课程尚未开始
请大家耐心等待

关注微信公共账号，获得最新面试题信息及解答

Facebook: http://www.facebook.com/ninechapter

Weibo: http://www.weibo.com/ninechapter

Renren: http://page.renren.com/601712402

7. Data Structure

九章算法IT求职面试培训 第7章

www.ninechapter.com

Definition

Data Structure is a way to organize data. It provides some methods to handle data stream, e.g. insert, delete, etc.

Outline

Linear Data Structure

- Queue
- Stack
- Hash

Tree Data Structure

- Heap
- Trie

Queue

Queue (first in first out)

Operations:

- O(1) Push
- O(1) Pop
- O(1) Top

Always used for BFS

Stack

Stack (first in last out)

Operations:

- Push O(1)
- Pop O(1)
- Top O(1)

Min-Stack

Implement a stack, enable O(1) Push, Pop, Top, Min. Where Min() will return the value of minimum number in the stack.

http://lintcode.com/en/problem/min-stack/

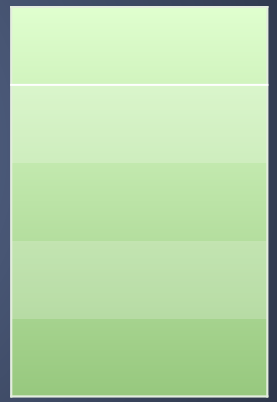
Min-Stack

Using two stacks.
The first one is the regular stack.
The second one only store minimum numbers if a smaller number comes.

http://answer.ninechapter.com/solutions/min-stack/

Push 7,9, 10, 2, 1

stack



minStack

Push 7,9, 10, 2, 1

stack

minStack

Minstack 始终记录到当前push操作的最小数。

Min-Stack

Push(x)

1. stack.push(x)
2. minStack.push(max(现在栈顶元素, 新加入元素))

Pop()

1. stack.pop()
2. minStack.pop()

Implement a queue by two stacks

Q.Push(x): S1-Push(x)
Q.Pop(): if S2.empty() --> S1->S2; S2.pop()
Q.Top(): Similar with Q.Pop()

http://lintcode.com/en/problem/implement-queue-by-stacks/

First in last out

7,9,10,2,1

stack

1,2,10,9,7

First in last out

Implement a queue by two stacks

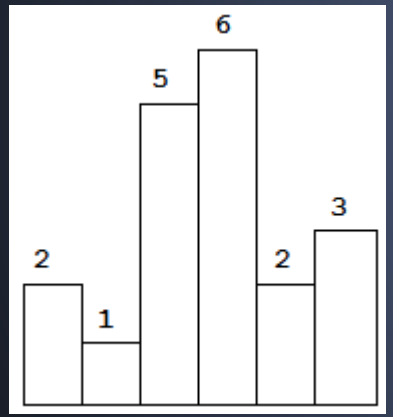
Implement a queue by two stacks. Support O(1) push, pop, top.

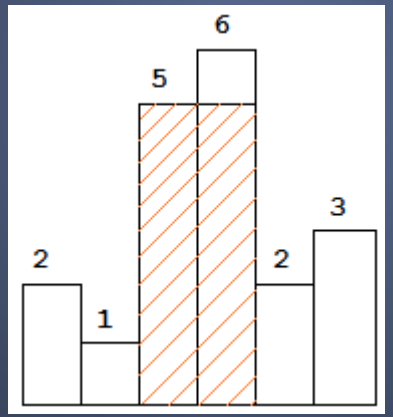
明白栈反序输出的本质first in last out

Largest Rectangle in histogram

http://lintcode.com/en/problem/largest-rectangle-in-histogram/
http://answer.ninechapter.com/solutions/largest-rectangle-in-histogram/

