


```

5
6 public static IntList dcatenate(IntList A, IntList B) {
7     //TODO: fill in method
8     IntList res = A;
9     while (A.rest != null) {
10         A = A.rest;
11     }
12     A.rest = B;
13     return res;
14 }
15
16 /**
17  * Returns a list consisting of the elements of A followed by the
18  * * elements of B. May NOT modify items of A. Use 'new'.
19  */
20 public static IntList catenate(IntList A, IntList B) {
21     //TODO: fill in method
22     IntList res = new IntList(A.first, null);
23     IntList ptr = res;
24     A = A.rest;
25     while (A != null) {
26         ptr.rest = new IntList(A.first, null);
27         ptr = ptr.rest;
28         A = A.rest;
29     }
30     ptr.rest = B;
31     return res;
32 }
--

```

2. SLLists

理论上讲，IntList可以实现一个链表能够做的所有事，不过它难以使用，代码难以理解。最主要的是，如果一个程序员想使用它，就必须了解它复杂，要手动写出next的指向，这会大大降低这种数据结构的使用率。

所以，在原有IntList的基础上，我们设计一种新的类SLLists(Single Linked Lists)，并为它提供一系列的方法。

首先我们创建一个新的类IntNode：

```

1 public class IntNode {
2     public int item;
3     public IntNode next;
4
5     public IntNode(int i, IntNode n) {
6         item = i;
7         next = n;
8     }
9 }

```

除了名字不同以外，它好像和IntList没有任何区别，没错，我们还需要创建另外一个类SLList：

```

1 public class SLList {
2     public IntNode first;
3
4     public SLList(int x) {
5         first = new IntNode(x, null);
6     }
7 }

```

它将IntNode进行了封装，会使用者提供了更简洁的使用方法，当使用IntList和SLList时，你可以清楚的看出SLList优秀的地方：

```

1 IntList L1 = new IntList(5, null);
2 SLList L2 = new SLList(5);

```

SLList隐藏了IntNode中靠next取得的联系，使用者不必关心SLList内部如何实现，只知道如何用它来存储数据就足够了。我们再为SLList添加几个：

```

1 public class SLList {
2     public IntNode first;
3
4     public SLList(int x) {

```

```

5     first = new IntNode(x, null);
6 }
7
8 /** Adds an item to the front of the list. */
9 public void addFirst(int x) {
10     first = new IntNode(x, first);
11 }
12
13 /** Retrieves the front item from the list. */
14 public int getFirst() {
15     return first.item;
16 }
17 }

```

通过比较，你会发现SLList使用起来更简单了：

```

1 SLList L = new SLList(15);
2 L.addFirst(10);
3 L.addFirst(5);
4 int x = L.getFirst();

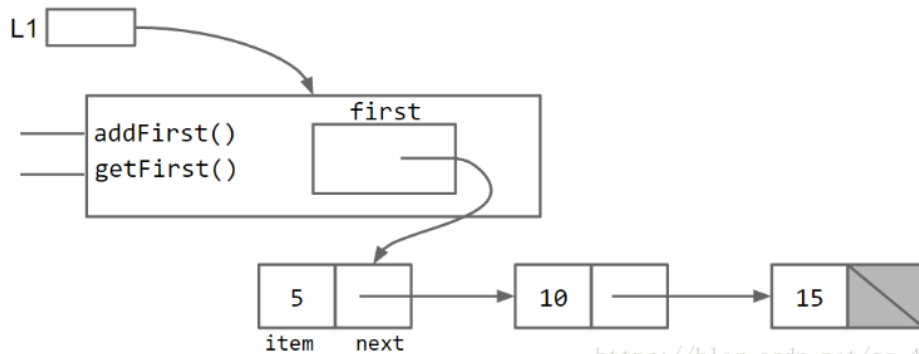
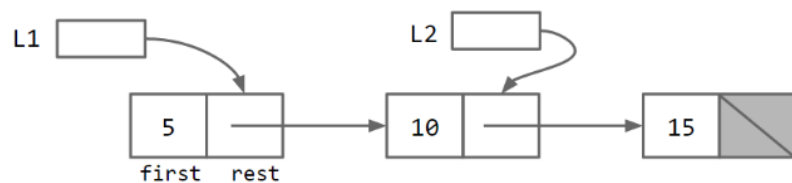
```

```

1 IntList L = new IntList(15, null);
2 L = new IntList(10, L);
3 L = new IntList(5, L);
4 int x = L.first;

