Julia's coding blog - Practice makes perfect

From January 2015, she started to practice leetcode questions; she trains herself to stay focus, develops "muscle" memory when she practices those questions one by one. 2015年初, Julia开始参与做Leetcode, 开通自己第一个博客. 刷Leet code的题目, 她看了很多的代码, 每个人那学一点, 也开通Github, 发表自己的代码, 尝试写自己的一些体会. She learns from her favorite sports - tennis, 10,000 serves practice builds up good memory for a great serve. Just keep going. Hard work beats talent when talent fails to work hard.

Tuesday, April 26, 2016

Leetcode 146: LRU - Cache

April 26, 2016

Design Last Recently Used Cache

https://github.com/jianminchen/LeetCode-Java-Solutions/blob/master/146.lru-cache.java

C# implementation with one test case:
https://gist.github.com/jianminchen/3dc7da7e0465819e6aa8c10c71901723

Question and answer:
1. What do you learn through the practice?
Use dummy head and dummy tail to help maintain double linked list as cache.

2. Do not forget to set the node to last one when it is visited.

3. Talk about API design:
AddToLast,
set
get - get method is confusing, since it is also to do repositioning of visited node in the linked list - the cache - better to add another function to let get() call, function - **getKeyAndThenMoveToLast**
use Dictionary to hold the key value pair,

4. when a node is added to last, with two dummy node in this double linked list, no temp variable needed to set up new connections. Just be patient, dummy tail's prev point: copy, and break, and set a new one.

May 17, 2016
Julia loves this article talking about cache system design; once she does some code review on basic LRU, she starts to enjoy the computer

A Chinese Canadian, she enjoys careers in China, United States and Canada. Change life style to play sports everyday, and enjoy every day to practice data structure and algorithm problem solving using C#. Used to live in yichun, Jiangxi province, Shanghai; Boca Raton, Florida; Now I am in Vancouver, Canada.

Her profiles:
https://www.linkedin.com/in/jianminchen
http://juliachencoding.blogspot.ca/
http://www.cnblogs.com/juliachenOnSoftware/
https://social.msdn.microsoft.com/Profile/jianmin%20chen

science, she just loves the engineering, good ideas to solve problems.
http://goo.gl/pL5ee4

classical reader/ writer problem / lock / multiple shards/ commit logs/ memcached

June 2, 2016

5. **How to design LRU? Data structure? What for?**
copy from blog:http://blog.gainlo.co/index.php/2016/05/17/design-a-cache-system/?utm_source=email&utm_medium=email&utm_campaign=email

LRU

One of the most common cache systems is LRU (least recently used). In fact, another common interview question is to discuss data structures and design of an LRU cache. Let's start with this approach.
The way LRU cache works is quite simple. When the client requests resource A, it happens as follow:

- If A exists in the cache, we just return immediately.
- If not and the cache has extra storage slots, we fetch resource A and return to the client. In addition, insert A into the cache.
- If the cache is full, we kick out the resource that is least recently used and replace it with resource A.

The strategy here is to maximum the chance that the requesting resource exists in the cache. So how can we implement a simple LRU?

LRU design

An LRU cache should support the operations: lookup, insert and delete. Apparently, in order to achieve fast lookup, we need to use hash. By the same token, if we want to make insert/delete fast, something like linked list should come to your mind. Since we need to locate the least recently used item efficiently, we need something in order like queue, stack or sorted array.
To combine all these analyses, we can use queue implemented by a doubly linked list to store all the resources. Also, a hash table with resource identifier as key and address of the corresponding queue node as value is needed.
Here's how it works. when resource A is requested, we check the hash table to see if A exists in the cache. If exists, we can immediately locate the corresponding queue node and return the resource. If not, we'll add A into the cache. If there are enough space, we just add a to the end of the queue and update the hash table. Otherwise, we need to delete the least recently used entry. To do that, we can easily remove the head of the queue and the corresponding entry from the hash table.