Largest rectangle in histogram



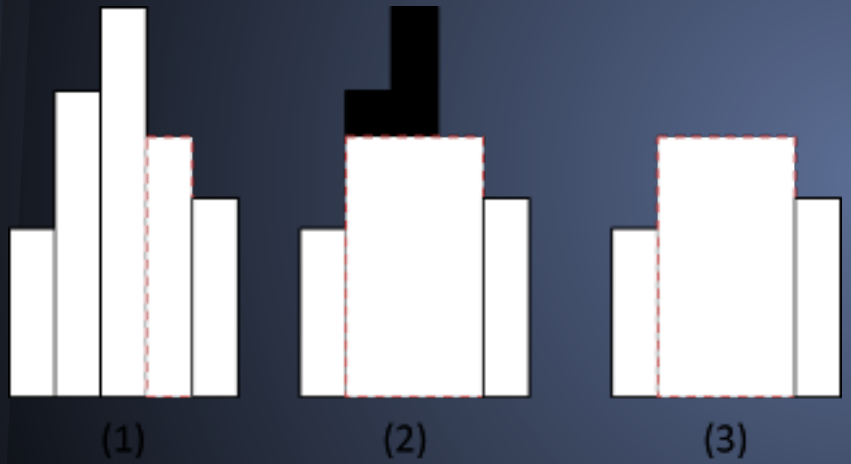


1. 枚举宽度. O (n^2) 有无用的状态

2. 枚举高度

1. 什么时候push
2. 什么时候pop
3. 什么时候计算面积

Largest rectangle in histogram



Maintain an incremental stack:
(1) if a[i] > stack top:
push a[i] to stack
(2) if a[i] <= stack top:
keep poping element out from stack until the top of stack is smaller than current.

时间复杂度 O(n) or O(n^2) ?

数据结构的复杂度:均摊时间复杂度

Construct MaxTree

http://lintcode.com/en/problem/max-tree/
Given an array of integers. Define a MaxTree on this array which the root is the maximum number in the array, the left subtree is the MaxTree represent by the left subarray of the maximum number, and so is right subtree.
Construct a MaxTree by a given integer array in O(n) time and space.

Example: [2 1 5 6 0 3]

Hash

Hash

- Operations
  - Insert - O(1)
  - Delete - O(1)
  - Find - O(1)
- Hash Function
- Collision
  - Open Hashing (LinkedList)
  - Closed Hashing (Array)

Hash Function

Typical: From string to int.

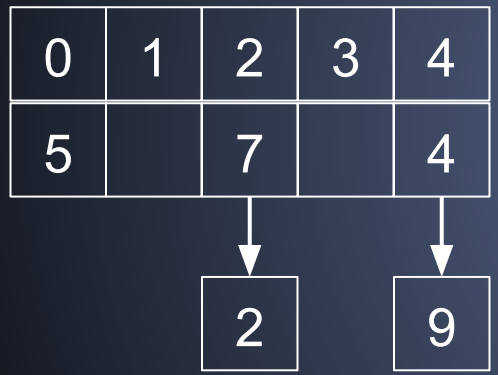
```
int hashfunc(String key) {
    // do something to key
    // return a deterministic integer number
    return md5(key) % hash_table_size;
}
```

APR hashfunc - Magic Number 33

```
int hashfunc(String key) {
    int sum = 0;
    for (int i = 0; i < key.length(); i++) {
        sum = sum * 33 + (int)(key.charAt(i));
        sum = sum % HASH_TABLE_SIZE;
    }
    return sum
}
```

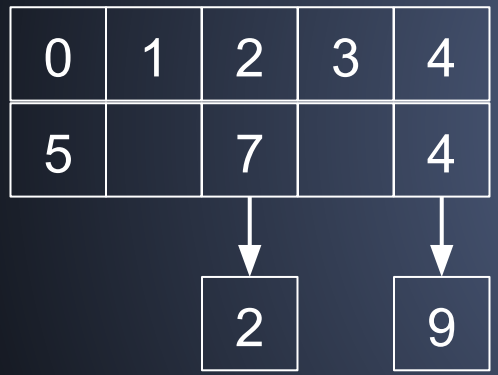
Collision - Open Hashing vs Closed Hashing