```

再来比较一下它们的内部实现：



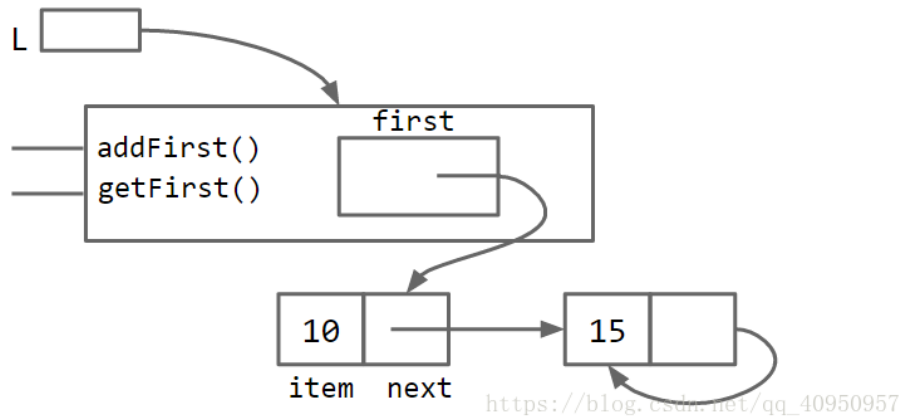
https://blog.csdn.net/qq_40950957

不过，如果使用者这样使用呢？

```

1 SLList L = new SLList(15);
2 L.addFirst(10);
3 L.first.next.next = L.first.next;

```



这将会导致死循环，所以我们要把first声明为私有变量，禁止使用者调用它，这样，上面的操作将会报错。不过这时，我们有两个.java文件，使/都要导入这两个文件，所以，我们可以将IntNode写入SLList类中，这也称为Nested Classes。并且IntNode不会调用它以外的SLList中的值，所以我们static类型，实现如下：

```

1 public class SLList {
2     public static class IntNode {
3         public int item;
4         public IntNode next;
5         public IntNode(int i, IntNode n) {
6             item = i;
7             next = n;
8         }
9     }
10
11     private IntNode first;
12     ...

```

我们再为它添加size方法：

```

1 /** Returns the size of the list starting at IntNode p. */
2 private static int size(IntNode p) {
3     if (p.next == null) {
4         return 1;
5     }
6
7     return 1 + size(p.next);
8 }
9
10 public int size() {
11     return size(first);
12 }

```

然而，size方法内部使用了递归或者迭代，如果长度为1000的链表的size需要耗时2秒，那么计算长度为1,000,000的链表的size就可能会耗时2000秒。要将size方法的时间复杂度设计为常数，这里用到了缓存(caching)的思想，我们添加一个size变量，并时刻跟踪改变它的值：

```

1 public class SLList {
2     ... /* IntNode declaration omitted. */
3     private IntNode first;
4     private int size;
5
6     public SLList(int x) {
7         first = new IntNode(x, null);
8         size = 1;
9     }
10
11     public void addFirst(int x) {
12         first = new IntNode(x, first);
13         size += 1;
14     }

```

```

15
16     public int size() {
17         return size;
18     }
19     ...
20 }

```

这样，无论链表多么长，size方法都会很快的返回它的值。下面我们再仔细思考，如果一个链表为空，我们如何表示它？一种方法是为其设计空参

```

1 public SLList() {
2     first = null;
3     size = 0;
4 }

