favorite ideas:

1. Act with prudence - Seb Rose

"doing it right" vs "doing it quick"

Julia's comment: thinking about it later :-) Prudence means " be careful".

2. Apply Functional Programming Principles - Edward Garson

Julia's comment:
Look into the term: "high degree of referential transparency"
Questions from Julia, list a 3-5 principles of functional programming principles, most popular ones.

3. Ask, "What Would the User Do?" (You are not the user) - Giles Colborne
Julia's comment:
Look up the term: the false consensus bias - psychologists call it.

4. Automate your coding Standard - Filip van Laenen

Julia's comment: Look up the meaning of "antipatterns", things talked about are interesting: test coverage, coding standard. Read the article later.

5. Beauty is in simplicity - Jorn Olmheim

Julia's comment: Four things we strive for in our code:
Readability (1), maintainability (2), speed of development (3), the elusive quality of beauty (4)
Simple objects with a single responsibility, with simple, focused methods with descriptive names.
Desirable goal is to have short methods of 5-10 lines of code. (Julia likes this argument!)

6. Before you refactor - Rajith Attapattu

Julia's comment: great topic, and good to hear some advice.

7. Beware the Share - Udi Dahan

8. The Boy Scout Rule - Robert C. Martin (Uncle Bob)

Julia's favorite: Make it better: improve name of one variable, or split one function into two smaller functions. Or break a circular dependency, or add an interface to decouple policy from detail.

And more about team spirit, help one another and clean up after one another.

9. Check Your Code First before looking to blame others - Allan Kelly
Once you eliminate the impossible, whatever remains, no matter how improbable, must be the truth.
http://www.allankelly.net/

10. Choose Your tools with Care - Giovanni Asproni
existing tools - components, libraries, and frameworks
http://www.atlantec.ie/giovanni-asproni-to-be-added/

11. Code in the language of the Domain - Dan North
consulting service:
http://dannorth.net/about/

12. Code is Design - Ryan Brush
http://conferences.oreilly.com/strata/stratany2014/public/schedule/speaker/1 38579

13. Code layout matters - Steve Freeman

Julia is still working on the code layout; she thinks that the good layout helps, her favorite book: the art of readable code

15. Code Reviews - Mattias Karlsson
Code review - increase code quality and reduce defect rate.

used to be: Lead programmer does the review, architect reviews everything.

16 Coding with Reason - Yechiel Kimchi

http://programmer.97things.oreilly.com/wiki/index.php/Coding _with _Reason

Julia likes the idea -
1. Try to reason about software correctness by hand
2. Automated tools are preferable but not always possible
3. a middle path - reasoning semiformally about correctness

Julia likes this middle path, in detail,
To divide all the code under consideration into short sections - from a single line, such as a function call, to blocks of less than 10 lines - and argue about their correctness.

Julia likes to memorize those good advices, so just write it down word by word.
1. Avoid using goto statements, as they make remote sections highly interdependent.
2. Avoid using modifiable global variables, as they make all sections that use them dependent.
3. Each variable should have the smallest possible scope. For example, a local object can be declared right before its first usage.
4. Make objects immutable when relevant.
5. Make the code readable using spacing, both horizontal and vertical.
6. Make the code self-documenting by choosing descriptive name for objects, types, functions, etc.
7. If you need a nested section, make it a function. <- Julia's favorite: avoid nested section, add a function
8. Make your functions short and focus on a single task. **<- great advice, obey it.**

The old 24-line limit still applies. Although screen size and resolution have changed, nothing has changed in human cognition since 1960s. (**24 lines, Julia can do it!**)
9. Function should have few parameters (**four is a good upper bound**). This does not restrict the data communicated to functions: Grouping related parameters into a single object benefits from object invariants and saves reasoning, such as their coherence and consistency.

17. Convenience is Not an -ility - Gregor Hohpe

It is a great topic about API design. A good API follows a consistent level of abstraction, exhibits consistency and symmetry, and forms the vocabulary for an expressive language.

API design reading, found one through search:
http://lcsd05.cs.tamu.edu/slides/keynote.pdf

18. Deliberate practice

Leading Lean Software Development (Addison-Wesley Professional)

19. Domain-Specific Languages

https://about.me/mhunger

20 Do not afraid to break things - Mike Lewis

21. **Encapsulate Behavior, Not just State - Julia's favorite article**

http://www.researchgate.net/profile/Einar _Landre/publications

http://programmer.97things.oreilly.com/wiki/index.php/Encapsulate _Behavior, _not _Just _State

Examples:
It is easy to find a class with a single 3,000-line main method, or a class with only set and get method for its primitive attributes.
Analysis: do not understand the object-oriented thinking, do not take advantage of the power of objects as modeling constructs.

Given an example, 3 classes: Customer, Order, and Item.
A Customer object is the natural placeholder for the credit limit and credit validation rules.
An Order object knows about its associated Customer, and its addItem operation delegates the actual credit check by calling customer.ValidateCredit(Item.Price()).

Less experienced OO developer, design one objects: OrderManager or OrderService to wrap all business rules. In the design, Order, Customer, and Item are treated as little more than record types.
All logic is factored out of the classes and tied together in one large, procedural method with a lot of internal if-then-else constructs.

But, these methods are easily broken and almost are impossible to maintain. Because encapsulation is broken.

22. Hard work does not pay off - Olve Maudal

23. Know your limits
http://www.boost.org/doc/libs/1 _34 _0/people/greg _colvin.htm

24. A message to the Future - Linda Rising

Her website:
http://www.lindarising.org/

Julia really likes the short story written in the chapter. The problem solved is difficult, and the solution should be just as difficult for everyone (maybe even for themselves a few months after the code was written) to understand and maintain. But, the code should be easy to understand.

25. **Only the code tells the truth - Peter Sommerlad**

https://www.youtube.com/watch?v=XLsZkA77h8c
Notes from video:
Less code = More Software
Let Julia read those sentences and laugh about mistakes she made as well:
1. Complexity is one of the biggest problems with software if not THE biggest.
2. It is much easier to create a complicated "solution" than to really solve a problem.
3. Much software complexity is accidental not inherent to the problem solved.
4. It starts in the small, one statement at a time.
5. Architects and developers need to value Simplicity!
. Good Abstractions are the key, as are
. Managing Dependencies (Avoid global variables)
6. Software needs to be simpler to solve more complex problems.
7. Simple software requires work and skill but pays off in the long run.

Famous Quotes by Sir C.A.R. (Tony) Hoare

1. Inside every large problem, there is a small program trying to get out.
2. There are two ways of constructing a software design:
1. one way is to make it so simple that there are obviously no deficiencies, and
2. the other way is to make it so complicated that there are no obvious deficiencies.
The first method is far more difficult.

Good advice:

1. Strive for good names.
2. Structure your code with respect to cohesive functionality, which also naming.
3. Decouple your code to achieve orthogonality.
4. Write automated tests explaining the intended behavior and check the interfaces.
5. Refactor mercilessly when you learn how to code a simpler, better solution.
6. Make your code as simple as possible to read the understand.

- to be continued.

More reading about authors:

1. Enjoy reading the blog of one of authors:

To junior developer:

http://dearjunior.blogspot.ca/2014/11/make-implicit-concepts-explicit-i n-code.html

Note:

Eric Evans, the thought leader of Domain Driven Design.

http://dearjunior.blogspot.ca/2012/03/dry-and-false-negatives.html

1.6.4 The clean code talks (2015-11-18 10:05)

Nov. 18, 2015

Review the video about the testing, and try to get more from this lecture:

"The Clean Code Talks – Inheritance, Polymorphism, & Testing"

https://www.youtube.com/watch?v=4F72VULWFvc

action items:
1. put sample code in C #, and then, check in git;
2. write down key words, and easy for review and follow as rules.

Julia's sample code in C #:
C # code using conditional implementation, one class Node

https://github.com/jianminchen/OODesignPractice/blob/master/ConditionalVersio n.cs
https://github.com/jianminchen/OODesignPractice-1-2-3-B/blob/master/OODesign _CleanCodeTalk _Condition-alVersion.cs

better solution: Node, OpNode, ValueNode:
https://github.com/jianminchen/OODesignPractice-1-2-3-B/blob/master/OOD esign _CleanCodeTalk _OpNodeVal-ueNode.cs

optimal solution:
https://github.com/jianminchen/OODesignPractice-1-2-3-B/blob/master/OODe sign _CleanCodeTalk _OpNodeVal-ueNodeAndMore.cs
Perfect example to learn S.O.L.I.D. OO principles, open for extension, close for change. The above optimal solution does not have any if statement, and any new arithmetic operation just needs a new class, no need to touch existing code. Julia passes the learn Open/Close principle. Move on to next one, Liskov substitution principle!