Insert: 5 7 4 2 9



Collision - Open Hashing vs Closed Hashing

Insert: 5 7 4 2 9

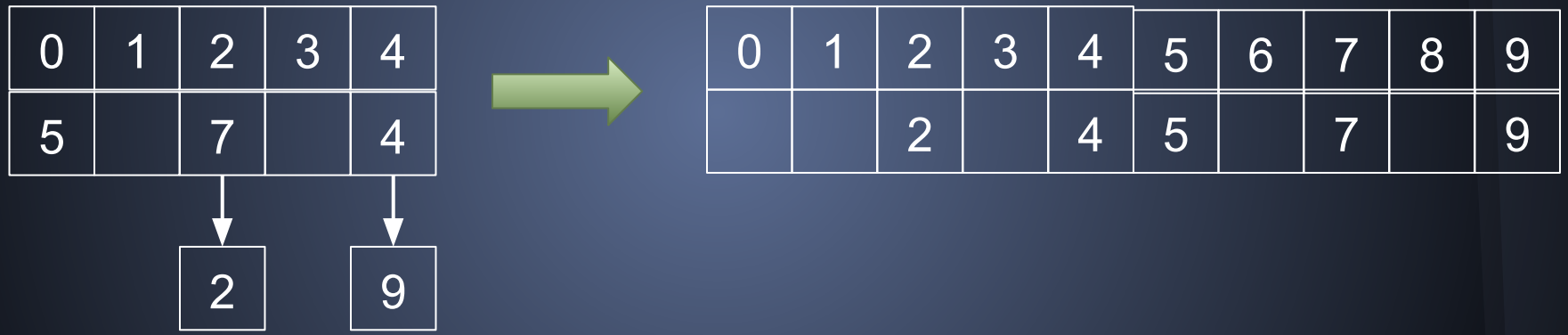


0	1	2	3	4
5	9	7	2	4

Rehashing

Rehashing

Insert: 5 7 4 2 9



Java

What’s Differences of:

- HashTable
- HashSet
- HashMap

Which one is thread Safe?

LRU Cache

http://lintcode.com/en/problem/lru-cache/

http://answer.ninechapter.com/solutions/lru-cache/

Example: [2 1 3 2 5 3 6 7]

LRU Cache

为什么不用Queue？

需求：
1.删除头的元素
2.尾巴加一个元素
3.删除中间的元素
4.元素按照时间排序
5.快速查找一个元素

LRU Cache

LinkedHashMap = DoublyLinkedList + HashMap

```
HashMap<key, DoublyListNode>
DoublyListNode {
    prev, next, key, value;
}
```

Newest node append to tail.

Eldest node remove from head.

Longest Consecutive Sequence

http://lintcode.com/en/problem/longest-consecutive-sequence/http://answer.ninechapter.com/solutions/longest-consecutive-sequence/

Heap

Priority Queue, heapq

Heap

Operations

- Add O(log N)
- Remove O(log N)
- Min/Max O(1)

Heap - Implementation

Low level data structure: Dynamic Array

```
Heap {
    elems[], size;
}
```

elems[1] - root, also the minimum elem in elems.

i's left child: i*2, right child: i*2+1

Internal Method:

siftup, siftdown

Heap - Implementation

Add:

Push back to elems; size ++; Siftup;

Remove:

Replace the elem to be removed with the last elem (elems[size]); size --; Siftup and Siftdown.