```

不过它的addLst方法将变得很庞大，而且之后每添加一个类似的方法，我们都需要考虑 **链表为空** 的情况。

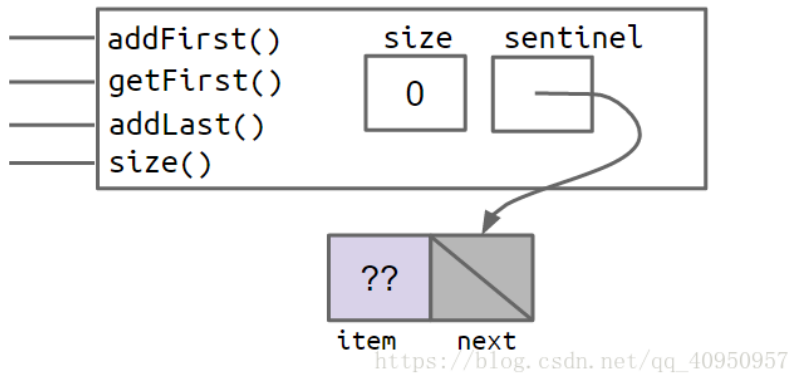
```

1 public void addLast(int x) {
2     size += 1;
3
4     if (first == null) {
5         first = new IntNode(x, null);
6         return;
7     }
8
9     IntNode p = first;
10    while (p.next != null) {
11        p = p.next;
12    }
13
14    p.next = new IntNode(x, null);
15 }

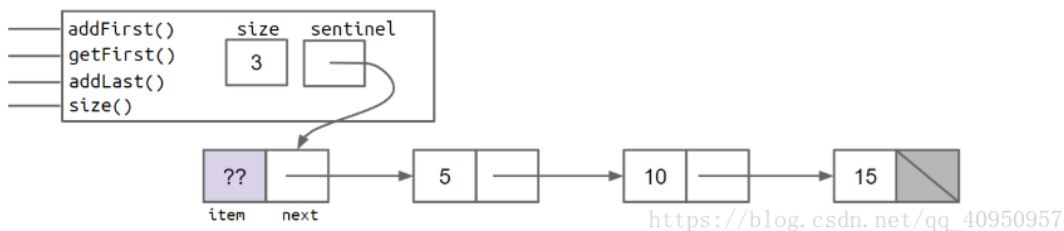
```

所以，在设计伊始，我们设计的模型要在根本上解决这个问题。这里的设计思想一开始可能会难以接受，不过渐渐你会发现它的优秀之处。我们采用**添加一个空的node节点**，称之为sentinel node。

于是，当我们通过 `SLList L = new SLList()` 实例化一个SLList对象时，它的内部将会变成这样：



当为链表添加5，10，15三个元素时，它将变成这样：



这样，addLast方法就变得简洁易读多了。

```

1 public void addLast(int x) {
2     size += 1;
3     IntNode p = sentinel;
4     while (p.next != null) {
5         p = p.next;
6     }
7
8     p.next = new IntNode(x, null);
9 }

```

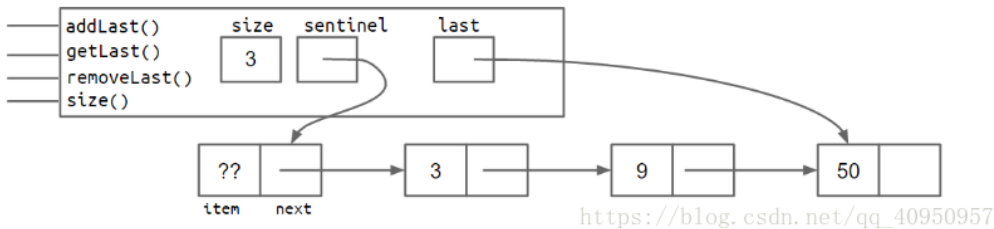
但是，addLst方法同样存在一个问题，就是链表的长度越长，它的执行时间也就越长。为了解决这一问题，我们同样可以使用储存的思想，为它添量，时刻记录尾节点的位置。

```

1 public class SLList {
2     private IntNode sentinel;
3     private IntNode last;
4     private int size;
5
6     public void addLast(int x) {
7         last.next = new IntNode(x, null);
8         last = last.next;
9         size += 1;
10    }
11    ...
12 }

```

它的内部就变成了下面这样：



再考虑，如果我们要添加一个 **移除最后一个元素** 的removeLast方法，该如何实现？大致的思路就是 **获取到倒数第二个节点**，然后将倒数第二个节点的null，可是，**如何获取到倒数第二个节点呢？**

从sentinel向后遍历是不可能的了，那将使方法具有线性复杂度，其实，在现有的模型结构下很难实现，所以，我们不得在现有模型的基础上再次该数据结构，也就是下面的 **DLList**。