Eviction policy

When the cache is full, we need to remove existing items for new resources. In fact, deleting the least recently used item is just one of the most common approaches. So are there other ways to do that?
As mentioned above, The strategy is to maximum the chance that the requesting resource exists in the cache. I'll briefly mention several approaches here:

- Random Replacement (RR) – As the term suggests, we can just randomly delete an entry.
- **Least frequently used (LFU)** – We keep the count of how frequent each item is requested and delete the one least frequently used.
- W-TinyLFU – I'd also like to talk about this modern eviction policy. In a nutshell, the problem of LFU is that sometimes an item is only used frequently in the past, but LFU will still keep this item for a long while. W-TinyLFU solves this problem by calculating frequency within a time window. It also has various optimizations of storage.

Skip concurrency and distributed cache in this design.

Next, talk about coding part - data structure and algorithm:

C# implementation.
https://gist.github.com/jianminchen/3dc7da7e0465819e6aa8c10c71901723

int **capacity** - specify the size of cache, cache should be with limited size, since resource is limited, specially for high speed access. source code on line 24.

int **size** - current size of cache - track current size of cache to determine if eviction is needed or not.
source code on line 24.

Design a double linked list, so ListNode class is defined with two pointers, **prev, next**
source code from line 10 - line 22.

every entry has key, value. Use int to simplify the coding. Source code, line 12.
line 12 public int key, val;

Also, we need to add two more variables: dummy head, dummy tail to help maintain the double linked list. source code on line 25.

Also, we need to be able to find the key in O(1) since it is in cache. Extra space is used, maintain a hash map using Dictionary class in C#:
line 27 Dictionary(int key, ListNode)

Let us count how many variables inside the class LRUCache:
6 variables: - memorize the variable count - 6 - Try to recall.

```
private int capacity, size;
private ListNode dummyHead, dummyTail;
private Dictionary<int, ListNode> map;
```

ListNode class as a node in a double linked list:

```
 public int key, value;
 publie ListNode prev, next;
```

4 variables.

June 2, 2016
Warm up the algorithm: (a lot of hurdles, just read source code.)
https://gist.github.com/jianminchen/207e20632f7837285ee9b913b0d34f3a

Second warm up the algorithm:
https://gist.github.com/jianminchen/3bd8cdab7a31662d402c62fff9c0b597

solution - use queue ...
HackerRank: Even Tree - C# solutions to study (III...
HackerRank: Even Tree - Graph Problem (II) - Codin...
HackerRank: Even Tree - Graph Problem (I) - Just t...
Find if a Directed Acyclic Graph has a cycle.
Find the lowest common ancestor of two nodes in a ...
Tortoise-hare algorithm
Fisher-Yates algorithm
Largest continuous sub array problem
gray code
Remove Vowels From A String in Place
A binary tree is substree of another binary tree
valid parenthesis pairs
longest palindrome substring
Merge two sorted linked list
Search in Matrix
Two sum pair
Leetcode 17 Phone number combination - practice (I...
Insert value into a circle linked list
Two rectangles to see if they overlap
Find optimal weight
Matrix shortest distance set - get its Maximum val...
Window minimum
K nearest point
C# tutorial - reading first, coding follows
HackerRank: facts to share
video: How to Keep Calm at Crunch Time - Ask Ian #...
HackerRank: Matrix Rotation (Series 3 of 5) - usin...
HackerRank: Matrix Rotation (Series 2 of 5)
K index - Algorithm (II) using Queue

No comments:

Post a Comment

Enter your comment...

Comment as: Select profile...

