

LintCode领扣题解 (/problem) / 重哈希 · Rehashing

重哈希 · Rehashing

中文

LintCode 版权所有 (/problem/?tags=lintcode-copyright)

hash table (/problem/?tags=hash-table)

描述

哈希表容量的大小在一开始是不确定的。如果哈希表存储的元素太多（如超过容量的十分之一），我们应该将哈希表容量扩大一倍，并将所有的哈希值重新安排。假设你有如下哈希表：

```
size=3, capacity=4

[null, 21, 14, null]
  ↓   ↓
  9   null
  ↓
 null
```

哈希函数为：

```
int hashCode(int key, int capacity) {
    return key % capacity;
}
```

这里有三个数字9，14，21，其中21和9共享同一个位置因为它们有相同的哈希值1(21 % 4 = 9 % 4 = 1)。我们将它们存储在同一个链表中。

重建哈希表，将容量扩大一倍，我们将会得到：

```
size=3, capacity=8

index:  0    1    2    3    4    5    6    7
hash : [null, 9, null, null, null, 21, 14, null]
```

给定一个哈希表，返回重哈希后的哈希表。

❶ 哈希表中负整数的下标位置可以通过下列方式计算： - **C++/Java**：如果你直接计算-4 % 3，你会得到-1，你可以应用函数：a % b = (a % b + b) % b得到一个非负整数。 - **Python**：你可以直接用-1 % 3，你可以自动得到2。

样例

样例 1

```
输入 : [null, 21->9->null, 14->null, null]
输出 : [null, 9->null, null, null, null, 21->null, 14->null, null]
```

在线评测地址: <https://www.lintcode.com/problem/rehashing/> (<https://www.lintcode.com/problem/rehashing/>)

收起题目描述 ^

语言类型

ALL (13)

python (5)

cpp (5)

java (3)

上传题解

令狐冲

更新于 12/30/2020, 8:30:09 PM

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    def addlistnode(self, node, number):
        if node.next != None:
            self.addlistnode(node.next, number)
        else:
            node.next = ListNode(number)

    def addnode(self, anshashTable, number):
        p = number % len(anshashTable)
        if anshashTable[p] == None:
            anshashTable[p] = ListNode(number)
        else:
            self.addlistnode(anshashTable[p], number)

    def rehashing(self, hashTable):
        HASH_SIZE = 2 * len(hashTable)
        anshashTable = [None for i in range(HASH_SIZE)]
        for item in hashTable:
            p = item
            while p != None:
                self.addnode(anshashTable, p.val)
                p = p.next
        return anshashTable
```

👍 获赞 4

💬 1 条评论

你的口袋题库

2000+算法真题、国内外名企题库免费开放



九章算法APP

令狐冲

更新于 9/3/2020, 11:19:29 PM

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
/**
 * Definition for ListNode
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) {
 *         val = x;
 *         next = null;
 *     }
 * }
 */
public class Solution {
    /**
     * @param hashTable: A list of The first node of linked list
     * @return: A list of The first node of linked list which have twice size
     */
    public ListNode[] rehashing(ListNode[] hashTable) {
        // write your code here
        if (hashTable.length <= 0) {
            return hashTable;
        }
        int newcapacity = 2 * hashTable.length;
        ListNode[] newTable = new ListNode[newcapacity];
        for (int i = 0; i < hashTable.length; i++) {
            while (hashTable[i] != null) {
                int newIndex
                    = (hashTable[i].val % newcapacity + newcapacity) % newcapacity;
                if (newTable[newIndex] == null) {
                    newTable[newIndex] = new ListNode(hashTable[i].val);
                    // newTable[newIndex].next = null;
                } else {
                    ListNode dummy = newTable[newIndex];
                    while (dummy.next != null) {
                        dummy = dummy.next;
                    }
                    dummy.next = new ListNode(hashTable[i].val);
                }
                hashTable[i] = hashTable[i].next;
            }
        }
        return newTable;
    }
}
```