3. DLLists

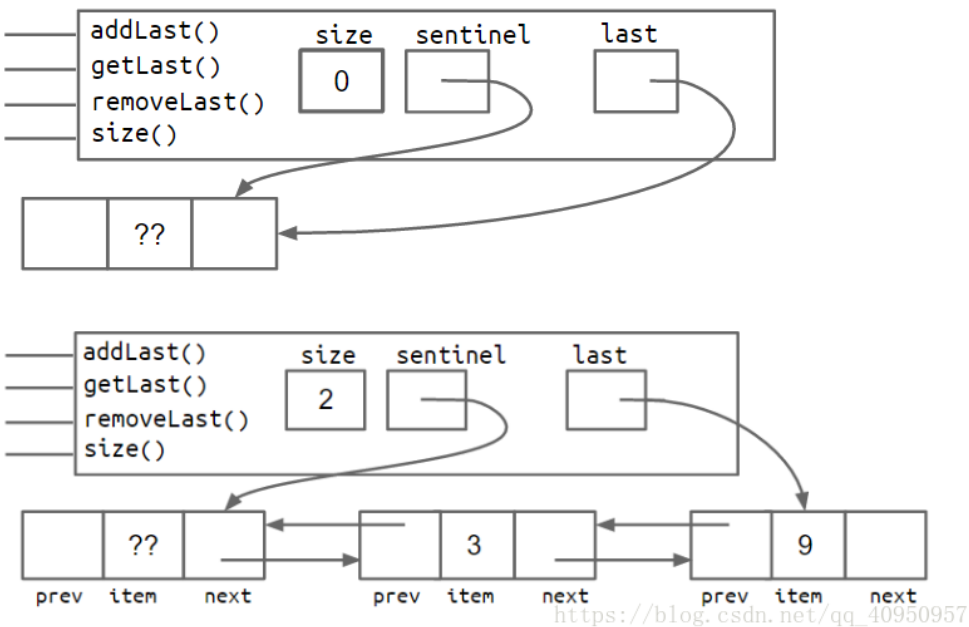
这一次的改造更加彻底，我们在每一个IntNode中 **添加一个prev变量**，也就是为整个链表添加了 **Back pointers**：

```

1 public class IntNode {
2     public IntNode prev;
3     public int item;
4     public IntNode next;
5 }

```

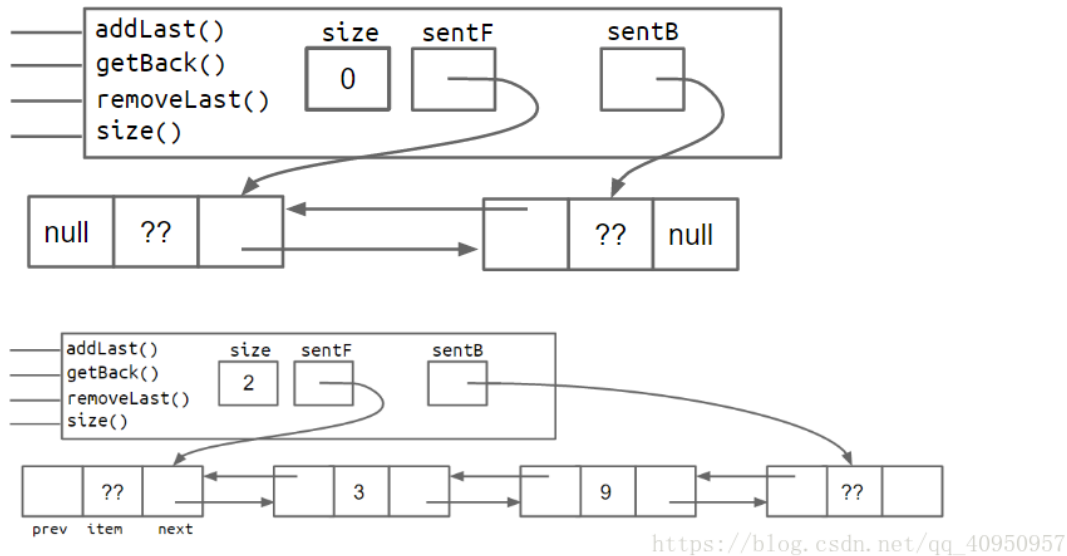
之所以称之为DLList，其实是Doubly linked list，也就是 **双向链表**，空链表和非空链表的内部结构如下所示：