Notes:
Premise - Most ifs can be replaced by polymorphism
Why?
Easy to read, test without ifs.
polymorphic systems are easier to maintain.

Julia: write down the sentence in the talk: supporting facts: one execution path, so easy to understand, test, and extend.

Use polymorphism
If an object should behave differently based on its state.
If you have to check the same conditions in multiple places.

- binding is not on compile time,

Use conditionals
Mainly to do comparisons of primitive objects: >,<,==, !=
There other uses, but today we focus on avoiding if

Do not return null in the method
To be if free
Never return a null, instead return a Null object, e.g. an empty list

Don't return error codes, instead throw an Exception (Run Time please!)

Rampant (wild, unchecked) subclassing
Polymorphism uses subclassing
Be careful about runaway subclassing (another talk focuses on that)
avoid pitfall: inheritance hierarchy - too complex

State based behavior

Replace conditionals with polymorphism
You have a conditional that chooses different behavior depending on the type of an object.

Move each leg of the conditional to an overriding method in a subclass.
Make the original method abstract.

Example:

```
double getSpeed() {
switch ( _type) {
case EUROPEAN:
return getBaseSpeed();
case AFRICAN:
return getBaseSpeed() - getLoadFactor() * _numberOfCoconuts;
case NORWEGIAN _BLUE:
return ( _isNailed)? 0 : getBaseSpeed( _voltage);
}
throw new RuntimeException ("Should be unreachable");
}
```

Suggestion to solve the problem: use subclasses
Isn’t there already a type system?

An Exercise:
Model this!
1 + 2 * 3

favorite interview question by the speaker:

Represent this as a tree

```
+
/ \
1 *
/ \
2 3
```

Node object to store the information

evaluate()
computes the result of an expression

Most of people come out solution like the following:
Using conditionals

```
class Node {
char operator;
double value;
Node left;
Node right;
double evaluate() {
```

```
switch(operator) {
case ' #': return value;
case '+': return left.evaluate() + right.evaluate();
case '*" return left.evaluate() * right.evaluate();
case ... // edit this for each new operator
}
}
}
```

Big problem on this:
graphic representation,
Node
op:char
value: double
left: Node
right:Node
——————
evaluate():double

Julia could not figure out the analysis here <- first time to memorize this analysis, and also try to learn reasoning

Analyzing attributes

	#	+	*
function	yes	yes	
value	yes		
left	yes	yes	
right	yes	yes	

Two different behaviors are fighting here, (see the above table),
if you are the operation node, then you need your left and right child; whereas value node, you just need value.

Either you need value or you need left and right, but you never need both.
One class have multiple tasks entangled, polymorphism is through if statement, not through polymorphism.

video time: 11:42/38:24
Let us break it up:
Node
——————
evaluate(): double
| |
ValueNode OpNode
value: double op: char
—————— left: Node
evaluate: double right: Node
——————-
evaluate(): double
As showing above, break Node into ValueNode and OpNode, so ValueNode does not have left and right child because

of no meaning over there.

Operations and values

```
abstract class Node {
abstract double evaluate();
}

class ValueNode extends Node {
double value;
double evaluate() {
return value;
}
}

class OpNode extends Node {
char operator;
Node left;
Node right;
double evaluate() {
switch(operator) {
case '+': return left.evaluate() + right.evaluate();
case '-': return left.evaluate() + right.evaluate();
case ... // edit this for each new operator
}
}
}
```

How to extend this? Every time you add a new operator, need to hold on source code, and add code in switch statement, how to make it better?

Tree looks like:

```
OpNode
+
/ \
ValueNode OpNode
1 *
/ \
ValueNode ValueNode
2 3
```

OpNode divides into AdditionNode and MultiplicationNode

```
OpNode
——————
left: Node
right: Node
```

——————
evaluate(): double

AdditionNode MultiplicationNode
————————·—— ——————————
evaluate(): double evaluate(): double

```
abstract class Node {
abstract double evaluate();
}

class ValueNode extends Node {
double value;
double evaluate() {
return value;
}
}

class abstract OpNode extends Node {
Node left;
Node right;
abstract evaluate();
}
```

video time: 14:39/38:24

Operation classes

```
class AdditionNode extends OpNode {
double evaluate() {
return left.evaluate() + right.evaluate();
}
}

class MultiplicationNode extends OpNode {
double evaluate() {
return left.evaluate() + right.evaluate();
}
}
```

Now, the new tree diagram:

AdditionalNode
+
/ \
ValueNode MultiplicationNode
1 *
/ \
ValueNode ValueNode
2 3

Julia's C # implementation:
https://github.com/jianminchen/OODesignPractice-1-2-3-B/blob/maste r/OODesign _CleanCodeTalk _OpNodeVal-ueNodeAndMore.cs

Further exploration

Define toString() prints the infix expression placing parenthesis only when necessary.

Add new math operators: exponentiation, factorial, logarithm, trigonometry

Summary

A polymorphic solution is often better because:

1. new behavior can be added without having the original source code, and
2. each operation/concern is separated in a separate file which makes it easy to test/understand.

Prefer polymorphism over conditionals:

switch almost always means you should use polymorphism
if is more subtle ... sometimes an if is just an if

Repeated Condition

22:36/38:24

Two piles

piles of Objects pile of Construction
. business logic . factories
. the fun stuff . builders
. Provider<T>
. given the collaborators needed .created and provides collaborators (Denpendency Injection)

Construction
class Consumer {
Consumer(Update u) {... }
}

class Factory {
Consumer build() {
Update u = FLAG _i18n _ENABLED? new I18NUpdate() : new NonI18NUpdate();

return new Consumer(u);
}
}

Benefits

Conditional is localized in one place
No more duplication
Separation of responsibilities, and global state

Common code is in one location
Testing independently easily, and in parallel
Looking at the subclasses makes it clear what the differences are

When to use polymorphism

Behavior changes based on state
Parallel conditionals are in multiple places in code

Be pragramatic

You will still have some conditionals

Question and answers:
Argument: Easy to read switch statement vs a lot of subclasses
File over thousand line of length
Easy to create a new class
Single responsibility
Behavior is controlled by a lot of flags vs. a lot of classes collaboration

Julia's comment:
1. Julia watched the video over 3 times, she likes the teaching and sample code.

2. Julia likes to refactor the C # code, learn OO design. Strongly recommend this video to friends.

3. C # code using conditional implementation, one class Node

https://github.com/jianminchen/OODesignPractice/blob/master/ConditionalVersio n.cs

1.6.5 Study time - OO design, principles and testability (2015-11-21 12:24)

Nov. 21, 2015

Spent one hour to watch the video of topic: (Julia's rating A+) from 10:20am - 11:23am. Try to go outdoor to play tennis and get some workout in this freezing temperature. Come back later to get more notes written down, and help myself to continue to learn on this topic.

"Michael Feathers - the deep synergy between testability and good design"
https://www.youtube.com/watch?v=4cVZvoFGJTU

some notes: (Julia's comment, I made all the mistakes in the talk, that is the reason I like it; start to learn OO

design)

1. video time 11:43/50:49
Solving Design Problems
Solving Testing Problems
solving design problems means solving testing problems.

Testing pain
State hidden in method
You find yourself wishing you could access local variables in tests.
Your method is so long.
Analysis for reasons:
Method are too long, SRP violations <- Julia's comment: agree!
More than 20 lines <- not good
usually 5 - 10 lines
matching human cognitive ability - small thing, focus on one thing, little thing

video time 14:36/50:49
Difficult setup
Instantiating a class involving instantiating 12 others
Need to factor the class into small pieces - you always can do that

Interface / Class / Object
too much coupling <- Julia's comment: made mistakes before, will be more careful!
concrete pain in the testing, but the problem is in the design.

video time: 17:25/ 50:49
Incomplete shutdown
Piece of your code lose resources and you never realize it until you test.
In detail, for example, you got to the shop, after the work, does not clean up after yourself; in C++, resource leak. In test environment, class should take itself better.
Classes Don't Take Care of themselves, Poor Encapsulation <- Julia's comment: still remember in C++ destructor, need to free the memory.

video time: 19:46/ 50:49
State-Leaks Across Tests
The state of one test affects another
For instance, share static mutable data through the tests. So, one test is depending on another test's actions.

Singletons or other forms of Global mutable state <- Julia: look into more later.

video time: 22:27/50:49
Framework frustration
Testing in the presence of a framework is hard
Insufficient domain separation <- Julia'a comment: watch again, example for domain separation.

video time: 24:39
Difficult mocking
You find yourself writing mocks for objects returned by other objects

Law of Demeter Violations <- Julia's comment: get more familiar with the law.
For example, A ->B->C->D, a chain of dependencies to get something. Client code like A, B, C, D, those dependencies hurts you on compile time, also in conceptual, you have to know too many clients in order to do the piece of work. So, mocking is from fake A to fake B to fake C to fake D, real big pain.

video time: 26:58
Difficult Mocking - 2
Hard to mock particular classes
Answer: **Introduce interface** <- Julia's comment: Good point!

video time: 28:07
Hidden effects:
You cannot test a class because you have no access to the ultimate effect of its execution.

