插入排序的正确性证明

张天昀

南京大学计算机科学与技术系
171860508@smail.nju.edu.cn

2018 年 3 月 12 日

Insertion Sort Algorithm

Before

已排序的部分		未排序的部分	
$< x$	$> x$	x	.....

Insertion Sort Algorithm

Before

已排序的部分		未排序的部分	
$< X$	$> X$	X	.....

将未排序的第一个元素与前面已经排序的元素进行对比，找到合适的位置，插入已经排序的元素中。

Insertion Sort Algorithm

Before

已排序的部分		未排序的部分	
$< X$	$> X$	X	.....

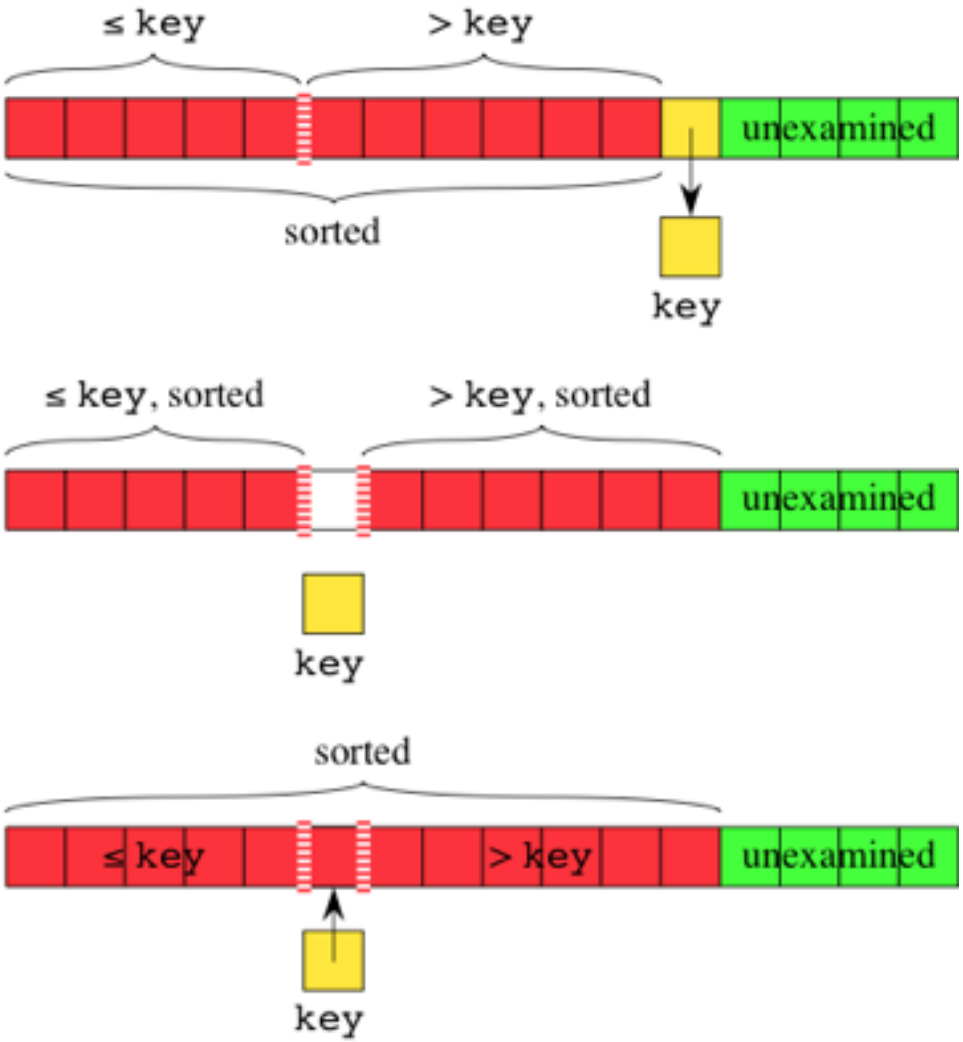
将未排序的第一个元素与前面已经排序的元素进行对比，找到合适的位置，插入已经排序的元素中。

After

已排序的部分			未排序的部分
$< X$	X	$> X$	.....