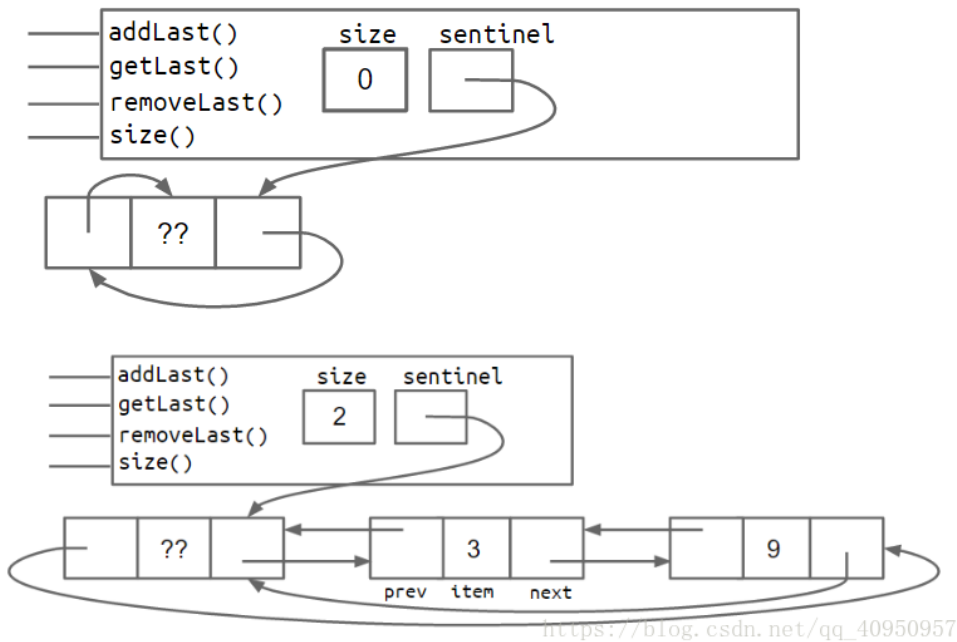
目前，我们设计的DLList已经可以在头部和尾部进行快速的添加、获取和删除操作，唯一不足的地方就是 `last` 指针有时指向带有数据的节点，有时这种特殊的空结点，在以后的实现中可能会造成混乱。

所以，我们可以用以下两种方法来避免：

- 一种是在尾部增加一个sentinel空结点。



- 一种是采用 **circle** 的思想，让首尾指针指向同一个sentinel空结点。



这两种设计思想都很完美，在这里，cs61b让学生自己实现这种数据结构，并添加一定的方法，而且需要设计成泛型，也就是Project 1A的第一部分我采取的第二种实现版本，这里是[我的实现](#)，上一个目录有说明文件。

至此，我们已经设计并实现了一种健壮实用易扩展的数据结构，更重要的是在设计过程中发现问题、解决问题的设计思想。**底层实现过程和设计思想** cs61b最重视的部分。

北京股王8年不亏铁律“1272”曝光，震惊众人！

陕西信息科技 · 爍燄

想对作者说点什么？

CS61B+CS170 👁 698
UCB的本科课程CS61B和CS170是利用java语言对数据结构与算法进行了详细的讲解... 来自： [一只小包子的博客](#)

CS61B Homework5作业回顾（附代码） 👁 408
HW5一开始做不出来的同学可以继续先看Lecture，Encapsulated List那一节老师会回... 来自： [everest115的博客](#)

（原创笔记）加州伯克利大学CS61b数据结构（Java描述）一：对象 👁 2326
OOP(object-oriented programming): object: a repository of data; class: type of object.... 来自： [Emacsor的博客](#)

区块链开发八周学会，小白程序员都能学？
区块链DApp开发学习大纲免费领

CS61B sp2018笔记 | Introduction to Java 👁 126
Introduction to Java Essentials 1. Reading 1.1 Hello World 这门课程虽然是用Java... 来自： [隐秀_](#)

CS61B sp2018笔记 | Generics and Autoboxing 👁 79
1. Automatic Conversions 1.1 Autoboxing and Unboxing Java中的泛型用到了&lt;... 来自： [隐秀_](#)

final review for berkeley cs61b
关于伯克利公开课cs61b的期末复习文档

07-21

CS61B Homework6作业回顾（附代码）

👁 92

本次作业重点是hascode和compress function，评价一个compress function好坏的标... 来自：[everest115的博客](#)**Berkeley CS61B_Data_Structures Video Lecture**

👁 1351

recorded from: http://webcast.berkeley.edu/course_details.php?seriesid=190697838... 来自：[ramboisme的专栏](#)**股市第三次机会来了！不看就亏大了！**

唐煌投资 · 熾燄

cs61a spring 类与继承笔记

👁 36

继承和属性查找

来自：[Siucaan](#)**相关热词**[cs61b](#) [cs61b是什么](#) [cs61b课程](#) [cs61b作业](#) [cs61b字幕](#)**cs61a spring 2018 Container笔记和补充**

👁 75

1.What is sequences? A sequence is an ordered collection of values. It is a collection ...

