Question 1

(a)

(i)

随问题规模n的增大，算法执行时间的增长率和f(n)的增长率相同，

称作算法的渐进时间复杂度，简称时间复杂度。其中f(n)是问题规模n的某个函数。

时间复杂度不包括这个函数的低阶项和首项系数

(ii)

5n^2+11n+9：

项的阶数有：5n^2>11n>9

去除低阶项的5n^2

去除首项系数为：n^2

即结果为：O(n^2)

(b)

选择排序：每次从待排序数组里选择一个最小(或最大)的数字，放到数组最前面

时间复杂度为n^2：每次从数组中遍历，找到一个最小的数字，为n；需要n次这样的遍历；即n\*n=n^2

Question 2

public class StringQueueImpl implements StringQueue {  
 private QueueCell head; //存储数据的节点，内部类  
 private int size = 0; // 队列大小  
  
 @Override  
 public boolean isEmpty() {  
 return size == 0;  
 }  
  
 @Override  
 public void add(String c) {  
 if (head == null) {  
 // 如果队列为空，创建一个新节点  
 head = new QueueCell(c);  
 } else {  
 // 不为空，则将新的值加入队尾  
 QueueCell current = head;  
 // 遍历，将current指向队尾  
 while (current != null) {  
 if (current.next == null) {  
 current.next = new QueueCell(c);  
 break;  
 } else {  
 current = head.next;  
 }  
 }  
 }  
 size++;  
 }  
  
 @Override  
 public String front() {  
 // 如果不为空，则直接返回头结点的值  
 if (size == 0 || head == null) {  
 throw new QueueException();  
 }  
 return head.value;  
 }  
  
 @Override  
 public void removeFront() {  
 // 如果为空，则抛出异常  
 if (size == 0 || head == null) {  
 throw new QueueException();  
 }  
  
 // 否则，队头指针向后移动一位；即移除对头元素  
 if (head.next != null) {  
 head = head.next;  
 } else {  
 head = null;  
 }  
 size--;  
 }  
  
 // 节点类  
 class QueueCell {  
 private QueueCell next;  
 private String value;  
  
 QueueCell(String str) {  
 value = str;  
 }  
 }  
  
 // 自定义异常类  
 class QueueException extends RuntimeException {  
 public QueueException() {  
 super("Queue is null");  
 }  
 }  
}

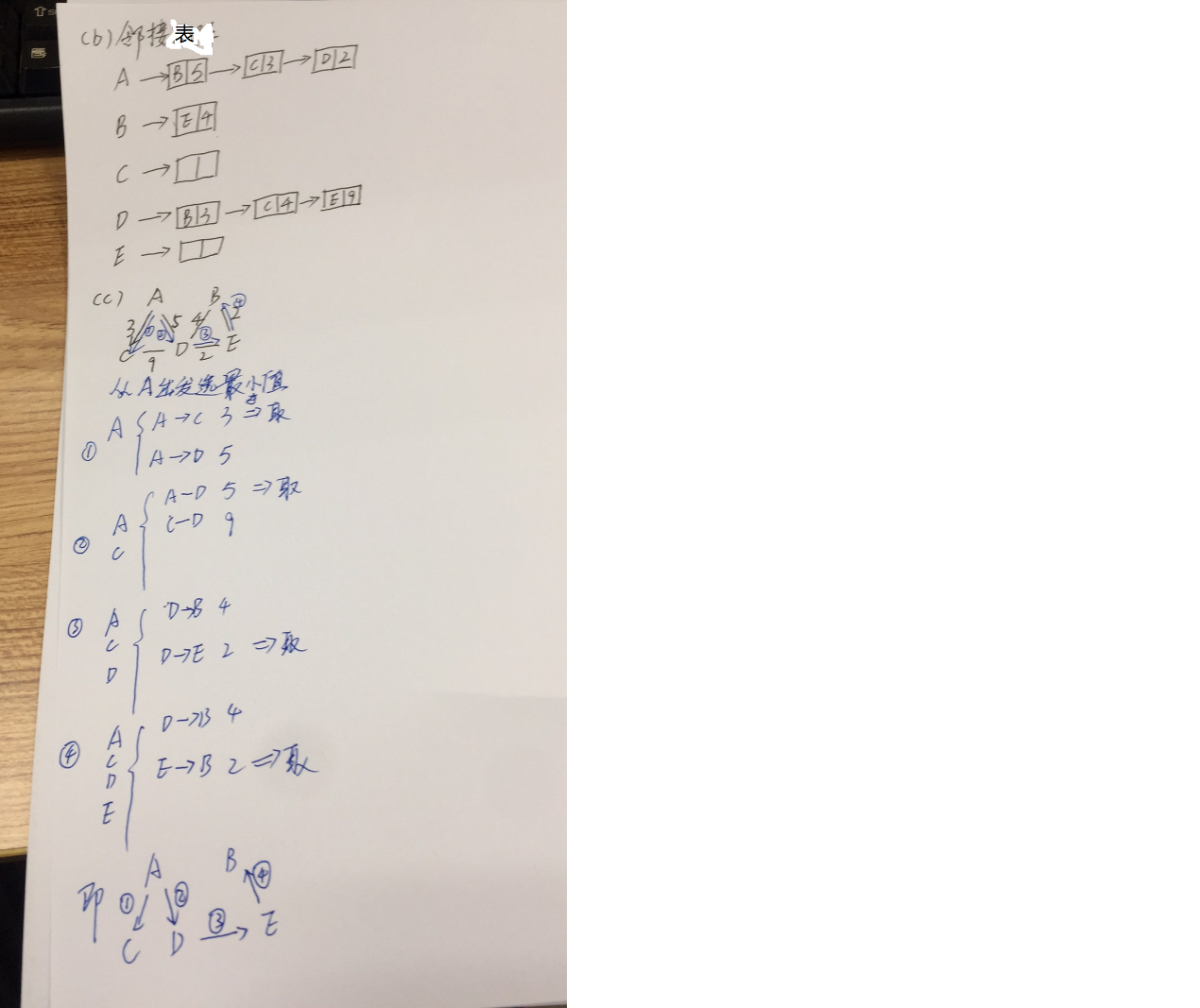
**Question 3**

(a)

描述：判断一个程序能否在有限的时间内结束或者死循环

证明：如果存在一个判断停机问题的程序H（H需要的输入是一个程序），我们再构造一个新的程序K，这个程序调用H但是与H的输出正好相反：如果K的输入经H判断为停机，则K不停机；如果K的输入经H判断为不停机，则K停机。现在矛盾出现了：如果我们把K输入K（即用H判断对于程序K，给出输入为K），那么K停机么？如果按逻辑推演，答案应该是：如果K不停机则K停机；如果K停机则K不停机。矛盾出现了。唯一解决矛盾的解释是：不存在这样万能的H。

(b)(c)：



**Question 4**

(a)

public int getNumOfNonLetters(BinaryTree<Character> bt) {  
 int num = 0; // 总字符数  
 int i = 0; // 是否为字母字符：是+1，否+0  
 if (bt.isEmpty()) {  
 // 如果为空，则字符数为0，返回  
 return num;  
 } else {  
 if (Character.isLetter(bt.rootValue())) {  
 i = 1;  
 }  
 // 递归调用，加上左右节点总数  
 num = getNumOfNonLetters(bt.leftChild())+1;  
 num = num+getNumOfNonLetters(bt.rightChild());  
 }  
 return num;  
}

(b)

参考博客：<https://blog.csdn.net/wxbmelisky/article/details/47787963>

<https://blog.csdn.net/wxbmelisky/article/details/47755753>