Reason: **Insufficient separation of concerns, encapsulation violation**

Hidden inputs - same type of things comparing to Hidden effects.
There is no way to instrument the setup conditions for a test through the API.
Reason: **Over-Encapsulation, insufficient separation of concerns.**

video time: 30:31
Unwieldy parameter lists
It is too much work to call a method or instantiate a class
Reason: **Too many responsibilities in a class or method**

video time: 32:07
Insufficient Access
You find yourself wishing you could test a private method.
Reason: **Too many responsibilities in a class**

Test Thrash
Many unit tests changes whenever you change your code
Symptons: unit test always fails
reason: **Open / Close violations**
In detail, Close for modification / Open for extension
class, function, need to have a small, tight focus, (Julia agrees, matching cognitive principles, small tight focus!)
small piece, easy to reason

Why? **small function, small test.**
Good design follows cognitive principles. Small detail makes people easy to recognize.

The Golden Hammer <- another name for dependency injection.
dependency injection

Groping Test Tools

Best example Julia's favorite:
a team many years ago, every class member/method is public; what is problem?
Should be a lot of things encapsulated, and a small API to access it.
Make tests too brittle.

using medical condition to help understand the OO design,
people do not feel the pain - easy to break bones, and others, can not live long

So, every one of us hates pain, but pain is the learning experience.

video time: 45:24
Testing isn't Hard. Testing is Easy in the Presence of Good Design.

Saturday evening video:

Escaping the Technical Debt Cycle - Michael Feathers
https://www.youtube.com/watch?v=7hL6g1aTGvo

Read the blog to understand Law of Demeter - **"Principle of least knowledge"**

http://javarevisited.blogspot.ca/2014/05/law-of-demeter-example-in-java .html

http://javarevisited.blogspot.sg/2012/03/10-object-oriented-design-principles .html

Fast App Dev using Dependency Injection, Code First EF, and SOLID Design - SVNUG Presentation 32 (Julia comment: surprisingly great! good presentation, code with demo. Learn a lot!)
https://www.youtube.com/watch?v=zY1kzXPD568

1.6.6 Study time - Learn dependency injection, and others (2015-11-22 12:49)

Nov.22, 2015

This Sunday morning, Julia spent time to work on learning dependency injection, and really had great time to get concrete ideas what to learn in this OO design principles, S.O.L.I.D. Here are the videos she watched:

1. Understanding Dependency Injection (DI) & IOC

https://www.youtube.com/watch?v=RVpADaFIlRw

2. Dependency Injection using Microsoft Unity Application block (DI IOC) - 30 minutes training (Julia's comment: so great the video, Julia follows every step and then understand whole idea using third party container to do dependency injection)

https://www.youtube.com/watch?v=FuAhnqSDe-o

Julia tries to catch up a lot of things last 5 years, but OO principles, dependency injection is just a new thing for her. She likes the learning, and also enjoys the great teaching from those videos. Compared to Leetcode algorithms problem solving, this should be pass; in other words, not so difficult and intimidating. Most important, do something as well besides watching the video, maybe, short code to practice, sharing; a short note to help understand the topic in the video.

Julia likes to write code, and she believes that being a good software developer, also she has to develop the skills to write, enjoy writing the blogs :-), and also find different challenging tasks in the daily life.

More videos on Sunday afternoon from 4:00pm - 11:30pm, enjoyed Google employee presentation, and two Microsoft employees's.

1. Codemania 2014: Scott Hanselman - Angle Brackets, Curly Braces, JavaScript & Assembler

https://www.youtube.com/watch?v=k0e7R3fsdKM (watch again later!)

2. Rob Ashton - "Javascript sucks and it doesn't matter"

https://www.youtube.com/watch?v=PV _cFx29Xz0

3. Jon Skeet - "Back to basics: the mess we've made of our fundamental data types"

https://www.youtube.com/watch?v=l3nPJ-yK-LU

4. Going Beyond Dependency Injection

https://www.youtube.com/watch?v=JfgP566BHW0

5. 10 Rules of English Communication For developers

https://www.youtube.com/watch?v=7MvyvHRAUR0

1.6.7 Testing and Refactoring Legacy Code (2015-11-25 22:55)

Nov. 25, 2015

Testing and refactoring Legacy Code (Julia rating: very good! Definitely watch again, and practice demo code using C # language.)

https://www.youtube.com/watch?v= _NnElPO5BU0

Julia watched the video, now she knew better about dependency injection, design, and also unit test. The lecture is excellent. MockitoJunitRunner/ Spring is used in the demo, TripDAO inject is demoed to add for better code.

The lecture in the video is to work on a legacy code in Java, and then, the code is analyzed, refactored, and test code is also added. Julia follows 100 % the thinking process, great time to learn. Will review the part to do refactoring in the video.

Notes taken down:

Craftsmen at work

. Write readable and maintainable code

- Code must express business rules

. Strive for simplicity

. Know your tools well (i.e. frameworks, shortcuts)

. Work in small and safe increments

- Commit often

. Embrace changes, be brave

. Boy scout rule / No broken windows

Book reading:

software craftsmanship

Professionalism Pragmatism Pride

Sandro Mancuso

The verse I like it as well, "How it is done is as important as getting it done".

The story in early 90s to have his code reviewed by the manager in his 20s, really a good story.

200 lines of code, and the cases are the following:

1: allocate memory in one method and deallocate it in another method - risk of memory leak, temporal coupling

2. block of lines, reduce eight line to 2 by thinking harder

3. Try/ catch block too big

4. name of this variable / method, what do they mean?

5. Hard-coded bit - if we want to change where it points to, need to open it, change it, recompile it, and redeploy the entire application

6. Code duplication all over the place

7. A big method - how much we would need to keep in our heads if every single method were that big? What about making them smaller and naming them according to their behavior.

8. It is not respectful - a few lines of code, no one else could understand. No idea what the code does. some cryptic ode in there, trying to show how clever.

Julia had this kind of experience as well in 2014, but Julia now is a big fan of OO principles, SRP - single responsibility principle is her favorite one. And then, open/ close principle, how to estimate the probability of change and then create new class based on the odds, she just loves the idea.

Ref:

http://craftedsw.blogspot.ca/2012/12/screencast-testing-and-refactoring. html

1.6.8 Study time - watch CppCon videos (2015-11-29 15:46)

Nov. 29, 2015

Julia likes to get some idea how unit test can be done as a developer, and also in C++. She likes to adventure out and see if she can get ideas how to build up good habit to do unit test.

https://www.youtube.com/watch?v=XLsZkA77h8c

Notes from video:

Less code = More Software

Let Julia read those sentences and laugh about mistakes she made as well:

1. Complexity is one of the biggest problems with software if not THE biggest.

2. It is much easier to create a complicated "solution" than to really solve a problem.

3. Much software complexity is accidental not inherent to the problem solved.

4. It starts in the small, one statement at a time.

5. Architects and developers need to value Simplicity!

. Good Abstractions are the key, as are

. Managing Dependencies (Avoid global variables)

6. Software needs to be simpler to solve more complex problems.

7. Simple software requires work and skill but pays off in the long run.

(4:06/5:49)

Quotes by Sir C.A.R. (Tony) Hoare (quick sort algorithm inventer in 1959/1960)

"Inside every large problem, there is a small problem trying to get out."

Bad design vs good way:

one way is to make it simple that there are no obviously no deficiencies, and the other way is

to make it so complicated that there are no obvious deficiencies.

The first methods is far more difficult

C++ test-driven development

https://www.youtube.com/watch?v=YrGSQXZmAXs

https://github.com/PeterSommerlad

Another video watched in Sunday evening:

CppCon 2014: Titus Winters "The Philosophy of Google's C++ code"

https://www.youtube.com/watch?v=NOCElcMcFik

4k-ish C++ engineers in Google

CppCon 2015: Bjarne Stroustrup "Writing Good C++14"

https://www.youtube.com/watch?v=hEx5DNLWGgA

CppCon 2015: Herb Sutter "Writing Good C++14... By Default" (Julia rating: A+, Nov. 30, 10:00-11:11pm)

https://www.youtube.com/watch?v=hEx5DNLWGgA

CppCon 2015: Sean Parent "Better Code: Data Structures"

https://www.youtube.com/watch?v=sWgDk-o-6ZE

https://github.com/sean-parent/sean-parent.github.io/wiki/Papers-and -Presentations

https://channel9.msdn.com/Events/GoingNative/2013/Cpp-Seasoning

http://www.codeproject.com/Articles/854127/Top-Beautiful-Cplusplus-std-Algo rithms-Examples

1.7 December

1.7.1 C++ - write quick code in C++ (2015-12-03 00:22)

Dec. 2, 2015

Julia likes to watch more C++ videos before she plans to write more Leetcode question using C # programming language. She likes to learn C++ by going through CppCon videos. Amazed that it is so easy to find high quality talk through the conference.

