哈希表与堆
Hash & Heap

课程版本 v4.2　　主讲 令狐冲

扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: ninechapter
微博: http://www.weibo.com/ninechapter
知乎: http://zhuanlan.zhihu.com/jiuzhang
官网: http://www.jiuzhang.com

大纲 Outline

- 哈希表 Hash
  - 原理
  - 应用
- 堆 Heap
  - 原理：小视频
  - 应用：优先队列 Priority Queue
  - 替代品：TreeMap

What is Data Structure?

可以认为是一个集合，并且提供集合上的若干操作

队列 Queue

支持操作：O(1) Push / O(1) Pop / O(1) Top

BFS的主要数据结构

多做做BFS的题就可以了

栈 Stack

支持操作：O(1) Push / O(1) Pop / O(1) Top

非递归实现DFS的主要数据结构

独孤九剑 —— 破箭式

BFS 的主要数据结构是 Queue

DFS 的主要数据结构是 Stack

千万不要搞反了！很体现基础知识的扎实度！

哈希表 Hash

支持操作：O(1) Insert / O(1) Find / O(1) Delete

Hash Table / Hash Map / Hash Set 的区别是什么？

Hash Function

使命：对于任意的key

得到一个*固定*且*无规律*的介于*0~capacity-1*的整数

Hash Function

- 一些著名的Hash算法
  - MD5
  - SHA-1
  - SHA-2

```
int hashfunc(String key) {
    return md5(key) % hash_table_size;
}
```

- 以 String 为例子

```
int hashfunc(String key) {
    int sum = 0;
    for (int i = 0; i < key.length(); i++) {
        sum = sum * 31 + (int)(key.charAt(i));
        sum = sum % HASH_TABLE_SIZE;
    }
    return sum;
}
```

Magic Number - 31

经验值

这个数字选择一个质数会更好(经验)

数太大 --- 影响计算速度

数太小 --- 冲突太多

像 Apache 的底层库中，用的是 33

Open Hashing vs Closed Hashing

再好的 hash 函数也会存在冲突(Collision)

https://www.cs.usfca.edu/~galles/visualization/ClosedHash.html

https://www.cs.usfca.edu/~galles/visualization/OpenHash.html

Rehashing

当hash不够大时怎么办？

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

http://www.jiuzhang.com/solutions/rehashing/

哈希表的饱和度

饱和度 = 实际存储元素个数 / 总共开辟的空间大小

size / capacity

一般来说，超过 1/10（经验值）的时候，说明需要进行 rehash

LRU Cache

http://www.lintcode.com/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.

问：Singly List 是否可行？

Singly List 是否可行？

可以，在 Hash 中存储 Singly List 中的 prev node 即可

如 linked list = dummy->1->2->3->null 时

hash[1] = dummy, hash[2] = node1 ...

Related Questions

- http://www.lintcode.com/problem/subarray-sum/
- http://www.lintcode.com/problem/copy-list-with-random-pointer/
- http://www.lintcode.com/problem/anagrams/
- http://www.lintcode.com/problem/longest-consecutive-sequence/

休息 5 分钟

总结一道题的经验，胜过刷十道题
把你的代码和总结发到九章面试题交流社区
www.jiuzhang.com/solutions

Heap

支持操作：O(log N) Add / O(log N) Remove / O(1) Min or Max

Max Heap vs Min Heap

PriorityQueue vs Heap

Heap 的基本原理和具体实现

请见课程小视频

http://www.jiuzhang.com/video/heap

Ugly Number

http://www.lintcode.com/problem/ugly-number-ii/

http://www.jiuzhang.com/solutions/ugly-number-ii/

Top k Largest Number II

http://www.lintcode.com/problem/top-k-largest-numbers-ii/

http://www.jiuzhang.com/solutions/top-k-largest-number-ii/

Merge K Sorted Lists

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

http://www.jiuzhang.com/solution/merge-k-sorted-lists/

三种方法，都需要练习

方法一：使用 PriorityQueue

方法二：类似归并排序的分治算法

方法三：自底向上的两两归并算法

时间复杂度均为 O(NlogK)

Related Questions

- http://www.lintcode.com/en/problem/high-five/ (A)
- http://www.lintcode.com/en/problem/k-closest-points/ (L, A, F)
- http://www.lintcode.com/problem/merge-k-sorted-arrays/
- http://www.lintcode.com/problem/data-stream-median/
- http://www.lintcode.com/problem/top-k-largest-numbers/
- http://www.lintcode.com/problem/kth-smallest-number-in-sorted-matrix/

TreeMap（optional）

又想知道最小值，又想支持修改和删除

https://docs.oracle.com/javase/7/docs/api/java/util/TreeMap.html

通常来说，面试中几乎没有必须要用 TreeMap 的题

相关习题

- http://www.lintcode.com/problem/building-outline/
- http://www.lintcode.com/problem/top-k-frequent-words/