重复以上过程，直到整个数组有序。

Insertion Sort Algorithm

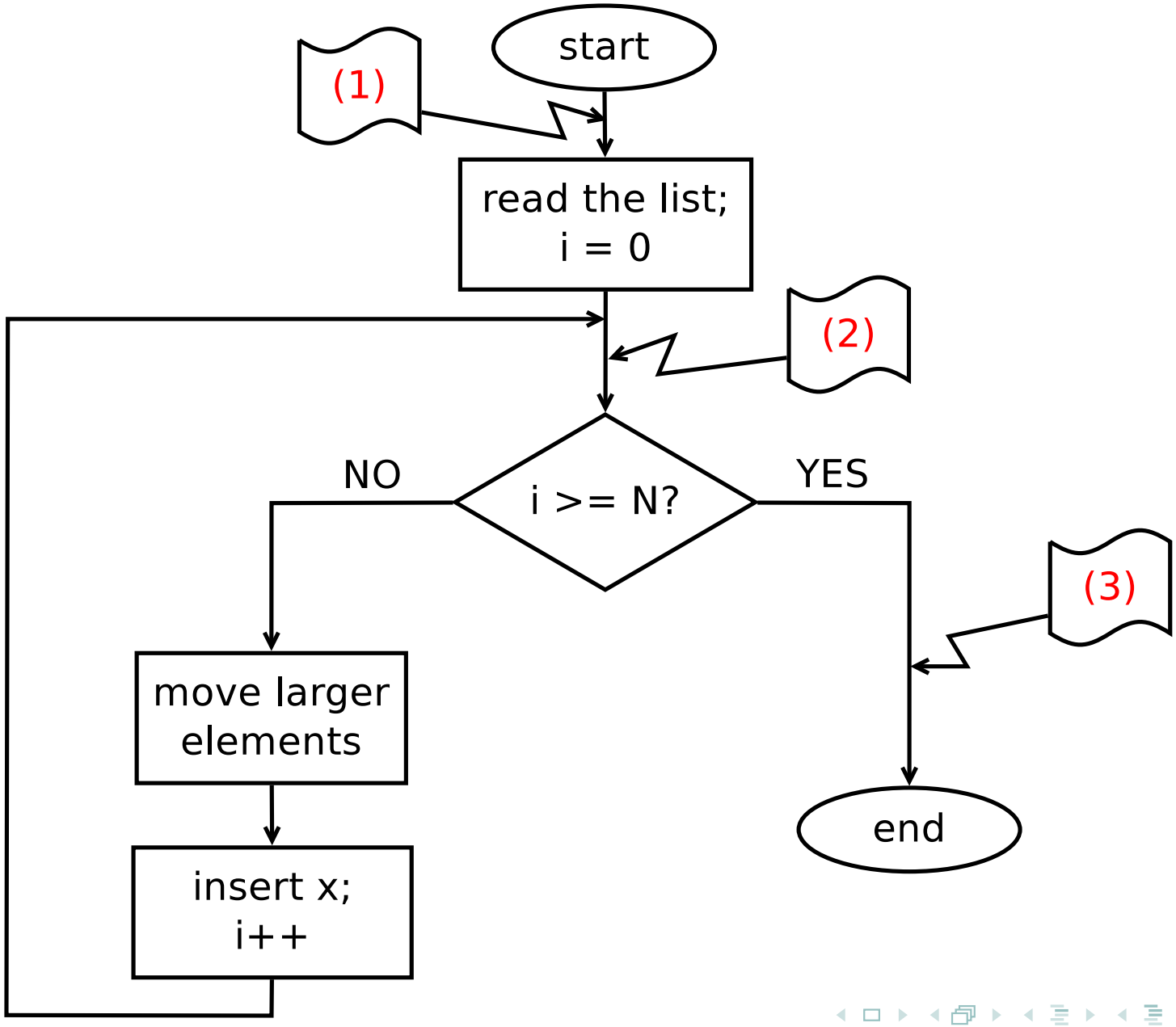


Insertion Sort Algorithm

C++

```
void sort(int* L, int length) {
    int tmp = 0, pos = 0;
    for (int i = 1; i <= length; i++) {
        tmp = L(i);
        pos = i;
        while (pos >= 0 && tmp < L(pos)) {
            L(pos) = L(pos-1);
            pos--;
        }
        L(pos) = tmp;
    }
}
```

Flowchart


start
(1)
read the list;
i = 0
(2)
i >= N?
NO
YES
move larger
elements
insert x;
i++
(3)
end

Loop invarient

totalwork = workdone + worktodo

Loop invarient

totalwork = workdone + worktodo

外循环

- totalwork = 排序整个列表
- workdone = 前 k 个已经排序
- worktodo = 把后面的 $n - k$ 个元素插入已经排序的序列中

Loop invarient

totalwork = workdone + worktodo

外循环

- totalwork = 排序整个列表
- workdone = 前 k 个已经排序
- worktodo = 把后面的 $n - k$ 个元素插入已经排序的序列中