Publish Preview

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

HackerRank: Connected Cells in a grid (V) - C++ so...
HackerRank: Connected Cells in a grid (IV) - JavaS...
Small talk - muscle memory from 100 hours to 500 h...
HackerRank: Connected Cell in a Grid (III) - C# so...
HackerRank - Connected Cell In A Grid (II) - C# so...
HackerRank: Connected Cell in a Grid - C# solution...
HackerRank: Matrix Rotation (Series 1 of 5)
HackerRank: String Calculate function (III) - LCP...
Advice for practice on coding question - easy, med...
rotate array
K Index algorithm
HackerRank: String Calculate function (III) - Suf...
HackerRank: String Calculate function (III) - Suf...
HackerRank: string function calculation (II) - str...
HackerRank: string function calculation - string a...
Mental skills to help players
HackerRank - count string (V) - C plus plus soluti...
HackerRank: count string (IV) - JavaScript
HackerRank: Count string (III)
HackerRank: Count strings (II)
HackerRank - String algorithm - Count Strings (I)
Elizabeth Holmes's Top 10 rules for success
Article reading: Get that coveted tech interview
Article reading: How to get hired at Microsoft
Article reading: Microsoft Interview Process

Labels

- 6 solutions (1)
- 8 neighbors using array to express (1)
- 8-Queens (1)
- 97 things for programmers (4)
- A comparison of Microsoft Web Technologies (1)
- a little progress every time (1)
- abstract example (1)
- academic advice (1)
- ACM cheat sheet (1)
- adherence to good design (1)
- adjacency list (1)
- Advanced Algo on HackerRank (3)
- Advice for beginners (1)
- algorithm (30)
- algorithm courses (1)
- algorithm interview (2)
- Algorithm night (2)
- Algorithm on coursera (1)
- algorithm research (1)
- Alien Dictionary (1)
- Alternating Characters (1)
- Amazon (1)
- Amazon Hiring (1)
- Amazon leadership principle study (1)
- Amazon/ High Standards (1)
- Ana Invanovic (1)
- anagram (6)
- Andy Murray (1)
- Angular (2)
- AngularJS (3)
- AngularUI (1)
- AngularUI fundamentals (2)
- antipattern (1)
- AorB (1)
- API design (1)
- argue (1)
- array (10)
- Array - brute force - 中间交换 (1)
- Array - brute force - 步步前移 (1)
- array construction (2)
- array construction (series 1 of 5) (1)
- Array Construction (Series 2 of 5) (1)

- Array Construction (Series 3 of 5) (1)
- array shuffle (1)
- arrowhead anti pattern (1)
- asp.net (1)
- attitude (1)
- automation test (1)
- avoid overcomplicated code (1)
- backtracking (8)
- bad design (1)
- base case (3)
- base case failures (1)
- be a good thinker (1)
- be in present (1)
- Bear and Steady Gene (I) (1)
- Bear and Steady Gene (III) (1)
- Bear And Steady Gene (IV) (1)
- Bear And Steady Gene (V) (1)
- Bear And Steady Gene (VI) (1)
- Bear Steady Gene (II) (1)
- beaten/ return (1)
- beautiful pairs (1)
- Being Geek (1)
- Benetrousle (1)
- best people (1)
- BFS (10)
- bible teaching (1)
- binary indexed tree (2)
- binary search (5)
- binary search tree (2)
- binary search tree related data type (1)
- Binary Tree (5)
- bit manipulation (7)
- bit operations (1)
- blog review (1)
- blogging (2)
- blogs reading (2)
- Bonetrousle (1)
- Book Reading (4)
- Book Reading: C# (1)
- Book Reading: Java PUzzles (1)
- Book reading: Productive programmer (1)
- Book Reading: Programming Challenges (1)

