void sort(int[ ] A, int lower, int upper)

//@requires 0 <= lower && lower <= upper && upper <= \length(A);

//@ensures is\_sorted(A, lower, upper);

{

for (int i = lower; i < upper; i++)

//@loop\_invariant lower <= i && i <= upper;

//@loop\_invariant is\_sorted(A, lower, i);

//@loop\_invariant le\_segs(A, lower, i, i, upper);

{

int m = min\_index(A, i, upper);

//@assert le\_seg(A[m], A, i, upper);

swap(A, i, m);

}

return;

}

证明循环不变量：

lower <= i && i <= upper

is\_sorted(A, lower, i)

le\_segs(A, lower, i, i, upper)

首先，我们验证循环不变量在初始情况下是满足的。

* 和因为和 (通过前提条件(@requires))。
* 有序，因为对，数组段 是空的（没有元素），因为右边边界已经排除在外了。
* 是真的，因为对，数组段 没有元素，其他的段是待排序数组段的整个部分。

接下来，我们假定在执行循环前，循环不变量是成立的，然后证明通过一次循环处理后，循环不变量仍然成立，那么就证明了循环不变量总是成立的。这一步证明过程留给大家做。

我们也应该验证加入循环体中的断言。

//@assert le\_seg(A[m], A, i, upper);

它表示小于或等于数组段中的所有元素，数学上简写为。这应该是min\_index函数的后置条件。

我们怎么验证排序函数的后置条件(@ensures)呢？通过循环不变量和循环条件i < upper的否定，即：，我们可以知道。而第二个循环不变量表明是有序的，这就验证了函数后置条件。