8. Data Structure

九章算法IT求职面试培训 第8章

www.jiuzhang.com

Outline

Linear Data Structure

- Queue
- Stack
- Hash

Tree Data Structure

- Heap
- Trie

Definition

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

Queue

Queue

Operations:

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

Always used for BFS

Stack

Stack

Operations:

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

Min-Stack

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

http://www.jiuzhang.com/solutions/min-stack/

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

Min-Stack

Push(x)

1. stack.push(x)
2. 如果<=minStack的最小值，那么就minStack.push(x)

Pop()

1. stack.pop()
2. 如果==minStack.top()，那么就minStack.pop()

Implement a queue by two stacks

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

http://www.jiuzhang.com/solutions/implement-queue-by-two-stacks/

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

Implement a queue by two stacks

Q.Push(x): S1-Push(x)

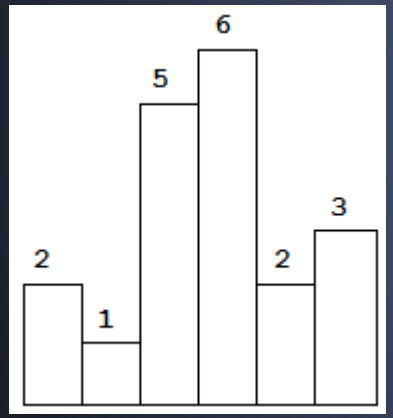
Q.Pop(): if S2.empty() --> S1->S2; S2.pop()

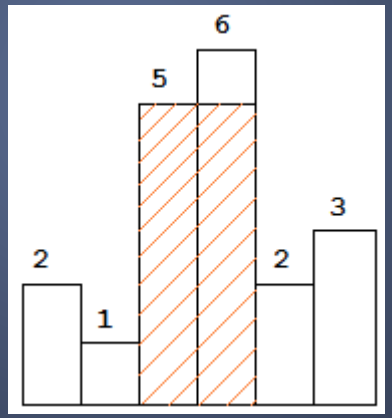
Q.Top(): Similar with Q.Pop()

Largest Rectangle in histogram

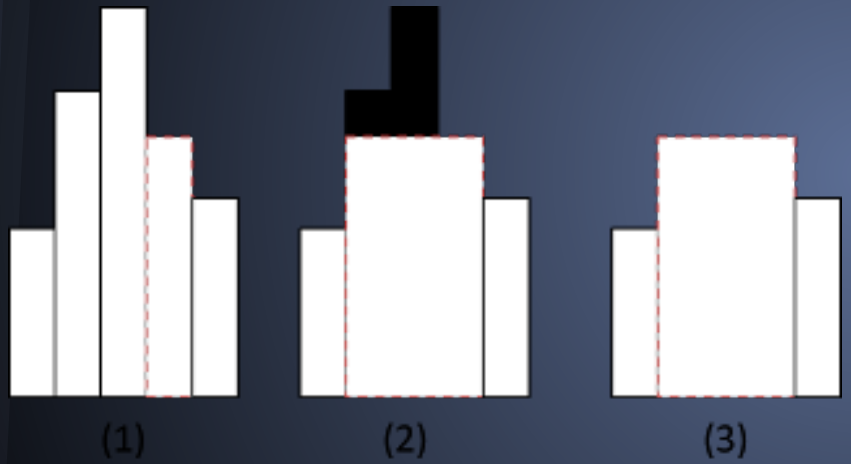
http://www.lintcode.com/en/problem/largest-rectangle-in-histogram/
http://www.jiuzhang.com/solutions/largest-rectangle-in-histogram/

Largest rectangle in histogram





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.

Max Tree

http://www.lintcode.com/en/problem/max-tree/
http://www.jiuzhang.com/solutions/max-tree/

5 min break

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.

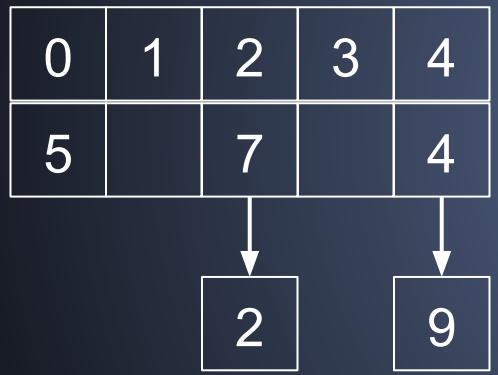
```
int hashfunc(String key) {
    // do something to key
    // return a deterministic integer number
    return md5(key) % hash_table_size;
}
```

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



0	1	2	3	4
5	9	7	2	4

Rehashing

http://www.lintcode.com/en/problem/rehashing/

Java

What’s Differences of:

- HashTable
- HashSet
- HashMap

Which one is thread Safe?

LRU Cache

http://www.lintcode.com/zh-cn/problem/lru-cache/
http://www.jiuzhang.com/solutions/lru-cache/

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

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://www.lintcode.com/zh-cn/problem/longest-consecutive-sequence/
http://www.jiuzhang.com/solutions/longest-consecutive-sequence/

Hash Related Questions

http://www.lintcode.com/en/problem/subarray-sum/

http://www.lintcode.com/en/problem/copy-list-with-random-pointer/

http://www.lintcode.com/en/problem/anagrams/

Heap

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://www.lintcode.com/en/problem/data-stream-median/

Heap Related Questions

http://www.lintcode.com/en/problem/heapify/

http://www.lintcode.com/en/problem/merge-k-sorted-lists/

Trie

Word Search II

http://www.lintcode.com/en/problem/word-search-ii/
http://www.jiuzhang.com/solutions/word-search-ii/

Word Search II

Given a matrix of upper alphabets. e.g.

ACAF

ACAD

ACAE

and a dictionary. Find all words in the dictionary that can be found in the matrix.