- book reading: The Algorithm Design Manual (1)
- book to read (1)
- bootstrap (2)
- bottom up approach (1)
- box engineer (1)
- Boyer-Moore (1)
- Boyer-Moore algorithm (1)
- BoyerMoore (1)
- Bronze Medal (2)
- brute force (4)
- brute force (center of string) (2)
- brute force Time O(NxN) Space O(N) (1)
- BST (3)
- BT lowest common ancestor (3)
- BTree Maximum Path Sum (1)
- BTree Vertical Order Traversal (1)
- BTree ZigZag Traversal (1)
- bucket sort (1)
- build a career (1)
- burst ballons (1)
- C# (1)
- C# expert (1)
- C# SortedSet or Sorted Dictionary (1)
- C# tutorial (1)
- C++ (10)
- C++ Set (1)
- C++. code standards (1)
- cache system (1)
- candidate coaching (1)
- career advice (2)
- carousel (2)
- Cendyn One (1)
- center of string (1)
- char to int (1)
- clean code (2)
- climbing stairs (1)
- clockwise (1)
- CMU (1)
- coaching (1)
- Coaching from Depei Zhang (1)
- code challenge (1)
- code guidelines (1)

- code review (5)
- Code Review - C# (1)
- Code Review New School Day(2) (1)
- Code Review New School Day(3) (1)
- Code Review New School Day(4) (1)
- code school (4)
- code smells (3)
- code standard (1)
- code study: LC380 (1)
- code study: Leetcode blogs (3)
- code styles (1)
- code styles study (1)
- CodeJam (1)
- coding standard (1)
- coding standards (1)
- combinations (1)
- comment (1)
- communication skills (1)
- communication techniques (1)
- company culture (1)
- Compare two linked list (1)
- competitive programming (1)
- competitive programming book (1)
- confidence and determination coaching (1)
- conforming to style guidelines (1)
- connected cell in a grid (3)
- containers (1)
- content marketing (1)
- continuous learning (1)
- Count (3)
- counter example (1)
- coupling (1)
- cpp code standard (1)
- cpp core guidelines (1)
- CSS (8)
- CSS selectors (1)
- Ctrl+C rule (1)
- Cyclomatic complexity (1)
- data anlytics (1)
- DB design (1)
- DBA (1)
- DDoS attack (1)

- delete a node (1)
- dependency injection (3)
- dependency list (1)
- dequeue (1)
- design (1)
- Design pattern (3)
- design principle (1)
- design talk (1)
- deterministic (1)
- DFA (1)
- DFS (16)
- DFS-phone number (1)
- DFS+backtracking ->DP (1)
- Dictionary vs Hashmap (1)
- difficult time (1)
- digital marketing (1)
- distance k from a node in binary tree (1)
- distributed system (1)
- distribution sort (2)
- divide and conquer (2)
- do one thing (1)
- Double ended queue (1)
- double linked list (2)
- DP (25)
- DP bottom up memorization (1)
- Dqueue (1)
- DRY (3)
- DRY principle (2)
- dummy head/tail (1)
- edmond lau (2)
- encapsulation (1)
- engineering talk (1)
- Eric Lippert (1)
- Everyone faces challenges and I am no differnt (2)
- express intent (1)
- external merge sort (2)
- facebook culture (1)
- fall/ get up (1)
- fast coding-..->LINQ (1)
- father and son (1)
- favor composition over inheritance (1)
- fear/insecurity/doubt (1)

- feel at home (2)
- Fibonacci sum (1)
- find merge point of two linked list (1)
- find the duplicates (1)
- first HackerRank Bronze Medal (1)
- Fisher-Yates algorithm (1)
- Five practices (2)
- fix the cycle (1)
- Fix the cycles (1)
- focus (1)
- Fog Creek Software (1)
- Forget everyone else and put the work in (1)
- framework (1)
- From HackerRank to Wechat public account (1)
- full stack web dev (1)
- functional programming (1)
- funny string (1)
- Game of Numbers (1)
- gemstones (1)
- get it right->cannot get it wrong (1)
- get node value (1)
- Get out on court and overcome yours (1)
- github leetcode solutions (1)
- global state (1)
- global variable (1)
- goal settings (1)
- good design (1)
- Google interview (3)
- Google talk: the effective engineer (1)
- google/quora/quip (1)
- graph (5)
- graph algorithm (1)
- graph alogrithm (4)
- great developer (1)
- greedy algorithm (5)
- Gridland Metro (1)
- Guide to Technical Development by Google (1)
- Hack/ Study/ Mentor Night (1)
- hacker culture (1)
- HackerEarth (2)