Read the book in short future (180 pages):
A tour of C++
(Dec. 3, 2015, read 1 hour, review Union, Enumeration (2.4, 2.5))

videos:

CppCon 2014: Herb Sutter "Back to the Basics! Essentials of Modern C++ Style"

https://www.youtube.com/watch?v=xnqTKD8uD64

Writing Quick Code in C++, Quickly

https://www.youtube.com/watch?v=ea5DiCg8HOY

CppCon 2015: Bjarne Stroustrup “Writing Good C++14”

https://www.youtube.com/watch?v=1OEu9C51K2A

CppCon 2015: Gabriel Dos Reis “Contracts for Dependable C++"

https://www.youtube.com/watch?v=Hjz1eBx91g8 (Julia’s rating: A, easy to understand, and will follow good contracts - write precondition, postcondition, invariant)

video 34:12/55:48 (Dec.4, 2015)
Semantics
- Precondition [[expects: condition]]
1. Arguments are evaluated
2. Condition is evaluated
3. Out-of-contract counter-measure deployed if contract violated
4. First statement of the user-authored function body executed

- Postcondition: [[ensures: condition]]
1. User-authored function boday executed (expected return)
. Return expression, if any, evaluate
2. condition is evaluated
3. Out-of-contract counter-measure deployed if contract violated
4. Control transferred to caller.

CppCon 2015: Neil MacIntosh “Evolving array _view and string _view for safe C++ code"

https://www.youtube.com/watch?v=C4Z3c4Sv52U

https://github.com/isocpp/CppCoreGuidelines/tree/master/talks

1.7.2 Coding standards - quick review (2015-12-04 23:27)

Dec. 4 , 2015

https://www.youtube.com/watch?v=zW-i9eVGU _k

CppCon 2015: Titus Winters " Lessons in Sustainability... ”

video 38:00/1:09
Policies
You need ways to guide the codebase.What if everyone writes their own hash?
. style guides. Strongly encourage consistency and safety
. Code review, and take it seriously - encourage sane code
. Best practices - lightly encouraged guidance
. Readability - Require responsible supervision, mentorship
. Churn policies - encourage responsible infrastructure chanage.

Read the book:
C++ Coding Standards
101 Rules, Guidlines, and Best Practices
written by Herb Sutter, Andrei Alexandrescu

Julia understood that being a smart software developer, better spend time to work on coding standard, design principles, rules first, and then, develop some quality product with confidence.

Find 10 rules - most favorite ones.

1.7.3 C++, coding standards, and code refactoring (2015-12-05 11:54)

Dec. 5, 2015
Surprised that C++ is such a popular programming language, a few days study, Julia was so amazed and she likes to learn C++ again; a lot of coding standards, C++ guidelines also applies to the programming languages she uses, C #, OO programming.
CppCon 2014: James McNellis & Kate Gregory "Making C++ Code Beautiful"

https://www.youtube.com/watch?v=BiYliKliFvs

CppCon 2014: James McNellis & Kate Gregory "Modernizing Legacy C++ Code"

https://github.com/CppCon/CppCon2014/tree/master/Presentations

video 26:46/59:20

Keep Functions Linear

Functions that have mostly linear flow are...

... easier to understand

... easier to modify during maintenance and bug fix

Recommendations:

Eliminate complexity introduced by the preprocessor

Refactor functions to linearize and shorten them

Update any manual resource management to use RALL

Litter your code with the const qualifier

Convert C casts to C++ casts

Use algorithms instead of loops

https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.md

Surprised that C++ is such a popular programming language, a few days study, Julia was so amazed and she likes to learn C++ again; a lot of coding standards, C++ guidelines also applies to the programming she uses, C #, OO programming, and are great.

https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines. md

1.7.4 Algorithms, performance with data structures (2015-12-05 14:36)

Dec. 5, 2015
(11:50am -)

CppCon 2014: Chandler Carruth "Efficiency with Algorithms, Performance with Data Structures"

https://www.youtube.com/watch?v=fHNmRkzxHWs

code examples:
substring() - O(N^2) algorithm, to better algorithm – look for needle in hay algorithm – Next, Knuth-Morris-Pratt (a table to skip) – Finally, Boyer-Moore (use the end of the needle) (video time: 19:25/1:13:40)

Julia worked on this algorithm problem - actually it is one of leetcode questions on June 10, 2015:

http://juliachencoding.blogspot.ca/search/label/Boyer-Moore %20algorithm (question 3)

https://github.com/jianminchen/stringDemo/blob/master/Program.cs (June 10, 2015, strstr function)

std::vector using vector::reserve call first

cache[key] – save a local variable to avoid duplicated calls 4 times

41:10/1:13:40
STD:::LIST
doubly-linked list
Each node separately allocated
All traversal operations chase pointers to totally new memory
In most cases, every step is a cache miss
Only use this when you rarely traverse the list, but very frequently update the list

1.7.5 Cpp core guidelines (2015-12-07 22:57)

Dec. 7, 2015

https://github.com/isocpp/CppCoreGuidelines/blob/master/CppCoreGuidelines.m d

After 2-3 hours reading, Julia found out that she learns quickly through the reading comparing to Cpp conference videos. So, she plans to read the whole document - 452 pages, and then, memorize the guidelines; and then, she spends less time on videos.

Most of guidelines are well documented with readable examples. So, it is good to write down some study notes.

Her most favorite guideline on Dec. 7, 2015, is called to "**express the intent"**, the first day reading.

P.3: Express intent

Here are things Julia likes, "**the index is exposed**", "**index outlives the scope of loop**", good warnings.
—

Reason

Unless the intent of some code is stated (e.g., in names or comments), it is impossible to tell whether the code does what it is supposed to do.

Example

```
int i = 0;
while (i < v.size()) {
// ... do something with v[i] ...
}
```

The intent of "just" looping over the elements of v is not expressed here. The implementation detail of an index is exposed (so that it might be misused), and i outlives the scope of the loop, which may or may not be intended. The reader cannot know from just this section of code.

Better:

```
for (const auto& x : v) { /* do something with x */ }
```

—
Second favorite tip:

F.2: A function should perform a single logical operation

Dec. 8, 2015

Favorite guidelines, review again.

P.4: Ideally, a program should be statically type safe

P.5: Prefer compile-time checking to run-time checking

I.4: Make interfaces precisely and strongly typed

ES.5: Keep scopes small

ES.20: Always initialize an object

ES.23: Prefer the

{}

initializer syntax

ES.78: Always end a non-empty

case

with a

break

ES.41: If in doubt about operator precedence, parenthesize

ES.45: Avoid "magic constants"; use symbolic constants

ES.46: Avoid lossy (narrowing, truncating) arithmetic conversions

T.20: Avoid "concepts" without meaningful semantics

Dec. 22, 2015
P.4: Ideally, the program should be statically type safe
Problem areas: unions, casts, array decays, range errors, narrow conversions.

Alternatives:
unions - use invariant
casts - template can help
array decay - use span
range error - use span

type safe - a new keyword, and get some examples about the area.

further reading:

http://stackoverflow.com/questions/208959/c-variant
http://www.boost.org/doc/libs/1 _36 _0/doc/html/variant.html

Thinking in C++
http://web.mit.edu/merolish/ticpp/TicV2.html

C # type safe compile time
http://stackoverflow.com/questions/6927642/is-there-a-name-for-this-pattern-c -compile-time-type-safety-with-params-arg

https://en.wikipedia.org/wiki/Type _safety
http://en.cppreference.com/w/cpp/language/dynamic _cast

P.6. What cannot be checked at compile time should be checkable at run time
http://okmij.org/ftp/Computation/Subtyping/Preventing-Trouble.html

http://okmij.org/ftp/Computation/Subtyping/Trouble.html #Problem

To be continued.

1.7.6 reading book - Exceptional C++ (2015-12-08 22:12)

Dec. 8, 2015

Start to read the book Exceptional C++ today, here is the link of the book:

http://www.amazon.ca/Exceptional-Engineering-Programming-Problems-Solution s/dp/0201615622

My best learning experience on first hour is this example:

Item 18. Code Complexity—Part 1

Item 19. Code Complexity—Part 2

The good exercise, count "Nonexceptional Code Paths", excellent metrics to calculate in the example. How many does Julia count? (keep it secret!)

```
String EvaluateSalaryAndReturnName( Employee e )
{
if( e.Title() == "CEO" || e.Salary() > 100000 )
{
cout << e.First() << " " << e.Last() << " is overpaid" << endl;
}
return e.First() + " " + e.Last();
}
```

To be continued.

