ZJ's Algorithm

Collection of Best Solutions + Original Tutorials

SUNDAY, NOVEMBER 2, 2014

LRU Cache: Java O(1) solution with some test cases

1. reference for a simple LRU cache:
http://www.geeksforgeeks.org/implement-lru-cache/
http://www.programcreek.com/2013/03/leetcode-lru-cache-java/
An High-Throughput concurrent version by Ebay:
http://www.ebaytechblog.com/2011/08/30/high-throughput-thread-safe-lru-caching/#.VFZSGPnF-NM
Better but still simple: page replacement algorithm LIRS

3. In OS virtual memory, key is page number and value in hardware: memory or disk.

4. My Simple Implementation:
 4.1. HashMap key -> ListNode; key<-ListNode

 4.2. Doubly linked list: Head is most recently used, tail is least recently used, track size (max size is same with main memory).

 4.3. Implement doubly linked list: find cur, pre, post first. Use dummy preHead, postTail rather than check if (pre/post==null)
 only need: setHead(key, val); remove(DoublyListNode n); removeTail();

[Java Code]

SEARCH THIS BLOG

Search

ABOUT ME

Ethan Leo

Follow 0

View my complete profile

POPULAR POSTS

```java
import java.util.HashMap;

public class LRUCache {
 class Node {
  int key;
  int val;
  Node prev;
  Node next;
  public Node(int key, int val) {
   this.key = key;
   this.val = val;
  }
 }
 class DoublyLL {
  int size = 0;
  Node preHead;
  Node postTail;
  public DoublyLL() {
   preHead = new Node(-1,-1);
   postTail = new Node(-2,-2);
   preHead.next = postTail;
   postTail.prev = preHead;
  }
  public void setHead(Node n) {
   if(n.prev!=null && n.next!=null) { // if n in the list
    remove(n);
   }
   Node post = preHead.next;
   n.next = post;
   post.prev = n;
   preHead.next = n;
   n.prev = preHead;
   size++;
  }
  public Node removeTail() {
   if(size!=0) {
    Node n = postTail.prev;
    Node pre = n.prev;
    pre.next = postTail;
    postTail.prev = pre;
    n.next = null;
    n.prev = null;
    size--;
    return n;
   }
   return null;
  }
```

LeetCode java: Wildcard Matching
几个月前面TripAdvisor跪在这个题上，当时还以为被同胞黑了，其实就是LC原题，人家在放水... ——经验是：所有以为被人黑的时候，基本上都是自己太弱了 GeekforGeek 的解法不行，因为test case中有许多连续*的情况，而且直接象regular expr...

LeetCode in Java: Implement strStr()
从干草堆里找根针？面试遇到这个应该至少不是好意，那就别客气了。速度秀一下暴力，果断上Boyer-Moore，因为简单而且好写。写KMP？严重怀疑当场出错！ BM不行就纸上画图，会快些。BM有两种shift，可以只写简单的bad-character shift。 [Ref...

LeetCode in Java: Word Break II DP/暴力过大集合
Both brute force and DP solutions can pass test cases. 我和小伙伴们都惊呆了... 只要在最前面用WB I的dp先检测一下输入的有效性，就能过。 这题有人用像WB I一样的dp，boolean改为存储中间结果，最后T...

Leetcode java 解题报告：Balanced Binary Tree
[Problem] Given a binary tree, determine if it is height-balanced. For this problem, a height-balanced binary tree is defined as a bina...

