队列

规范 常见问题 项目 提交

为 deque 和随机队列编写泛型数据类型。此分配的目标是使用数组和链接列表实现基本数据结构，并向您介绍泛型和迭代器。

取消排队。 *双端队列*或*deque*（发音为"deck"）是堆栈的概括，是支持从数据结构的前面或后面添加和删除项的队列。创建实现以下 API 的通用数据类型：`Deque`

```
public class Deque<Item> implements Iterable<Item> {

    // construct an empty deque
    public Deque()

    // is the deque empty?
    public boolean isEmpty()

    // return the number of items on the deque
    public int size()

    // add the item to the front
    public void addFirst(Item item)

    // add the item to the back
    public void addLast(Item item)

    // remove and return the item from the front
    public Item removeFirst()

    // remove and return the item from the back
    public Item removeLast()

    // return an iterator over items in order from front to back
    public Iterator<Item> iterator()

    // unit testing (required)
    public static void main(String[] args)

}
```

*角箱。*引发以下角情况的指定异常：

- 如果客户端调用 或 使用 参数调用， 引发 。`IllegalArgumentExceptionaddFirst()addLast()null`
- 如果客户端调用任一或当 deque 为空时，将引发 。`java.util.NoSuchElementExceptionremoveFirst()removeLast`
- 当没有更多的项要返回时，如果客户端在迭代器中调用该方法，则引发 a。`java.util.NoSuchElementExceptionnext()`
- 如果客户端在迭代器中调用 方法，则引发 。`UnsupportedOperationExceptionremove()`

*单元测试。*方法必须直接调用每个公共构造函数和方法，以帮助验证它们是否按规定工作（例如，通过将结果打印到标准输出）。`main()`

*性能要求。*您的 deque 实现必须在*持续最坏情况下*支持每个 deque 操作（包括构造）。包含n个项的 deque 最多只能使用 $48n + 192$ 字节的内存。此外，迭代器实现必须在*持续最坏情况下*支持每个操作（包括构造）。

随机队列。 *随机队列*类似于堆栈或队列，只不过删除的项在数据结构中的项之间随机选择。创建实现以下 API 的通用数据类型：`RandomizedQueue`

```
public class RandomizedQueue<Item> implements Iterable<Item> {

    // construct an empty randomized queue
    public RandomizedQueue()

    // is the randomized queue empty?
    public boolean isEmpty()

    // return the number of items on the randomized queue
    public int size()

    // add the item
    public void enqueue(Item item)

    // remove and return a random item
    public Item dequeue()

    // return a random item (but do not remove it)
    public Item sample()

    // return an independent iterator over items in random order
    public Iterator<Item> iterator()

    // unit testing (required)
    public static void main(String[] args)

}
```

*迭代器.*每个迭代器必须按统一随机顺序返回项目。同一随机队列的两个或多个迭代器的顺序必须是*相互独立的*;每个迭代器必须维护自己的随机顺序。

*角箱。*引发以下角情况的指定异常：

- 如果客户端调用参数，则引发 。`IllegalArgumentException`enqueue()`null`
- 如果客户端调用任一或随机队列为空时，将引发 。`java.util.NoSuchElementException`sample()dequeue()
- 当没有更多的项要返回时，如果客户端在迭代器中调用该方法，则引发 a。`java.util.NoSuchElementException`next()
- 如果客户端在迭代器中调用 方法，则引发 。`UnsupportedOperationException`remove()

*单元测试。*方法必须直接调用每个公共构造函数和方法，以验证它们是否按规定工作（例如，通过将结果打印到标准输出）。 `main()`

*性能要求。*您的随机队列实现必须支持每个随机队列操作（除了创建迭代器）在*恒定的摊销时间*。也就是说，在最坏的情况下，对于某些常量c，m*随机队列*操作的任何混合序列（从空队列开始）最多必须采用cm步长。包含n个项的随机队列最多只能使用 $48n + 192$ 字节的内存。此外，迭代器实现必须支持操作，并在*持续最坏情况下*;线性*时间的*构造;您可以（并且需要）每个迭代器使用线性的额外内存量。 `next()`hasNext()

客户端。编写一个客户端程序，该程序将整数k作为命令行参数;使用 从 标准输入读取字符串序列 ， 并随机统一打印它们的确切k。最多从序列中打印一次每个项目。 `Permutation.java`StdIn.readString()

```
~/Desktop/queues> cat distinct.txt
A B C D E F G H I

~/Desktop/queues> java Permutation 3 < distinct.txt
C
G
A

~/Desktop/queues> java Permutation 3 < distinct.txt
E
F
G
```

```
~/Desktop/queues> cat duplicates.txt
AA BB BB BB BB BB CC CC

~/Desktop/queues> java Permutation 8 < duplicates.txt
BB
AA
BB
CC
BB
BB
CC
BB
```

您的程序必须实现以下 API：

```
public class Permutation {
   public static void main(String[] args)
}
```

*命令行参数。*可以假定 $0 = k = n$，其中n是标准输入上的字符串数。请注意，您没有获得n。

*性能要求。*的运行时间必须是输入大小的线性。 您只能使用恒定的内存量*加上最多*n 个最大大小的一个或最大大小的对象。 （对于额外的挑战和少量的额外积分，最多使用一个或最大大小的*对象。*
PermutationDequeRandomizedQueueDequeRandomizedQueue

Web 提交。提交仅包含 、和 的 .zip 文件。您的提交不能调用库函数，除了那些在StdIn、 StdOut、 StdRandom、 java.lang、 java.util.iteror、 和java.util.NosuchElement异常 .特别是，不要使用java.util.LinkList 或java.util.Array。 RandomizedQueue.javaDeque.javaPermutation.java

This assignment was developed by Bob Sedgewick and Kevin Wayne.