👍 获赞 2

💬 8 条评论



令狐冲

更新于 9/3/2020, 11:24:54 PM

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution {
public:
    void addlistnode(ListNode* node, int number) {
        if (node->next != NULL)
            addlistnode(node->next, number);
        else
            node->next = new ListNode(number);
    }
    void addnode(vector<ListNode*> &anshashTable, int number) {
        int p = (number + anshashTable.size()) % anshashTable.size();
        if (anshashTable[p] == NULL)
            anshashTable[p] = new ListNode(number);
        else
            addlistnode(anshashTable[p], number);
    }
    vector<ListNode*> rehashing(vector<ListNode*> hashTable){
        vector<ListNode*> anshashTable;
        for(int i = 0; i < hashTable.size() * 2; i++)
            anshashTable.push_back(NULL);
        int HASH_SIZE = anshashTable.size();
        for(int i = 0; i < hashTable.size(); i++) {
            ListNode* p = hashTable[i];
            while (p != NULL) {
                addnode(anshashTable, p->val);
                p = p->next;
            }
        }
        return anshashTable;
    }
};
```

👍 获赞 0

💬 3 条评论

**Tin**

更新于 10/25/2020, 6:31:15 PM

本来很简单的一道题, 但是有隐藏要求, 即后到的node要赛到链表后面, 这要求很无脑, 只是为了OJ容易写而已。

凡标了Lintcod版权的题, 都没有经过深思熟虑, 要求不周全。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
class Solution:

    def rehashing(self, hashTable):

        if not hashTable: return []
        H_SIZE = len(hashTable) * 2
        new_table = [None] * H_SIZE

        for head in hashTable:
            while head:
                new_slot = head.val % H_SIZE
                node = ListNode(head.val, new_table[new_slot])
                new_table[new_slot] = node
                dummy, tail = node.next, node.next
                while tail and tail.next:
                    tail = tail.next
                if dummy:
                    tail.next, node.next = node, None
                    new_table[new_slot] = dummy

            head = head.next
        return new_table
```

👍 获赞 4

💬 添加评论



J.

更新于 9/2/2020, 3:06:28 PM

1. 算扩容之后的 size ($\text{origin_len} * 2$)
2. 遍历原来的 hashtable 和每个 listnode
3. 重算在 new_hashtable 的 index
4. 在新的位置已经有 node 就到 tails 里面找 tail 串, 否则就初始化 heads[i] 和 tails[i]

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
"""
Definition of ListNode
class ListNode(object):

    def __init__(self, val, next=None):
        self.val = val
        self.next = next
"""

class Solution:
    """
    @param hash_table: A list of The first node of linked list
    @return: A list of The first node of linked list which have twice size
    """
    def rehashing(self, hash_table):
        if not hash_table:
            return

        CAPACITY = len(hash_table) * 2
        heads = [None] * CAPACITY
        tails = [None] * CAPACITY

        curr = _node = i = None
        for node in hash_table:
            curr = node

            while curr:
                i = curr.val % CAPACITY
                _node = ListNode(curr.val)

                if heads[i]:
                    tails[i].next = _node
                else:
                    heads[i] = _node

                tails[i] = _node

                curr = curr.next

        return heads
```

👍 获赞 3

💬 添加评论



九章用户7865T1

更新于 10/4/2020, 11:10:13 PM

由于lintcode要求insert在list的末尾, 为了不walk through整个list来找到list的最后一个, 可以直接用一个 ListNode[] tail 来记录新table每个bucket的tail node.

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm\_source=code
 */
public ListNode[] rehashing(ListNode[] hashTable) {
    int cap = 2 * hashTable.length;
    ListNode[] table = new ListNode[cap], tail = new ListNode[cap];
    for (int i = 0; i < hashTable.length; i++) {
        while (hashTable[i] != null) {
            ListNode cur = hashTable[i];
            int hashCode = (cur.val + cap) % cap;
            hashTable[i] = cur.next;
            // tail不为null, 直接insert在末尾
            if (tail[hashCode] != null) {
                tail[hashCode].next = cur;
            } else {
                // tail为null, 说明这个node是这个bucket的第一个node
                table[hashCode] = cur;
            }
            tail[hashCode] = cur;
            cur.next = null;
        }
    }
    return table;
}
```

