**int binsearch(int x, int[ ] A, int n)**

**//@requires 0 <= n && n <= \length(A);**

**//@requires is\_sorted(A, 0, n);**

**/\*@ensures (-1 == \result && !is\_in(x, A, 0, n))**

**|| ((0 <= \result && \result < n) && A[\result] == x);**

**@\*/**

**{**

**int lower = 0;**

**int upper = n;**

**while (lower < upper)**

**//@loop\_invariant 0 <= lower && lower <= upper && upper <= n;**

**//@loop\_invariant (lower == 0 || A[lower-1] < x);**

**//@loop\_invariant (upper == n || A[upper] > x);**

**{**

**int mid = lower + (upper-lower)/2;**

**//@assert lower <= mid && mid < upper;**

**if (A[mid] == x) return mid;**

**else if (A[mid] < x) lower = mid+1;**

**else /\*@assert(A[mid] > x);@\*/**

**upper = mid;**

**}**

**return -1;**

**}**

证明上面代码中的三个循环不变量：

//@loop\_invariant 0 <= lower && lower <= upper && upper <= n; (1)

//@loop\_invariant (lower == 0 || A[lower-1] < x); (2)

//@loop\_invariant (upper == n || A[upper] > x); (3)

证明：

1. 在初始情况下，有lower = 0和upper = n，而且函数前置条件：

0 <= n && n <= \length(A);

所以，循环不变量(1)、(2)和(3)都成立。

2. 假设在某次进入循环前，3个循环不变量都成立，即有：

0≤lower ≤upper ≤n （循环不变量1）

lower = 0 或 A[lower-1] < x （循环不变量2）

upper = n 或 A[upper] > x （循环不变量3）

另外：lower < upper （循环条件）

在此条件下，我们需证明经过一次循环处理后，三个循环不变量是否仍然成立。

计算mid = lower + └ (upper-lower)/2 ┘。

现在分三种情况进行讨论：

(1) A[mid] = x：

这种情况是找到了x，函数将返回到被调用的地方，不需要判断循环不变量是否成立；尽管如此，由于在这种情况下lower和upper都没有改变，所以循环不变量(1)、(2)和(3)仍然成立。

(2) A[mid] < x：

那么，，

由代码中计算公式、我们的假设和初级算术规则，得到第一个循环不变量成立。

对于第二个循环不变量，我们计算：

 

 

 

因此，第二个循环不变量也成立。

对于第三个循环不变量，因为，所以保持成立。

(3) A[mid] > x：

那么，，

再一次，通过初级算术运算，可推导出，即第一个循环不变量成立。

对于第二个循环不变量，因为，所以第二个不变量保持成立。

对第三个循环不变量，我们计算：

 

 

所以第三个循环不变量也成立。