- HackerRank (22)
- HackerRank Bronze Medal (2)
- HackerRank contest (1)
- HackerRank NCR codesprint (3)
- HackerRank world codesprint #4 (1)
- HackRank (3)
- Hard (3+) (2)
- hard algorithm (2)
- Harvard professor/ Google manager/ Matt Welsh (1)
- Hashmap Time O(N) (1)
- hashset (2)
- hashtable (1)
- have guts to fail (1)
- Heap (2)
- high standards for practice (1)
- HourRank 6 (1)
- how to ace it - interview (1)
- how to find a mentor (1)
- html (3)
- hunger for more (1)
- I know what it takes to win (1)
- I learned to stay and work hard every day to get the chance to be the best (1)
- if logic (2)
- If you are behind (1)
- in place (1)
- in-place (1)
- industry research (1)
- inheritance (1)
- inheritance example (1)
- inject depencies (1)
- inorder traversal (2)
- insert a node at a specific position in a linked list (1)
- insert a node at tail of a linked list (1)
- insert a node at the head of a linked list (1)
- Inside every large problem there is a small problem trying to get out (1)
- intellectual curious (1)
- interface (1)
- Interval algorithm (1)
- interview (4)

- Interview cheat sheet (1)
- interviewer (1)
- Invert (1)
- iterative (4)
- ITInt5 (1)
- jagged array (1)
- Java (3)
- Java Design Pattern (1)
- Java Style Guide (1)
- Java TreeSet (1)
- JavaScript (12)
- JavaScript 2016 Year-End Review (1)
- JavaScript Array (1)
- JavaScript for C# developer (1)
- JavaScript study (1)
- JavaScript Style Guide (1)
- Joel Spolsky (1)
- journaling practices/ matches (1)
- Journey to LINQ (1)
- Jquery (2)
- Julia takes a break (1)
- Julia's 2 cents at work (1)
- Julia's new school (1)
- julyedu.com (1)
- Juniper (1)
- K index (1)
- k most frequent numbers in array (1)
- Karolina Pliskova (1)
- keep functions linear (1)
- Kevin O'Leary (1)
- key design for anagram (3)
- kindergarten adventures (1)
- kindergarten adventures (Series 1 of 3+) (1)
- Kindergarten adventures (Series 2 of 3+) (1)
- kindergarten adventures algorithm (Series 3 of 5+) (2)
- Kristina Mladenoiv tennis pro (1)
- LC 125 (2)
- LC 127 (1)
- LC 269 (1)
- LC1 (1)
- LC102 (1)

- LC103 (1)
- LC11-20 (1)
- LC124 (2)
- LC125 (1)
- LC125 optimal solution (1)
- LC139 (1)
- LC142 (1)
- LC15 (1)
- LC159 (1)
- LC164 (1)
- LC17 (4)
- LC20 (2)
- LC236 (1)
- LC238 (1)
- LC239 (1)
- LC295 (2)
- LC297 (1)
- LC310 (1)
- LC312 (1)
- LC314 (2)
- LC317 (2)
- LC318 (1)
- LC319 (1)
- LC322 (1)
- LC328 (1)
- LC329 (1)
- LC33 (1)
- LC331 (1)
- LC347 (1)
- LC4 (1)
- LC5 (1)
- LC53 (1)
- LCP (3)
- LCP array (1)
- leadership principles (1)
- Leander Paes (1)
- Learn from others (1)
- learn to lose (1)
- Learning Style change (1)
- leet code questions (1)
- leetcode (26)
- Leetcode 1-10 (1)
- leetcode 1-251 questions with web links

(1)