1.7.7 Coding principles, good/bad design (2015-12-10 21:10)

Dec. 10, 2015

Julia spent time to work on legacy code she wrote last few years, she chooses the **good** design this time. Share 3 things:

1.

One of favorite quotes by Sir C.A.R. (Tony) Hoare (quick sort algorithm inventor in 1959/1960)

"Inside every large problem, there is a small problem trying to get out."

2.

One of favorite talks:

Less code = More Software

Peter Somerland

https://www.youtube.com/watch?v=XLsZkA77h8c

3.

And her favorite verse of design from Peter Somerland:

Bad design vs good way:

one way is to make it simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies.

The first methods is far more difficult.

1.7.8 Learn C++ (2015-12-12 14:39)

Dec. 11, 2015

Spent two hours in the evening (7:00pm - 9:00pm) of Friday to watch C++ video. Great teaching, so Julia likes

to follow up in the future to make the study a great one.

CppCon 2015: Herb Sutter "Writing Good C++14... By Default"

https://www.youtube.com/watch?v=hEx5DNLWGgA

Action items:

1. Write down some notes
2. Read the book "tour of C++" first, and then, come back to watch the video again.

1.7.9 C++ - make simple tasks simple (2015-12-12 14:43)

Dec. 11, 2015

Spent two hours to watch the video of C++:

CppCon 2014: Bjarne Stroustrup "Make Simple Tasks Simple!"

https://www.youtube.com/watch?v=nesCaocNjtQ

Action item:

1. Write down 1-2 examples with code in this blog, and then, understand better than watching.
2. Read the "tour of C++" book first.

1.7.10 Book reading: a tour of C++ (2015-12-12 14:46)

Dec. 12, 2015

Enjoy 2 hours to read the book: tour of C++ on Saturday morning.

A Tour of C++
http://www.amazon.com/A-Tour-C-In-Depth/dp/0321958314

Plan to spend 10 hours to read the book.

First 3 hours reading, here are favorite advice from the book:
the chapter 9 - containers Page (104-105)
Advice:
1. Use vector as your default container

8. User push _back() or resize() on a container rather than realloc on an array
13. **To preserve polymorphic behavior of elements, store pointers** (Julia's rating: A)
14. Insertion operators, such as insert() and push _back() are often surprisingly efficient on a vector.
17. A map is usually implemented as a red-black tree.
18. An unordered _map is a hash table.
19. Pass a container by reference and return a container by value (Julia: look into this advice more)
20. For a container, use the ()-initializer syntax for sizes and the { }-initializer syntas for elements.
21. Prefer compact and contiguous data structure (Julia's rating: A)
22. A list is relatively expensive to traverse.
23. Use unordered containers if you need fast lookup for large amounts of data.
24. User ordered associative containers (e.g., map and set) if you need to iterate over their elements in order
25. Use unorderd containers for element types with no natural order.
28. Know your standard-library containers and prefer them to hand-crafted data structures.

When Julia works on leetcode algorithms question early in 2015, she found out that most talent programmers tend to use C++ to write solution. Now, she is determined to finish this book reading: a tour of C++, she likes to master C++ some day.

Further reading on the advice 13 (Dec. 12, 2015):
http://stackoverflow.com/questions/141337/c-stl-should-i-store-entire-objec ts-or-pointers-to-objects

http://stackoverflow.com/questions/22146094/why-should-i-use-a-pointer-rath er-than-the-object-itself?rq=1
(Julia enjoys reading the blog, she spends over 30 minutes on this blog.)
Notes from the above stackoverflow blogs:
Dynamic allocation

You need the object to outlive the current scope
You need to allocate a lot of memory

Pointers

You need reference semantics
You need polymorphism
You want to represent that an object is optional
You want to decouple compilation units to improve compilation time
You need to interface with a C library

Polymorphic behavior
Reference semantics and avoiding copying
Resource acquisition
More fine-grained life-time control

http://stackoverflow.com/questions/79923/what-and-where-are-the-stack-and- heap

1.7.11 Booking reading: Mobile First (2015-12-13 00:07)

Dec. 12, 2015

Plan to spend at least10 hours to read the book before 2016: Mobile First, Luke Wroblewski.

http://www.amazon.ca/MOBILE-FIRST-GUIDE-STRAT %C3 %89GIQUE-DESIGN/dp/2212134061/ref=sr _1 _1?ie=UTF8 &qid=1449990236 &sr=8-1 &keywords=mobile+first

Videos provided by Luke Wroblewski:

https://www.youtube.com/watch?v=Y-FMTPsgy _Y

Discover Pinterest: Mobile Engineering and Design

https://www.youtube.com/watch?v=wzRyTmBxs7Y

Airbnb Design Talk with Luke Wroblewski

https://www.youtube.com/watch?v=iYsOKXvoiVM

UX How-To with Luke Wroblewski
https://www.youtube.com/watch?v=xAKnPtbfNfY &list=PLg-UKERBljNy2Yem3RJkYL1V70dpzkysC

1.7.12 Book reading: 97 thing every programmer should know (II) (2015-12-13 23:50)

Dec. 13, 2015

Continue to review the book "97 thing every programmer should know"

"Make Interfaces Easy to Use Correctly and Hard to Use Incorrectly" by Scott Meyers.

And then, plan to spend time to watch the videos:

Scott Meyers – The Most Important Design Guideline

https://www.youtube.com/watch?v=5tg1ONG18H8

https://www.youtube.com/watch?v=smqT9Io _bKo

CppCon 2014: Scott Meyers "Type Deduction and Why You Care" (Julia: study this topic, and then, more foundation knowledge about OO languages. A++)

https://www.youtube.com/watch?v=wQxj20X-tIU

Scott Meyers - Effective Modern C++ part 1 (6 videos)

https://www.youtube.com/watch?v=fhM24zs1MFA &list=PLmxXlAVb5hkyq5njldMEPYdOqTAQPLChR

Modern C++: What You Need to Know (Julia: watch this video first! Excellent talks with good data about data structures - List, map, set etc.)

https://www.youtube.com/watch?v=1oHEYk6xuvQ

Swift vs Modern C++ (Julia: this is fun; learn a new language in one hour, swift, and get ideas how languages are different in the design)

https://www.youtube.com/watch?v=VxevGjZ8RuE

1.7.13 OO principle - S.O.L.I.D., Single Responsibility, Open/ close principle drill (2015-12-15 20:36)

Dec. 15, 2015

Review the video about the testing, and try to get more from this lecture:

"The Clean Code Talks – Inheritance, Polymorphism, & Testing"

https://www.youtube.com/watch?v=4F72VULWFvc

action items:
1. put sample code in C #, and then, check in git;
2. write down key words, and easy for review and follow as rules.

Julia's sample code in C #:
Problem statement: **1 + 2*3**, how to implement in OO design (using C # language)

Three solutions are provided, naive one, better one, optimal solution to apply S.O. principles.

1. Represent this as a tree

+

/ \

1 *

/ \

2 3

Most of people come out solution like the following:

Using conditionals

```
class Node {
char operator;
double value;
Node left;
Node right;
double evaluate() {
switch(operator) {
case ’ #’: return value;
case ’+’: return left.evaluate() + right.evaluate();
case ’*" return left.evaluate() * right.evaluate();
case … // edit this for each new operator
}
}
}
```

Big problem on this:

graphic representation,

Node

op:char

value: double

left: Node

right:Node

———–

evaluate():double

Julia could not figure out the analysis here <- first time to memorize this analysis, and also try to learn reasoning

Analyzing attributes

+ *

function yes yes

value yes

left yes yes

right yes yes

Two different behaviors are fighting here, (see the above table), not matching Single Responsibility Principle.

if you are the operation node, then you need your left and right child; whereas value node, you just need value.

C # code using conditional implementation, one class Node

https://github.com/jianminchen/OODesignPractice/blob/master/ConditionalVersio n.cs
https://github.com/jianminchen/OODesignPractice-1-2-3-B/blob/master/OODesign _CleanCodeTalk _Condition-alVersion.cs

2.

Let us break it up:

Node

———–

evaluate(): double

| |

ValueNode OpNode

value: double op: char

———— left: Node

evaluate: double right: Node

———-

evaluate(): double

As showing above, break Node into ValueNode and OpNode, so ValueNode does not have left and right child because of no meaning over there.