来自：[Siucaan](#)

没有开花的树

[关注](#)

162篇文章



阿_毅

[关注](#)

241篇文章



段传涛

[关注](#)

577篇文章

cs61a 2018 spring Lab4 Lists and Data Abstraction 笔记

👁 89

list Lists are Python data structures that can store multiple values. list comprehension...

来自：[Siucaan](#)**CS61B sp2018笔记 | Efficient Programming**

👁 12

Efficient Programming "An engineer will do for a dime what any fool will do for a dolla...

来自：[隐秀_](#)**CS61A 系列课程笔记（一）**

👁 1338

嗯 今天刚看第二周的课程，大量的 reading 材料我是真的要崩溃了，全英。。。我... 来自：[柠檬黄先生的博客](#)**股市第三次机会来了！不看就亏大了！**

唐煌投资 · 熾燄

CS61B Homework3作业回顾（附代码）

👁 127

Part 1.smoosh()函数，目的是将数组中出现的所有数字产生重复的部分删除，例如11... 来自：[everest115的博客](#)**伯克利计算机低年级核心课程之CS61A-SICP**

👁 8775

<http://bbs.eol.cn/forum.php?mod=viewthread&extra=page%3D1&tid=2391800> 给想... 来自：[fengjiexyb的专栏](#)**伯克利大学数据结构课程**

👁 104

<http://datastructur.es/sp17/> CS61B-Data Structures UC berkeley 来自：[yaoxiaofeng_000...](#)**C++实现控制台版2048（内附彩蛋）**

👁 124

前言 之前做过一个JavaScript版本的2048游戏，最近在学习C++，昨天晚上突然...

来自：[隐秀_](#)**CS61B Homework9作业回顾（附代码）**

👁 57

本次作业主要学会运用Disjoin Sets，DisjointSets的class作业已经写好给出。class中... 来自：[everest115的博客](#)**别再用MongoDB了**

百度广告

CS61B Homework7作业回顾(附代码)

👁 36

发现编程真的是很强调逻辑。首先Track down the whole tree, split 3-key node into 2 ... 来自：[everest115的博客](#)**cs61a课程总结--lecture7 递归（和一种数据结构）**

👁 520

数据结构 tuple 是一种至类型

来自：[vczhfan的专栏](#)

下载

Data Structures Lecture Notes (UCB CS61b)

10-20

Data Structures Lecture Notes (UCB CS61b) Data Structures Lecture Notes (UCB CS61b)

Erlang 中lists的用法详解和例子说明，详细 全

👁 3449

-。 - 收集的。官方doc也不尽详细呢。。一，带函数Pred 1, all(Pred, List) -> boolean()... 来自：[Erlang技术交流](#)

erlang lists模块函数使用大全

👁 8038

一，带函数Pred 1, all(Pred, List) -> boolean() 如果List中的每个元素作为Pred函数的参... 来自：[持续前进](#)

北京宝妈带娃也能赚钱，半年存款惊呆众人！

桥至投资 · 熯熯

lists这个类无法使用maven打包

👁 314

org.assertj.assertj-core 来自：[oppoppoppo的...](#)

Ubuntu上更新出错的解决方法。

👁 6399

http://ubuntuforums.org/showthread.php?p=10606904http://www.ithowto.ro/2008... 来自：[forlong401的专...](#)

Lists数组转换list

👁 666

将数组转换为list的时候，尽量不要用Arrays.asList，Arrays.asList()只是把数组伪装成... 来自：[sunhuwh的专栏](#)

在Ubuntu下解决E: 无法对目录 /var/lib/apt/lists/ 加锁的问题

👁 2431

使用sudo apt-get update命令时出现如下错误： E: Could not get lock /var/lib/apt/list... 来自：[BigsillyZhao的博客](#)

Ispell 在emacs中常见问题

👁 4360

1： M-x flyspell-word 后总是提示“Error: No word lists can be found for the language ... 来自：[ryuali2010的专栏](#)