- Leetcode 124 (1)
- Leetcode 126 (1)
- Leetcode 126: word ladder II (1)
- Leetcode 146 (3)
- Leetcode 16 (1)
- Leetcode 17 (1)
- Leetcode 236 (1)
- Leetcode 239 (1)
- Leetcode 300 (1)
- Leetcode 322 (1)
- Leetcode 322: coin change (1)
- Leetcode 4 (1)
- Leetcode 56: Merge Intervals (1)
- Leetcode 66: plus one (1)
- Leetcode 88 (1)
- leetcode book (1)
- leetcode gitbook (1)
- Leetcode solution (1)
- leetcode solutions in github (1)
- lessons learned (1)
- Let recursive do the work (1)
- level order (1)
- Level Order Traversal (2)
- life coach (1)
- linear solution (1)
- linked list (13)
- Linked List Cycle (1)
- LinkedList (2)
- LINQ (2)
- LINQ 7 tricks (1)
- list (1)
- List (C#) (1)
- List To Tree (1)
- longest common prefix (1)
- longest common subsequence (1)
- longest increasing subsequence arrays (1)
- longest substring (1)
- longest substring without repeating chars (1)
- longevity of the career (1)
- loop (1)
- loops (1)

- lowest common ancestor (3)
- LP70 (1)
- LRU (1)
- LRU Cache (2)
- Luke Wroblewski (1)
- make that everything (1)
- Margaret Gould Stewart (1)
- math (5)
- Math induction proof dominates (1)
- matrix (1)
- Matrix Rotation (6)
- matrix rotation (Series 2 of 5) (1)
- Matrix Rotation (Series 1 of 3) (1)
- Matrix Rotation (Series 3 of 3) (1)
- Matrix Rotation (Series 4 of 4) (1)
- Matrix Rotation (Series 5 of 5) (1)
- matrix rotation various ideas (1)
- Matrix Spiral Print (1)
- Maximum Gap (1)
- maximum heap (1)
- maximum path sum (2)
- Maximum subarray sum (1)
- maximum value (2)
- Meag Tic-Tac-Toe (1)
- medium algorithm on HackerRank (1)
- meetups in Vancouver (1)
- memorization (2)
- mental skills (2)
- mental strength (2)
- mental toughness (2)
- mentor (1)
- merge k sorted array (1)
- merge sort (2)
- merge two sorted arrays (1)
- merge two sorted linked list (1)
- Microsoft interview process (2)
- minimum algorithm (2)
- Minimum Cost (Series 1 of 10) (1)
- Minimum Cost (Series 10 of 10) (1)
- Minimum Cost (Series 2 of 10) (1)
- Minimum Cost (Series 3 of 10) (1)
- Minimum Cost (Series 4 of 10) (1)
- Minimum Cost (Series 5 of 10) (1)

- Minimum Cost (Series 6 of 10) (1)
- Minimum Cost (Series 7 of 10) (1)
- Minimum Cost (Series 8 of 10) (1)
- Minimum Cost (Series 9 of 10) (1)
- minimum heap (1)
- missing C# code (1)
- MIT recursive lecture notes (1)
- mobile engineering (1)
- Mobile first (1)
- mobile web development course (1)
- mock interview (4)
- module 11 or module 21 (1)
- Morris order (1)
- motivation (1)
- muscle memories/ 100hrs - 500hrs (1)
- muscle memories/ wrist to back (1)
- MVC (2)
- N-Queens (3)
- N-Queens problem (2)
- NCR (2)
- NCR codesprint (2)
- netflix company culture (1)
- never complain (1)
- new school (1)
- NFA (1)
- Nishikori tennis pro (1)
- no fear (2)
- nonexceptional code paths (1)
- noSQL (1)
- not play to learn (1)
- NP Problem (2)
- O of SOLID principle (1)
- O(1) Space (3)
- O(1) time (1)
- O(1) time in subtree (1)
- O(1) time vs O(n) (1)
- O(N) (2)
- O(n) algorithm (1)
- O(n) Time (2)
- O(n^2) time (1)
- O(n^4)->O(n^3) (1)
- O(nw) Time (1)
- OCP (1)