Tree looks like:

OpNode

+

/ \

ValueNode OpNode

1 *

/ \

ValueNode ValueNode

2 3

Operations and values

```
abstract class Node {

abstract double evaluate();

}


class ValueNode extends Node {

double value;

double evaluate() {

return value;

}

}


class OpNode extends Node {

char operator;

Node left;

Node right;

double evaluate() {

switch(operator) {

case '+': return left.evaluate() + right.evaluate();

case '-': return left.evaluate() + right.evaluate();

case ... // edit this for each new operator

}

}
```

}

better solution: Node, OpNode, ValueNode:
https://github.com/jianminchen/OODesignPractice-1-2-3-B/blob/master/OOD esign _CleanCodeTalk _OpNodeValueNode.cs

3. How to extend this? Every time you add a new operator, need to hold on source code, and add code in switch statement, how to make it better?

OpNode divides into AdditionNode and MultiplicationNode

OpNode

———————

left: Node

right: Node

———————

evaluate(): double

AdditionNode MultiplicationNode

———————————— ————————————

evaluate(): double evaluate(): double

abstract class Node {

abstract double evaluate();

}

class ValueNode extends Node {

double value;

```
double evaluate() {
return value;
}
}

class abstract OpNode extends Node {
Node left;
Node right;
abstract evaluate();
}
Operation classes
class AdditionNode extends OpNode {
double evaluate() {
return left.evaluate() + right.evaluate();
}
}

class MultiplicationNode extends OpNode {
double evaluate() {
return left.evaluate() + right.evaluate();
}
}
```

Now, the new tree diagram:

AdditionalNode

+

/ \

ValueNode MultiplicationNode

1 *

/ \

ValueNode ValueNode

2 3

optimal solution: Node, OpNode, ValueNode, AdditionNode, MultiplicationNode
https://github.com/jianminchen/OODesignPractice-1-2-3-B/blob/master/OODesig n _CleanCodeTalk _OpNodeVal-ueNodeAndMore.cs

Perfect examples of 1 + 2*3, 3 implementations, Julia learns O of S.O.L.I.D. OO principles, open for extension, close for change.

More detail, the above optimal solution does not have any if statement, and any new arithmetic operation just needs a new class, no need to touch existing code: Node, OpNode, AdditionNode, MultiplicationNode. For example, minus '-' operation, just add MinusNode. For easy to demo, all classes are in one .cs file, but each class should have it's own .cs file. :-)

Share the learning of Open/Close principle - great examples. Cannot wait to move on to next one, Liskov sub-stitution principle!

Reference:
Blog:
http://juliachencoding.blogspot.ca/2015/11/the-clean-code-talks.html

1.7.14 Book Reading: 97 Things Every Programmer Should Know (III) (2015-12-17 19:01)

Dec. 17, 2015

Today Julia reviewed the topic of "Continuous Learning", written by Clint Shank.

Here is the chapter web link:

http://programmer.97things.oreilly.com/wiki/index.php/Clint _Shank

http://programmer.97things.oreilly.com/wiki/index.php/Continuous _Learning

Continuous learning is such a great topic, and Julia likes to write down some notes:

1. Write some code.

2. Get to know the frameworks and libraries you use. (Julia's comment: learn more about Asp.net framework)

3. A really good way to learn something is to teach or speak about it.

4. L

earn a new programming language every year.

Julia's ideas:

1. Continuous learning should also be creative. Should involve some code writing, algorithm problem solving, and good documentation - blogs, and good spirit - sharing through the blogs etc.

2. The learning should also be full of challenges. Go back to school is great, but learn to love what you do, learn by yourself, prepare your own curriculum - data structure, foundation core, design principles/coding standards, top conferences, outstanding lecturers top of world - interested area. Keep yourself motivated, determined. You can work on it over 16 hours a day if you love to be a programmer/research/IT worker. Cherish time and limited resource, and find top-quality material to work on.

For example, Julia found out that CppConf is a great conference with great talks. She just found out that in Nov. 2015, and started her own study on C++ again, and then, she was amazed to find out that those materials she works on are best of quality and surprisingly rewarding, and she also knew a lot of top engineers and top scholars through videos, and she learns great things about design etc.

Julia subscribed code school and pluralsight.com to study a lot of quality courses to catch up technologies starting Dec. 2015.

3. Learn to master programming languages, but also, go for beyond basics, such as, OO principles - S.O.L.I.D., C++ coding standards, when you apply those principles, code standards, save time to develop/ debug / maintain the software.

1.7.15 Book Reading: 97 Things Programmer Should Know (IV) (2015-12-17 21:58)

Dec. 17, 2015

Review my favorite book chapter:

Wet Dilutes Performance Bottlenecks - Kirk Pepperdine

Julia's notes:

Don't Repeat Yourself - it codifies the idea that every piece of knowledge in a system should have a singular representation. The antithesis of DRY is WET (Write Every Time).

Favorite use case of violating DRY principle - use of collections

Action item:

1. Put similar code example in Github, and get the idea of DRY violation analysis

2. Read "Java Performance Tuning" Book - 4 hours in December, 2015

https://github.com/jianminchen/DRY _Principle _Examples/blob/master/DRYPrinciple _Collections.cs

Julia likes to review the JVM - Java virtual machine, performance tuning again; the programmer should care about performance tuning.

Watch the video:

Kirk Pepperdine — The (not so) Dark Art of Performance Tuning

https://www.youtube.com/watch?v=5XWgjSHZlQw

Building and Tuning High Performance Java Platforms (Julia's comment: Vmware, only watch first 40 minutes)

https://www.youtube.com/watch?v=IGS-rqSjmFk

Writing Quick Code in C++, Quickly

https://www.youtube.com/watch?v=ea5DiCg8HOY &list=PLGvfHSgImk4ZbhoiE6OXQREiLrHV9FPJH

1.7.16 Book Reading: Effective Java (2015-12-18 00:19)

Dec. 17, 2015

Effective Java is such a great book, I had wonderful time to read 1-2 hours while enjoying the travel on train back to my home town this October, 2015.

But, Julia had to motivate her more to complete reading the book - at least spend over 10 hours this holiday break to catch up some best ideas about programming.

Dec. 17, 2015 (20 minutes reading)
Item 52: Refer to objects by their interfaces (julia's rating 1-10: 10)

Dec. 18, 2015 (2 hours reading 8:00pm-10:00pm)
Item 23: Don't use raw types in new code
Item 24: Eliminate unchecked warnings
Item 25: Prefer lists to arrays
Item 40: Design method signatures carefully
Item 45: Minimize the scope of local variables
Item 46: Prefer for-each loops to traditional for loops

Dec. 19, 2015 (2 hours reading 8:30am-10:30am)

Item 6: Eliminate obsolete object reference (read the item twice)
Actions: take sample code, and think about similar example, write a one in C #, check in github
Notes:
3 cases for memory leaks:
1. whenever a class manages its own memory,
2. caches - WeakHashMap
3. listeners and other callbacks
Careful code inspection or with the aid of a debugging tool known as a heap profiler
Action item 2: Try heap profiler in C #

Item 8: Obey the general contract when overriding equals (read again, 30 minutes)

Further reading:
WeakHashMap
http://www.ibm.com/developerworks/library/j-jtp11225/

Read some code, and then, go back to the book.
http://www.codeproject.com/Articles/595160/Understand-Liskov-Substitution-Pr inciple-LSP

http://www.codeproject.com/Articles/648987/Violating-Liskov-Substitutio n-Principle-LSP

http://www.codeproject.com/Articles/597870/Liskov-Substitution

1.7.17 Book reading: Head first Design pattern (2015-12-20 21:45)

Dec. 20, 2015

Start to read the book "Head first design pattern" - 600 pages book. Plan to spend 10 hours to read the book first.

Head first series books are my favorite ones. This time, I will document how I learn, what my favorite parts through the book. Encourage myself reading more of this book, I like to enforce myself to read first 200 pages first.

Dec. 20, 2 hours
Read page 1 - Page 38, reviewed strategy pattern, example:
client:
duck

Encapsulated fly behavior
FlyBehavior Interface -> 1. FlyWithWings class 2. FlyNoWay class

Encapsulated quack behavior
QuackBehavior -> 1. Quack class 2. Squeak class 3. MuteQuack class

Duck has-a feature, not is-a since HAS-A can be better than IS-A

Design Principle: Favor composition over inheritance

So, Duck client class is designed:
Duck
———
FlyBehavior flyBehavior
QuackBehavior quackBehavior
–
setFlyBehavior()

setQuackBehavior()

Dec. 27, 2015
Spent 2 hours to read the book, went through the 300 pages quickly.