别再用MongoDB了

百度广告

update-manager 软件包的错误/var/lib/apt/lists的解决命令

👁 1656

无法初始化软件包信息 update-manager软件包错误 sudo rm /var/lib/apt/lists/* -vf s... 来自：[ACanoe的专栏](#)

redis--lists类型及操作

👁 1822

lists类型List是一个链表结构，主要功能是push、pop、获取 一个范围的所有值等等， ... 来自：[杨森源的博客](#)

下载

A Java Reference: Assorted Java Reference Material

10-08

UC Berkeley CS61B Data Structure推荐阅读 Fall 2018

下载

CS61B java intro

03-31

berkeley cs61b java introduction，对初学者很有帮助

Vue警告

👁 4417

Vue警告component lists rendered with v-for should have explicit keys问题描述： 来自：[Dear_Mr的博客](#)

靠死工资怎么买房买车，他们是这样赚钱的...

虹服 · 熯熯

erlang list的使用与优化建议

👁 5963

erlang有两种复合结构，tuple和list，两者的区别是tuple子元素的个数是固定不变的... 来自：[没有开花的树](#)

Lists的使用

👁 87

List（接口）顺序时List最重要的特性：它可保证元素按照规定的顺序排列。List为Coll... 来自：[qq_37003223的...](#)

Vuejs（一）入门

👁 5746

Vue.js（一）入门官方网址🔗第一次真正接触到 Vue.js 其实是源于一次头条中的视... 来自：[风起叶落，若是...](#)

Machine Learning第六周**笔记一**：评估学习算法和bias/variance👁 3325

Machine Learning 评估学习算法 (evaluating a learning algorithm) bias/variance来自： [Marcovaldong的...](#)

下载

【伯克利CS61B教材——Java 学习教程】 Head First Java 中文版05-08

中文版的伯克利CS61B教材——经典Java 学习教程 《Head First Java 第二版》

老中医说：饭后用一物，体重不过百！北京人必看！

尹承熙 · 熨熨

Java之**Lists.Partition**项目中的使用👁 2205

开心一笑 【媳妇儿问我：“孩子都快出生了，你名字想好了没呀？”我说：“都想好了...

来自： [阿毅](#)

python列表 (**lists**)👁 58

1、序列是python中最基本的数据结构。序列中的每个元素都对应一个下标——索引位...

来自： [发飙的海豚的课...](#)

E: 无法获得锁 /var/lib/apt/**lists**/lock - open (11: 资源暂时不可用) 如何解决👁 112

在ubuntu中使用sudo apt-get update 遇到一下问题 yanga11ang@ffff-00:~\$ sudo apt-...

来自： [yanga11ang的博...](#)

python3实现冒泡排序小记👁 142

对一个列表lists里的元素进行冒泡排序 # 实现某种比较大小的方法 def mysort(a, b): ...

来自： [f1ybee的专栏](#)

erlang 列表处理函数 (**lists**)👁 855

all(Pred, List) -> bool() List中是否所有的元素都满足Pred条件 any(Pred, List) -> bool() ...

来自： [用心一也](#)



招聘海归硕士
百度广告

ubuntu错误锦集👁 82

Err:41 http://us.archive.ubuntu.com/ubuntu xenial/main arm64 Packages 404 Not Fo...

来自： [TIANJIAZHAO...](#)

Erlang list的++操作和append函数的底层实现👁 1032

当提到Erlang中list的++操作符时，我们常会想到它的性能问题。有些人知道++操作...

来自： [铿锵玫瑰的专栏](#)

下载

【Java 学习教程】 Head First Java 第二版 英文版05-08

经典Java 学习教程——Head First Java 第二版 英文版，伯克利CS61b教材

下载

CS61B 教材1 Head First Java 2nd Edition (2005)04-10

CS61B 教材1 Head First Java 2nd Edition (2005) 高清英文版

CCNA中文**笔记** 第9章Access **Lists**👁 223

转自： http://nc.zjtcn.net/showart.asp?art_id=91&cat_id=1Chapter9 Managing Traffi...

来自： [Robert's Log](#)



招聘海归硕士
百度广告

CCNA中文**笔记**第9章:Access **Lists**👁 486

作者：红头发 Chapter9 Managing Traffic with Access Lists Introduction to A...

来自： [lanndmentt的专栏](#)

竞品分析 | **Lists**👁 55

直播类：斗鱼VS虎牙

来自： [女王的专属领地](#)