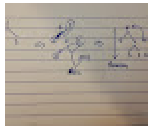
Leecode Java: Min Stack 三种解法
这个题可以 1. 自己写一个Stack，每个Node里面存着对应的min 2. 用java的api，一个Stack存value，一个存min（CC3.2）优化：减少存储重复值min值，因为： POP logic IF min == nod...

```java
 public void remove(Node n) {
  Node pre = n.prev;
  Node post = n.next;
  pre.next = post;
  post.prev = pre;
  n.next = null;
  n.prev = null;
  size--;
 }
}

int capacity;
HashMap<Integer, Node> hm;
DoublyLL dl;
   public LRUCache(int capacity) {
       this.capacity = capacity;
 hm = new HashMap<Integer, Node>();
 dl = new DoublyLL();
   }

   public int get(int key) {
       if(hm.containsKey(key)) {
  Node n = hm.get(key);
  dl.setHead(n);
  return n.val;
 }
 return -1;
   }

   public void set(int key, int value) {
 Node n;
 if(hm.containsKey(key)) {
  n = hm.get(key);
  n.val = value;
  dl.setHead(n);
 } else {
  n = new Node(key, value);
  hm.put(key, n);
  dl.setHead(n);
  if(dl.size>capacity) {
   Node t = dl.removeTail();
   if(t!=null) hm.remove(t.key);
  }
 }
   }
   public void printLL() {
    Node cur = dl.preHead.next;
    Node postTail = dl.postTail;
```



Binary Tree Upside Down

Binary Tree Upside Down 必须画图，然后可以发现，作为任何的node， all left children nodes are roots. n = n.left, then, n.left=parent.right; n.right=parent...

Leetcode 题解： Pow(x, n) Smart and Cool Solution

[Solution] Must handle negative x, n [naive O(n) Version] x*x*x... or pow(x, n/2)*pow(x, n/2) [Smart Version] public class Solution ...

LeetCode java: Minimum Window Substring

[Key points] One HashMap can represent requires as: char -> Integer i: i>0 need more i==0 require satisfied i<0 go...

LeetCode Java Solution: Populating Next Right Pointers in Each Node I II

1. Iteration(Best)： 横着连就是level order traversal. 通常使用BFS，但是题目中每个node自带next指针，相当于就是自带Queue了。所以，从上向下一层一层的连接即可。每层： a. 记录下一层的开始点 b. 连接各个Nodes...

LeetCode: Simplify Path java 最简版

[Analysis] Use a stack to track path from left to right Substring between '/':   str := "" OR ".": Do nothi...

CONTACT ME

Name

```
      System.out.println("DL: size = "+ dl.size);
      while(cur!=postTail) {
       System.out.println(cur.key + " -> " + cur.val);
       cur = cur.next;
      }
     }

     public static void main(String[] args) {
      LRUCache c = new LRUCache(5);
      c.set(0, 10);
      c.printLL();
      c.set(1, 11);
      c.printLL();
      c.set(2, 12);
      c.printLL();
      c.set(3, 13);
      c.printLL();
      c.set(4, 14);
      c.printLL();
      c.set(5, 15);
      c.printLL();
      c.set(2, 22);
      c.printLL();
      c.set(3, 33);
      c.printLL();
      System.out.println( c.get(0) );
      c.printLL();
      System.out.println( c.get(3) );
      c.printLL();
      System.out.println( c.get(5) );
      c.printLL();
      c.set(2, 42);
      c.printLL();
     }
}
```

Posted by Ethan Leo at 2:41 PM Recommend this on Google

No comments :

Post a Comment

Email *

Message *

Send

BLOG ARCHIVE

- ▶ 2015 (2)
- ▼ 2014 (51)
  - ▶ December (4)
  - ▼ November (16)
    - LeetCode in Java: Word Break II DP/暴力过大集合
    - LeeCode Java: Palindrome Partitioning II: 一维DP
    - LeetCode Java Solution: Populating Next Right Poin...
    - LeetCode Java Solution: Jump Game II, One pass
    - LeetCode java solution: Subsets II, without/with h...
    - LeetCode java: Wildcard Matching
    - Leetcode Java: Search in Rotated Sorted Array I II...
    - LeetCode Java: Find Minimum in Rotated Sorted Arra...
    - LeetCode Java: Text Justification - Elegant

Enter your comment...

Comment as: Select profile...

Publish Preview

Soluti...

Leecode Java: Min Stack 三种解法

LeetCode: Divide Two Integers

LeetCode java: Minimum Window Substring

The Hardest LeetCode question: Word Ladder II

LeetCode: Simplify Path java 最简版

Leetcode: Regular Expression Matching

LRU Cache: Java O(1) solution with some test cases...

- ► October (6)
- ► September (8)
- ► August (1)
- ► July (6)
- ► June (10)

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)