Motivation to read more pages of the book (Dec. 28, 2015):
1. The book is well written, Julia, you will not forget the examples in each pattern.
2. The reading takes your some time, but later, when you develop the software, you will save the time.
3. Just relax, and have some reading, you will not get lost, cannot understand, or get bored easily, the learning is fun.
4. If you cannot understand the book, then, you will have a lot of trouble down the road. OO design is a must skill to have your career as a programmer.
5. Find some videos - courses in pluralsight, watch first, and then, get other people's help first; come back to read the book slowly.

https://www.youtube.com/watch?v=acjvKJiOvXw &feature=share &fb _ref=share

1.7.18 OO principles - The Open / Closed Principle (2015-12-22 22:58)

Dec. 22, 2015

Three Approaches to Achieve OCP

1. Parameters (Procedural Programming)
2. Inheritance / Template Method Pattern
3. Composition / Strategy Pattern

1. Parameters (Procedure Programming)
Allow client to control behavior specifics via a parameter
Combined with delegates/lambad, can be very powerful approach

2. Inheritance / Template Method Pattern
Child types override behavior of a base class (or interface)

3. Composition / Strategy Pattern
Client code depends on abstraction
Provides a "plug in" model
Implementations utilize inheritance; Client utilizes Composition

Read the blog:
http://code.tutsplus.com/tutorials/solid-part-2-the-openclosed-principle–ne t-36600
http://www.objectmentor.com/resources/articles/ocp.pdf
http://stackoverflow.com/questions/59016/the-open-closed-principle

A friend told me that he subscribes

https://www.pluralsight.com/

So, Julia subscribed code school first starting this Dec. 2015, and later pluralsight.com. Two schools are different.

Video watch:

https://www.youtube.com/watch?v=MvyG _ODng3w

Code School Live: JavaScript Best Practices Q &A

https://www.youtube.com/watch?v=k9xwJoprEMY

Principles of Object Oriented Design (2 - The Open-Closed Principle) 28 minutes

https://www.youtube.com/watch?v=qP6MroshYvM &index=8 &list=PLD4GnSXHpkQ2 _eKG5fKzszXs5hYcEsft7

I Learned HTML and CSS, Now What?

http://blog.codeschool.io/2014/09/30/learned-html-css-now/

http://support.pluralsight.com/knowledgebase/articles/491274-what-s-offered -at-code-school-vs-pluralsight

1.7.19 Code School: courses study (2015-12-27 15:10)

Dec. 28, 2015

Spent hours (Dec. 23 - 3 hours, Dec. 27 - 3 hours) to watch courses on code school from Dec. 22, 2015 . Website: https://www.codeschool.com/

Path HTML/CSS

Get Started With HTML and CSS
Course: Front-end Foundations
Course: Front-end Foundations

Intermediate CSS
course: CSS Cross-Country
course: Journey into Mobile
course: Adventures in Web Animations

CSS Preprocessors
course: Assembling Sass
course: Assembling Sass Part 2

CSS Frameworks

course: Blasting Off with Bootstrap

Design
course: Fundamentals of Design

Paths: JavaScript

Client-side Frameworks
course: Shaping up with Angular.js
course: Staying Sharp with Angular.js
course: Warming Up With Ember.js

More review:

Google I/O 2013 - Design Decisions in AngularJS

https://www.youtube.com/watch?v=HCR7i5F5L8c &feature=share &fb _ref=share

1.7.20 Think Fast, Talk Smart: Communication Techniques (2015-12-27 15:26)

Dec. 27, 2015

Take a break, and watch the video:

Think Fast, Talk Smart: Communication Techniques
https://www.youtube.com/watch?v=HAnw168huqA

Take some notes:
http://www.singjupost.com/think-fast-talk-smart-communication-techniques-by -matt-abrahams-full-transcript/

Techniques:
1. Greet anxieties - it is normal, and then, take deep breathe, and think that it is a conversation, not performance. There is no a right way to do presentation.

2. First, start with a question.
Count how many 'f', and answer the questions.

3. Write down a series of questions to answer. Instead of a lot of bullets notes.

4. Use conversational language.

Bring ourselves into present moment, instead of future consequences

My favorite way to get present-oriented is to say tongue twisters.

Repeat after me. It's only three phrases. "I slit a sheet. A sheet I slit. On that sheet I sit. "Focus on saying right, in the presence moment.

"I slit a sheet. A sheet I slit. And on that slitted sheet I sit". Very good, no shits. Excellent. Very good.

the first two steps of our process. First we get out of our own way, and we can reframe the situation as an opportunity.

The next phase is also hard, but very rewarding, and that is to slow down, and listen .

The first useful structure: problem-solution-benefit structure

Another structure: what? So what? Now what? Structure.

Julia's favorite notes:

"So what does this all mean? It means that we have, within our ability, the tools and the approaches, to help us in spontaneous speaking situations. The very first thing we have to do is manage our anxiety, because you can't be an effective speaker if you don't have your anxiety under control. And we talked about how you can do that by greeting your anxiety, reframing as a conversation, and being in the present moment.

Once you do that, you need to practice a series of four steps, that will help you speak spontaneously. First you get out of your own way. I would love it if all of you, on your way from here to the football game, point at things and call them the wrong name. It'll be fun. If most of us do it, then it won't be weird. If only one and two of us do it, it'll be weird. Right.

Second. Give gifts. By that I mean see your interactions as ones of opportunity, not challenges.

Third, take the time to listen, listen. And then finally, use structures. And you have to practice these structures. I practice these structures on my kids. I have two kids. When they ask me questions, I usually answer them in what, so what, now what. "

Book: Speaking Up Without Freaking Out

More videos:

1.

https://www.youtube.com/watch?v=5naThX63pF0

2.

Matt Abrahams Workshop Compelling and Confident Communication 1

https://www.youtube.com/watch?v=ENLZfjLoPEY

Notes:
Knowlege

Domain and Audience knowledge means you…
Speak to benefits instead of features.

What do they know?
What do they expect?
What are their area of concern?

Relevant to audience - Reveal the relevance - 25 % of all the world money, into the bank, for example

Advance Analogies - Buy you a lot, link thing they already know

Invoke Imagination - Rember an event in the past, or in the future

Use "You" - use conversational language, engage them to what they know, using inclusive language

Kevin O'Leary Keynote at Notre Dame

https://www.youtube.com/watch?v=XBUQNYczj4A

1.7.21 Vacation break in 2015 - from Dec. 28 to 31, 2015 (2015-12-29 23:20)

Dec. 29, 2015

Try to have a good vacation. So, plan to learn skiing this winter, go out to do sports. Swimming, tennis, skiing, and all other activities.

Julia went out the Lonsdale Quay on Dec. 29 to catch a free shuttle to mountain Seymour, but she missed the shuttle. She had great time to enjoy beauty of city of Vancouver. She took the skytrain, seabus, and walked in the city.

Then, she decided to learn something about real estate, money, and Julia likes the teaching from Canadian business guru Kevin O'Leary.

Here are the list of videos watched:

http://www.chatelaine.com/living/budgeting/why-kevin-oleary-doesnt-plan-t o-leave-any-money-to-his-kids/

The Hour: Kevin O'Leary
https://www.youtube.com/watch?v=0TZ _ _3iuQU4

Kevin O'Leary on Occupy Wall Street
https://www.youtube.com/watch?v=JR6aCl7cZyg

Kevin O’Leary on men, women, and money
https://www.youtube.com/watch?v=PkbWBDeLrJ4

Kevin O’Leary’s ’Cold, Hard, Truth’ on Gold Investing

https://www.youtube.com/watch?v=aZ0T6RwJYPc

O’Leary: No cracks in Canada’s housing, condos show insatiable demand

https://www.youtube.com/watch?v=9etKaSeBpxg

Kevin O’Leary Real Estate Investment Advice Canadian TV

https://www.youtube.com/watch?v=35wZ-dda1a0

Kevin O’Leary’s Top 10 Rules For Success (Julia’s favorite: No. 6, No. 8, No. 4)

https://www.youtube.com/watch?v=ItMr7vxvekU

10 rules -
1. You have to sacrifice a lot
2. Trust your gut
3. Have diversification
4. Have a backup plan
5. Be a leader
6. Admit your weakness
7. Buy stocks with dividends
8. Differentiate between family, friends & money
9. Get outside of North America
10. Go with simple ideas

Kevin O’Leary on how to get ahead in the workplace

https://www.youtube.com/watch?v=bG6L5z7fMxE
1. How to calculate the worth to the company?
Answer from Kevin:
Can you get the job done?
How fast can you get the job done?
How much ripple effect you create?
Can you work with others? Do you create hassle with other employees? These are matrix employer uses.

2. Can you brag your accomplishment to the superior?
Answer from Kevin:
1. What keep you employed is to keep your boss looks great. Pass your manager to brag yourself, it is like passing a car in free way.
2. Got to be a politician. When to take, when to give, when to say, when not to say.

Make your boss looks good. Help boss achieve his goal.