- odds (1)
- on the court (2)
- one hour code screen (1)
- one level of Abstraction per function (1)
- one sole point (1)
- OnePlus2Times3 OO Design (1)
- OO (1)
- OO design (4)
- OO design interview (1)
- OO Design principles (1)
- OO Programming (1)
- Open/Close principle (3)
- operating system (1)
- optimal solution space O(1) (1)
- pageview by countries (1)
- palindrome (7)
- pangram (1)
- parentheses (3)
- parenthesis (1)
- partition (2)
- path sum (2)
- pattern match (1)
- peak element (1)
- peer review (1)
- phone number (3)
- pivot (1)
- play to win (3)
- pluralsight (10)
- positive thinking (1)
- post order traversal (1)
- Practice till you cannot get it wrong (2)
- Practice till you get it right (2)
- Preorder (1)
- print list (1)
- priority queue (1)
- problem solving (1)
- procedural function (1)
- product manager (1)
- Product of Array (2)
- Program challenge (1)
- programming contests (1)
- programming interview (1)
- programming principles (1)

- Programming style (1)
- python (1)
- queue (11)
- quick sort (1)
- quicksort (1)
- Radix Sort (3)
- radix sort code review (1)
- rand (1)
- random access (1)
- rat in maze (1)
- recurisve (1)
- recursion (1)
- recursion research (1)
- recursive (20)
- Recursive Call (1)
- recursive function (3)
- recursive function design (1)
- recursive intense training - Linked List (1)
- recursive lecture notes (1)
- reduce to minimum - module value (1)
- redundant code (1)
- redundant elimination in recursive (1)
- redundant if (1)
- redundant work on children (1)
- refactor (1)
- reference (1)
- remove duplicate (1)
- responsive design (1)
- Resume Screening (1)
- Revealing module pattern (1)
- reverse a doubly linked list (1)
- reverse a linked list (1)
- rotate array (1)
- scalability (1)
- Second HackerRank Bronze Medal (1)
- secret to the advancement (1)
- segment tree (3)
- Segment Tree and Lazy Propagation (1)
- segment tree tutorial (1)
- segmented tree (1)
- self-belief (1)
- serialization (1)

- shackle (1)
- Simon Halep Tennis pro (2)
- singleton (1)
- singly linked list (5)
- six times player of Olympics (1)
- slide window (3)
- sliding window (2)
- sliding windows maximum (1)
- small (1)
- smart pointer (2)
- smart programmer makes 10 times difference (1)
- software craftsmanship (1)
- software developer (1)
- software programmer (1)
- SOLID principle (1)
- sort algorithms (1)
- sorted array to tree (1)
- sorted list (1)
- sorted list to tree (1)
- sorting (1)
- space O(1) (1)
- spiral message (4)
- spiral message (series 2 of 3+) (1)
- spiral message (series 1 of 3+) (1)
- spiral message (series 3 of 3+) (1)
- sports training (1)
- sprinter vs. fast coding (1)
- SQL injection attack (1)
- SQL Server DBA (1)
- SQL Sript (1)
- srand (1)
- SRP (1)
- SRP principle (1)
- stack (9)
- stack exchange: code review (2)
- Stack Overflow (1)
- stackexchange code review (2)
- stackoverflow search (1)
- start point (1)
- strategy pattern (1)
- stress management (1)
- string (12)

- string functions review (1)
- string primitive (1)
- string problems (1)
- string search (1)
- StringBuilder (1)
- strStr (1)
- study on failing (1)
- subclass (1)
- subsequence (1)
- substring (2)
- succeed (1)
- successor search (1)
- suffix array (5)
- suffix array C# (1)
- swap (1)
- swap even bit and odd bit (1)
- system design (3)
- talk smart (1)
- taste success (1)
- taste success and hunger for more (2)
- tech events (1)
- Tech events in 2016 (1)
- tech interview (1)
- tennis (1)
- tennis sports and interviews (1)
- tennis sports study (1)
- tennis star (1)
- ternary tree (2)
- test (2)
- test thrash (1)
- testability (1)
- testing (2)
- Thanksgiving holiday Canada 2016 (1)
- the future of analytics (1)
- the Prototype Pattern (1)
- think fast (1)
- thoughts about sharing (1)
- thoughts on being competitive (1)
- Time analysis from O(N^3 -> N^2) (1)
- Time analysis O(N^3->N) (1)
- time complexity (1)
- time complexity highest priority (1)
- time O(N) (1)