Median Number

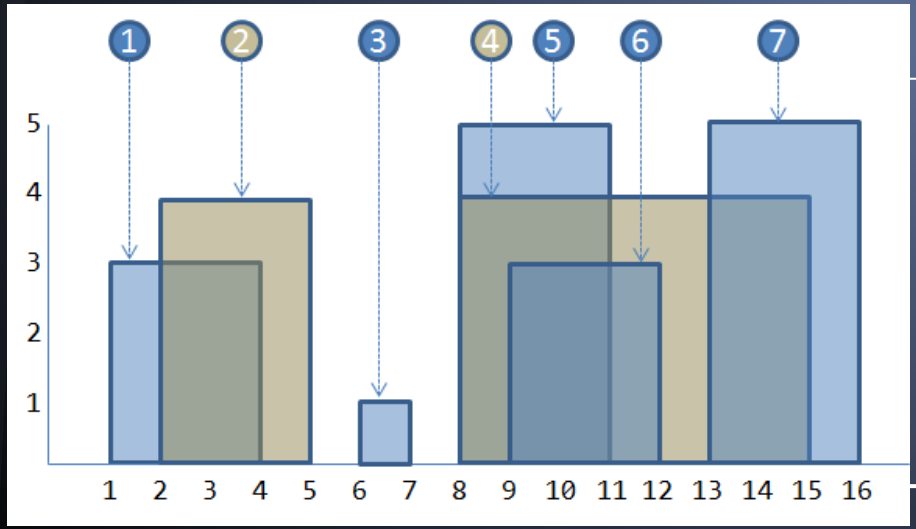
http://lintcode.com/en/problem/median/

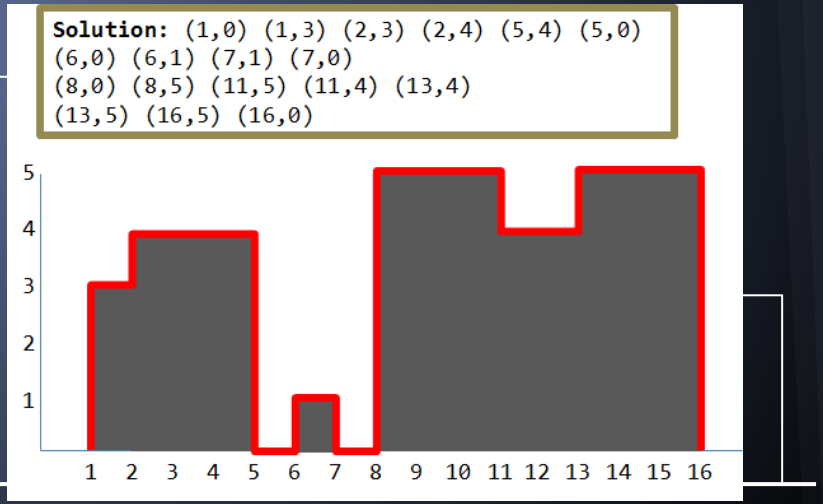
Numbers keep coming, return the median number

[2 1 3 2 5 3 6 7]

Building Outline

Given n buildings, each building is an rectangle located on x-axis, and indicated by (x1, x2, height). Calculate the outline of all buildings. Output them in order.



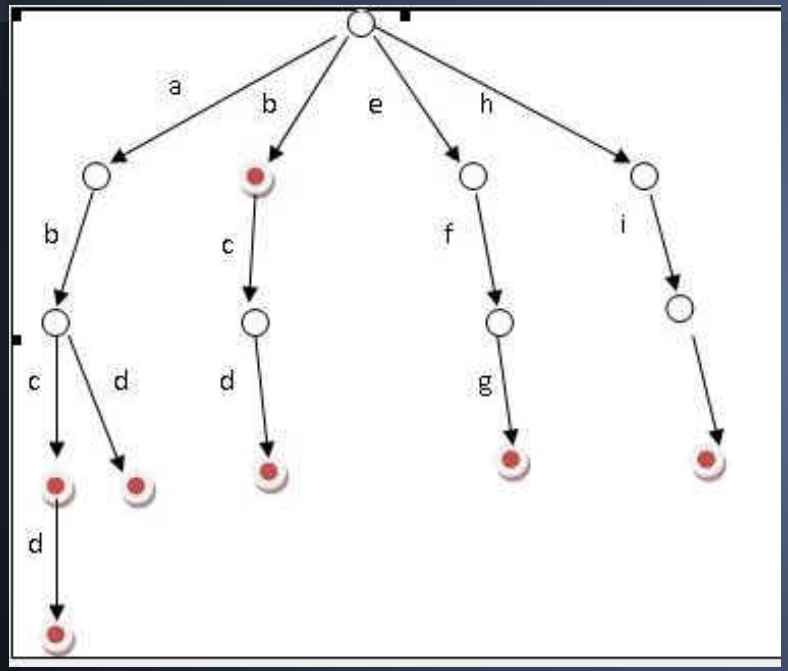


Swiping Line Algorithm

Separate (x1, x2, height) to (x1, height, BUILDING_START), (x2, height, BUILDING_END)
Sort all items by x's ascending order.
Swipe the items from left to right, keep a max heap store heights, when meet a BUILDING_START item, insert the height into the heap, when meet a BUILDING_END item, delete the height in the heap.
The max height in the heap is height in outline with current x you meet.

Trie

假设有[b, abc, abd, bcd, abcd, efg, hii]这6个单词，查找abc 在不在字典里面



Hash vs Trie

时间复杂度Hash O(1) 是对于一个字符串

Trie:

1.一个一个字符串遍历

2. 节约空间

Word Search II

Word Search II

Given a matrix of upper alphabets. e.g.
ACAF
ACAD
ACAE
and a dictionary[aca, acc]. Find all words in the dictionary that can be found in the matrix.