内循环

- totalwork = 找到合适的插入位置
- workdone = 移动了 k 次，后面的元素都比要插入的元素大
- worktodo = 继续向前寻找

Assertions

4 assertions:

(1) pre-condition

检查输入：合法的输入是一个可以排序（定义了比较运算符）的数组。

(2) loop invarients

外循环：经过 k 次循环后，$L(0) \ldots L(k)$ 已经排序。
内循环：移动 k 次后，当前位置后的已排序元素都比 $L(k+1)$ 大。

(3) post-condition

算法结束：整个数组 L 已经排序。

Proof of partial correctness

(1) $\rightarrow$ (2)

循环开始前，$k = 0$，此时 $L(0) \ldots L(k)$ 只有一个元素，已经排序。

Proof of partial correctness

(1) $\rightarrow$ (2)

循环开始前，$k = 0$，此时 $L(0) \ldots L(k)$ 只有一个元素，已经排序。

(2) $\rightarrow$ (3)

循环结束时，一共进行了 $N - 1$ 次循环，$k = N - 1$，此时 $L(0) \ldots L(N - 1)$ 都已经排序，即整个数组已经排序。

Proof of partial correctness

(2) $\rightarrow$ (2)

Proof of partial correctness

(2) $\rightarrow$ (2)

假设已经进行了 k 次循环，此时 $L(0) \ldots L(k)$ 已经有序。

<table>
<tr><td colspan="5">已排序的部分</td><td colspan="2">未排序的部分</td></tr>
<tr><td>$L(0)$</td><td>$L(1)$</td><td>$\ldots$</td><td>$\ldots$</td><td>$L(k)$</td><td>$L(k+1)$</td><td>$\ldots\ldots$</td></tr>
<tr><td colspan="3">$< L(k+1)$</td><td colspan="2">$> L(k+1)$</td><td colspan="2"></td></tr>
</table>

Proof of partial correctness

(2) $\rightarrow$ (2)

假设已经进行了 k 次循环，此时 $L(0) \ldots L(k)$ 已经有序。

已排序的部分					未排序的部分	
$L(0)$	$L(1)$	$\ldots$	$\ldots$	$L(k)$	$L(k+1)$	$\ldots\ldots$
$<$tmp				$>$tmp	tmp $= L(k+1)$	

第 $k+1$ 次循环中，先将 $L(k+1)$ 取出，

Proof of partial correctness

(2) $\rightarrow$ (2)

假设已经进行了 k 次循环，此时 $L(0) \ldots L(k)$ 已经有序。

<table>
<tr><td colspan="5">已排序的部分</td><td colspan="2">未排序的部分</td></tr>
<tr><td>$L(0)$</td><td>$L(1)$</td><td>$\ldots$</td><td>$\ldots\rightarrow$</td><td>$L(k)\rightarrow$</td><td></td><td>$\ldots\ldots$</td></tr>
<tr><td colspan="3">$<$tmp</td><td colspan="2">$>$tmp</td><td colspan="2">tmp = $L(k+1)$</td></tr>
</table>

第 $k+1$ 次循环中，先将 $L(k+1)$ 取出，将比它大的所有数都向后移 1 位。

Proof of partial correctness

(2) → (2)

假设已经进行了 k 次循环，此时 $L(0) \ldots L(k)$ 已经有序。

已排序的部分					未排序的部分	
$L(0)$	$L(1)$	$\ldots$	$\ldots\rightarrow$	$L(k)\rightarrow$		$\ldots\ldots$
$<$tmp				$>$tmp	tmp $= L(k+1)$	

第 $k+1$ 次循环中，先将 $L(k+1)$ 取出，将比它大的所有数都向后移 1 位。 此时前面的数都比它小，后面的数都比它大，在这个位置插入元素，得到的新的 $L(0) \ldots L(k+1)$ 有序，循环不变式成立。

已排序的部分						未排序的部分
$L'(0)$	$L'(1)$	$\ldots$	$L'(k)$	$\ldots$	$L'(k+1)$	$\ldots\ldots$
$< L'(k)$				$> L'(k)$		

Proof of termination

Code

```
for (int i = 1; i <= length; i++) {
    tmp = L(i);
    pos = i;
    while (pos >= 0 && tmp < L(pos)) {
        L(pos) = L(pos-1);
        pos--;
    }
    L(pos) = tmp;
}
```

因为每进行一次插入，计数器 i 都会自增，
所以从 1 开始，i 一定会到达数组的长度 n 并结束循环。
所以输入合法时整个程序会结束。