👍 获赞 2

💬 1 条评论

**九章用户X4JYL6**

更新于 6/9/2020, 7:03:52 AM

time complexity: $O(n)$ ans存每个slot的起始node buildingAns存每个slot的尾巴node

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
"""
Definition of ListNode
class ListNode(object):

    def __init__(self, val, next=None):
        self.val = val
        self.next = next
"""

class Solution:
    """
    @param hashTable: A list of The first node of linked list
    @return: A list of The first node of linked list which have twice size
    """
    def rehashing(self, hashTable):
        # write your code here
        cap=len(hashTable)*2
        ans=[None]*cap
        buildingAns=[None]*cap
        for node in hashTable:
            while node:
                key=node.val%cap
                newNode=ListNode(node.val)
                if ans[key]:
                    tail=buildingAns[key]
                    tail.next=newNode
                else:
                    ans[key]=newNode
                    buildingAns[key]=newNode
                node=node.next
        return ans
```

👍 获赞 1

💬 添加评论

**HUE**

更新于 7/30/2020, 9:08:22 PM

1. 算出新哈希表的 size, 並 new 出一個新的哈希表
2. 把原哈希表中的每個 node "複製" 一份插入新的哈希表的新位置
3. 如果新位置還沒有其他人, 直接插入
4. 若新位置已有其他人, 則找到該位置鏈表的最後一個 node, 把當前 node 接在它後面


```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
/**
 * Definition of ListNode
 * class ListNode {
 * public:
 *     int val;
 *     ListNode *next;
 *     ListNode(int val) {
 *         this->val = val;
 *         this->next = NULL;
 *     }
 * }
 */
class Solution {
public:
    /**
     * @param hashTable: A list of The first node of linked list
     * @return: A list of The first node of linked list which have twice size
     */
    vector<ListNode*> rehashing(vector<ListNode*> hashTable) {
        // write your code here
        int curSize = hashTable.size();
        int newSize = curSize * 2;

        vector<ListNode*> newHashTable (newSize, NULL);
        for (ListNode* node : hashTable) {
            while (node != NULL) {
                int newPos = (node->val % newSize + newSize) % newSize;
                if (newHashTable[newPos] == NULL) {
                    newHashTable[newPos] = new ListNode(node->val);
                }
                else {
                    ListNode* dummy = newHashTable[newPos];
                    while (dummy->next != NULL) {
                        dummy = dummy->next;
                    }
                    dummy->next = new ListNode(node->val);
                }
                node = node->next;
            }

            return newHashTable;
        }
    }
};
```

👍 获赞 0 💬 添加评论



feary

更新于 6/9/2020, 7:04:24 AM

我的理解这只是数据结构扩容, 不该重新new一个节点, 而是应该用原来的结构只是重组下位置。

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
/**
 * Definition of ListNode
 * class ListNode {
 * public:
 *     int val;
 *     ListNode *next;
 *     ListNode(int val) {
 *         this->val = val;
 *         this->next = NULL;
 *     }
 * }
 */
class Solution {
public:
    /**
     * @param hashTable: A list of The first node of linked list
     * @return: A list of The first node of linked list which have twice size
     */
    vector<ListNode*> rehashing(vector<ListNode*> hashTable) {
        // write your code here
        int n = hashTable.size() * 2;
        vector<ListNode*> re(n, NULL);
        for(auto it = hashTable.begin(); it != hashTable.end(); it++){
            if(*it != NULL){
                auto tem = *it;
                while(tem){
                    auto h = tem;
                    tem = tem->next;
                    h->next = NULL;
                    auto f = re[(h->val + n) % n];
                    if(f){
                        auto l = f;
                        f = f->next;
                        while(f){
                            l = f;
                            f = f->next;
                        }
                        l->next = h;
                    }else{
                        re[(h->val + n) % n] = h;
                    }
                }
            }
        }
        return re;
    }
};
```

👍 获赞 0

💬 添加评论

**felix**

更新于 6/9/2020, 7:04:20 AM

不同于答案的不断加到list tail,类似于教程中不断加到linkedlist队首, 但是不被接受! 不知道是因为有其他bug吗

```
/**
 * 本参考程序由九章算法用户提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括：九章算法班 2020升级版，算法强化班，算法基础班，北美算法面试高频题班，Java 高级工程师 P6+ 小班课，面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括：系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括：动态规划专题班，Big Data - Spark 项目实战，Django 开发项目课
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?utm_source=code
 */
class Solution:
    """
    @param hashTable: A list of The first node of linked list
    @return: A list of The first node of linked list which have twice size
    """
    def rehashing(self, hashTable):

        if hashTable is None:
            return None
        HASH_SIZE = len( hashTable ) * 2
        new_hashTable = [ None ] * HASH_SIZE
        for item in hashTable:
            if not item:
                continue

            tmp = item
            while tmp:
                tmp_next = tmp.next
                hash_code = tmp.val % HASH_SIZE
                tmp.next = new_hashTable[ hash_code ]
                new_hashTable[ hash_code ] = tmp
                tmp = tmp_next
        return new_hashTable
```