- Time O(N^2) (1)
- time O(N^3) (1)
- Time O(N^3) to Time (O^2) (1)
- time O(NxN) (1)
- timeout (2)
- tips about style (1)
- to tree (2)
- Top 10 rules for success (2)
- top-down (1)
- TopCoder (1)
- Topcoder's tutorial (1)
- topological sort (1)
- train insane or remain the same (2)
- traversal (1)
- tree (10)
- tree algorithms review (1)
- tree maximum path sum (1)
- tree path sum (1)
- Tree Serialization (1)
- TreeSet (1)
- Try again. Fail again. Fail better. (1)
- Two pointers (6)
- Two String (2)
- Two strings (1)
- Two sum (1)
- type deduction (1)
- Udacity (1)
- undirected graph (1)
- union find (3)
- union-find (2)
- unit test (3)
- Unity application block (1)
- user experience (1)
- Uva online judge (1)
- variable - Please using meaningful name (1)
- warlmartlabs (1)
- web links (1)
- web programming (2)
- Web usability (1)
- wechat public account (1)
- when you are good at something (1)
- Why LINQ (1)

- wise words from top tennis players (1)
- without repeating chars (1)
- word break (1)
- word ladder I (1)
- word ladder II (2)
- work hard strategies (1)
- world code sprint #6 (1)
- world code sprint #7 (1)
- write coding blog vs stackoverflow (1)
- writing benefits (1)
- writing blogs (1)
- writing style (1)
- writing talent (1)
- zigzag (1)
- 动态规划 (1)
- 回文 (2)

Total Pageviews

Popular Posts

HackerRank: Bear and Steady Gene algorithm (III)

March 5, 2016 Problem statement: https://www.hackerrank.com/contests/hourrank-6/challenges/bear-and-steady-gene A gene is represent...

HackerRank: Bear and Steady Gene algorithm (IV)

March 5, 2016 Problem statement: https://www.hackerrank.com/contests/hourrank-6/challenges/bear-and-steady-gene A gene is represen...

HackerRank: HourRank 6

March 2, 2016 Julia spent more than 1 hour to work on first question on HackerRank: HourRank 6, Lisa's workbook. It's a one...

HackerRank: Equal stacks

June 26, 2016 Problem statement: https://www.hackerrank.com/contests/june-world-codesprint/challenges/equal-stacks C# practice: https:...

HackRank - New Year Present - ACM ICPC Practice Contest 2016

Nov. 29, 2016 Problem statement:

https://www.hackerrank.com/contests/acm-icpc-practice-contest/challenges/newyear-present Julia spent ...

codereview.stackexchange.com - Julia's new school (II)

Nov. 30, 2016 Introduction: Julia was so excited to read so many experts's well-written answers first time, on live sites. She just c...

Short Palindrome - HackerRank - world codesprint #5

July 25, 2016 Spent 3+ hours to score 13 out of 40 points. But, Julia enjoyed the time to work on the problem. A few issues, wrong ans...

Tennis sports and interviews

Nov. 26, 2016 Julia likes to play competitions on HackerRank in 2016, whenever she has problems, dealing with strategies, emotions, specia...

Stackoverflow - code review research

Nov. 27, 2016 Try to post the first code review on code review on stack exchange, code review section. When I try to put the title: L...

HackerRank: Count string (III)

April 9, 2016 Problem statement: https://www.hackerrank.com/challenges/count-strings Solution to study: https://gist.github.com...

0