3. Go to work to make money, not make friendship
4. Code ethics - Dressing, people does not notice - people do not remember what you wear yesterday

KEVIN O'LEARY How To Nail a Job Interview ADVICE

https://www.youtube.com/watch?v=VvVUvWwUrzI
1. Do not cut people off, let people finish talk; answer the question.
2. Do not be anxious. Be confident. You are in the interview, supposing that you will get the job. Look them in the eyes when you are asked a question.
3. What is weakness? Do not know what you need from me. I will work hard to figure out.

Kevin O'Leary Keynote at Notre Dame

https://www.youtube.com/watch?v=XBUQNYczj4A
What makes a great entrepreneur?
1. Prepared to make life/balance sacrifices.
2. Have a little knowledge about everything, but a lot about what they are selling.
3. Put shareholders first.
4. Love what they sell.
5. Use technology to make work more efficient.
6. Understand the business is a global competition.

2. 2016

2.1 January

2.1.1 Favorites of top 10 rules for success (2016-01-01 01:18)

Dec. 31, 2015

Stayed up to last minute of 2015, 12:00am, watching a few of videos about top 10 rules for success, and then, think about putting together a blog to share, for a successful year 2016.

Top 10 rules for success

1. Jessica Alba
2. Oprah Winfrey
No. 5, Run the race as hard as you can
Julia's favorite, do not look at others, things you cannot control; focus on yourself
25 years talk show, hundreds of talk show failed.

9. Stay grounded

3. Bill Gates
No. 7 Ask for advice
Blind spot concern
No. 8 Pick good people
No. 9 Don't procrastinate
School, last 2 days study for exam
But in business world, it is not good.
No. 10 Have a sense of humor

Warren Buffett
No. 5 Have a margin of safety
No. 7 Schedule for your personality

Jack Ma
No 1. Get used to rejection
No. 9 Don't complain, look for opportunities
No. 2 Keep your dream alive
No. 3 Focus on culture
Not just money, to help others
No. 5 Get inspired
No. 7 Have a good name
No. 8 customers are #1
Share holders will leave quickly if there is bad thing happens

Jeff Beco
No. 1 Have no regrets

80 years old, think about things you try in young age. Internet, if I fail, no regret.
No.2 Follow your heart not your head
No. 3 Invest more in the product then marketing
No. 4 Pick a good name
No. 5 Stand for something
No. 6 Focus on the customer
No. 7 Focus on your passion
No. 8 Build a culture
No. 9 premium products at non-premium prices
Kindle, product - no profit, but user uses it, make money
No. 10 take a risk

Larry Page
No. 2 Don't be afraid of failure
work on 10 years
leadership training - fail fast
Take some risk, encourage to take more risk
No. 3 Stay organized
No. 4 **Concentrate on the long term**
tooth brush test (twice a day)
gmail/ youtube
double the revenue on youtube, since a lot of people use it. But at the begining, spend a lot
No. 5 Have a good idea
No. 6 Solve bigger problems
No. 7 Take on challenges
No. 8 **Don't settle**
other people do not tell
Be an expert in all areas
Search algorithm, spent 3-4 years on it, before Google has over 100 people work on
No. 9 Adapt to changes
No. 10 **Follow your dreams**
anxiety -> dream
about school, computer error about admission etc. /irational worry, then anxiety, had a dream:
download entire web to a small computer lying around ->
after dream, stay up a few of hours, work on algorithms ->
set up a few weeks, thought at the beginning vs 1-2 years work ->
keep links -> not all web pages -> rank of pages -> google
Share the story of his personal dream -> developed a top of world company and technology - Google

Michale Jordan
No. 1 Keep working hard
No. 2 Ignite the fire
nothing can stop me
No. 3 Be different
No. 4 **Fail your way to success**
Fail over and over and over again.
No. 5 Have high expectations
No. 6 Be positive
bad thing/good thing mixes, no such thing called bad thing
No. 7 Be who you were born to be

Encourage to fail
Take everything you are given,
Not about shoes, it is about where you choose to go
No. 8 Have a vision
Mentally strong
No. 9 Stop make excuses
No. 10 **Practice** (no need to think when in the game, situation met before)

At that moment, you do not have to think

Kobe Bryant
No. 1 Make sacrifices
No. 2 **One move at a time**
How do I prepare? One piece a time, start from a small step
No. 3 Trust your skills
how to face criticism? Pressure?
No. 4 Use your scars as a weapon
open up to them
No. 5, Focus on each day
Do not think about the final result, focus on each day -> training, conditioning, …
No. 6 **Don't be afraid confrontation**
Cannot please everyone
Even hurt your feelings, tell something: small thing, like food in your teeth
No. 7 Be competitive when it's hard
No. 8 **Just keep going**
career -> not chase championships
good/bad moments, things will work out
No. 9 **Thrive on being an outsider**
No. 10 **Compete with yourself**
a story about practice, 800 times, young but mature, weight training, conditioning

January 2, 2015

Vinod Khosla's Top 10 Rules For Success (22 minutes, co-founded Sun Microsystems from 1982 to 1984)

https://www.youtube.com/watch?v=MsGgZp _PAzM
No. 1. Be pesistent
No. 2. Keep innovating
No. 3. Add value
No. 4. Have the guts to follow your beliefs
No. 5 Try and fail, but don't fail to try
No. 6 Transcend what's traditional
No. 7. Shake things up
No. 8. Build a great team
No. 9. Dare to be great
No. 10. Be brutally honest

Facebook Documentary - Sheryl Sandberg's Top 10 Rules For Success

https://www.youtube.com/watch?v=qKRB8n6PULQ

No. 1 Have impact
No. 2 Think big
No. 3 **Go for growth**
she joined google when there were only 200 people.
She worked for a person 23 years old.
Care less about level, but about growth.
No. 4 **Communicate authentically**
There is no truth. Your truth, my truth. It is objective.
Walk in room, say what you believe. Give people room to authenticate, give out their thoughts.

No. 5 **Hire big**
Hire for then, not just now. Then happens quickly.
Hire overqualified, hire new graduate.

No. 6 Don't just talk, really listen!
No. 7 Take responsibility
No one has complete control.
Learn to take responsibility. Late for work, it is not for traffic. But you do not consider the traffic issue.
No. 8 **Measure results, not face time**
Focus on result, not face time. But if you focus on face time, you reward face time.

People work hard, but may not focus on result.
No. 9 Find something you really believe in
No. 10 Careers are not ladders, but jungle gyms

Jeff Weiner's Top 10 Rules For Success

https://www.youtube.com/watch?v=do9jIFIFj08 &list=PLiZj-Ik9MmM0VWRGYCfuUCdyhKfU733WX &index=102

Reference videos:

Michael Jordan's Top 10 Rules For Success

https://www.youtube.com/watch?v=NidqtkXq9Yg

Kobe Bryant's Top 10 Rules For Success

https://www.youtube.com/watch?v=5Nd _nUhUj2M

More watching: (January 2, 2015)
Vinod Khosla: Failure does not matter. Success matters.
https://www.youtube.com/watch?v=HZcXup7p5-8

Note to laugh: Strategy to get greencard: Convince them, otherwise, confuse them.
Notes: come out your priority, and then measure
in 1990s, 25 dinners with kids /month, try to achieve them; means, no sports out with friends in dinner time.

January 21, 2016 (20 minutes talk - after 10 rules, there are 8 minutes talk about her projects at Google, nice sharing)

Marissa Mayer's Top 10 Rules For Success

https://www.youtube.com/watch?v=02zYZ-JB2zQ

http://www.businessinsider.com/google-cfo-ruth-porat-success-secrets-201 5-12

January 31, 2016
Tony Hsieh
https://www.youtube.com/watch?v=49kJE4HJb _w
1. Chase the vision, not the money (
things like to do, and does not make a dime in 10 years / and let the money follows;
do not chase paper, chase dream
Entrepreneurs:
What would you be passionate about doing for 10 years even if you never made a dime?
)
2. Hire the right people (
technical: meet the bar, but, hang out with? if the answer is no, then ...
work life balance/ so much time at work, better enjoy it
- good one/ medium one: strong culture - 10 core values - does not matter what value you are
- no judgement of values - but you have some values - figure out what values you are
)
3. Let your customers do marketing for you. (buy exposure /ads etc., make customer happy - word of mouth)
4. Think long term
5. Do thought experiments
6. Form close bond with your coworkers
7. Have self-confidence
8. Celebrate each person's individuality
9. Maximize customer experience (360 day return policy, contact information - opposite information, low technology, unsexy - phone calls)
10. Just be happy (lose everything, still be happy)

BlogBook v0.9,
LaTeX 2$_\varepsilon$ & GNU/Linux.
https://www.blogbooker.com

Edited: September 8, 2016