👍 获赞 0

💬 添加评论



Gnocchi

更新于 6/9/2020, 7:04:07 AM

往新生成的hashTable里面加node的时候还是需要仔细考虑

```
/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
class Solution {
public:
    /**
     * @param hashTable: A list of The first node of linked list
     * @return: A list of The first node of linked list which have twice size
     */
    vector<ListNode*> rehashing(vector<ListNode*> hashTable) {
        // write your code here

        int tSize = hashTable.size();
        cout << tSize << endl;
        vector<ListNode*> newTable (tSize * 2, nullptr);
        int base = tSize * 2;

        for (int i = 0; i < tSize; i++) {

            ListNode * node = hashTable[i];

            while (node != nullptr){
                int newHashCode = (node->val%base + base)%base;
                if (newTable[newHashCode] == nullptr) {
                    newTable[newHashCode] = node;

                    node = node->next;
                    newTable[newHashCode]->next = nullptr;

                } else {

                    ListNode * curr = newTable[newHashCode];
                    while (curr && curr->next) {
                        curr = curr->next;
                    }

                    curr->next = node;
                    node = node->next;
                    curr->next->next = nullptr;

                }
            }
        }
        return newTable;
    }
};
```

👍 获赞 0

💬 添加评论

**Jet**

更新于 6/9/2020, 7:04:04 AM

Rehashing.Rehashing.Rehashing.

```

/**
 * 本参考程序由九章算法用户提供。版权所有, 转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作, 授课老师均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的求职课程包括: 九章算法班 2020升级版, 算法强化班, 算法基础班, 北美算法面试高频题班, Java 高级工程师 P6+ 小班课, 面试软技能指导 - BQ / Resume / Project 2020版
 * - Design类课程包括: 系统设计 System Design, 面向对象设计 OOD
 * - 专题及项目类课程包括: 动态规划专题班, Big Data - Spark 项目实战, Django 开发项目课
 * - 更多详情请见官方网站: http://www.jiuzhang.com/?utm_source=code
 */
/**
 * Definition of ListNode
 * class ListNode {
 * public:
 *     int val;
 *     ListNode *next;
 *     ListNode(int val) {
 *         this->val = val;
 *         this->next = NULL;
 *     }
 * }
 */
class Solution {
public:
    /**
     * @param hashTable: A list of The first node of linked list
     * @return: A list of The first node of linked list which have twice size
     */
    vector<ListNode*> rehashing(vector<ListNode*> hashTable) {
        // write your code here

        if(hashTable.size()==0){
            return hashTable;
        }
        size_t size=hashTable.size();
        int newsize=2*size;
        vector<ListNode*> result(newsize,nullptr);

        for(int i=0;i<hashTable.size();i++){//iterate hashTable vector
            if(hashTable[i]==nullptr){
                continue;
            }
            ListNode* node=hashTable[i];//iterate hashTable list
            while(node!=nullptr){
                int val = node->val;
                int index = (val%newsize+ newsize)%newsize;
                if(result[index]==nullptr){
                    result[index]=new ListNode(val);//
                    node=node->next;
                    continue;
                }
                ListNode* list=result[index];
                while(list->next!=nullptr){
                    list=list->next;
                }
                list->next=new ListNode(val);
                node=node->next;
            }
        }
        return result;
    }
};

```

👍 获赞 0

💬 添加评论

加载更多题解

进阶课程

直播+互动	直播+互动	直播+互动	互动课
九章算法班 2021 版 8周时间精通 57 个核心高频考点，9 招击破 FLAG、BATJ 算法面试。22....	系统架构设计 System Design 2021 版 成为百万架构师必上。30 课时带你快速掌握18大系统架构设计知识点与面...	九章算法面试高频题冲刺班 每期更新 15% 题目，考前押题，一举拿下FLAG & BATJ Offer	面向对象设计 OOD 应届生及亚马逊面试必考，IT求职必备基础