第六讲

1.

```
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-1;
  while (lower < upper)
    //@loop_invariant 0 <= lower && lower <= upper+1 && upper < n;
    //@loop_invariant (lower == 0 || A[lower-1] < x);
    //@loop_invariant (upper == n-1 || A[upper] >= x);
    {
      if(A[upper]==x) return upper;
      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;
}
```

循环不变量为：

（1）//@loop_invariant 0 <= lower && lower <= upper+1 && upper < n;

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

（3）//@loop_invariant (upper == n-1 || A[upper] >= x);

证明：

初始条件下，lower=0，upper=n-1，此时循环不变量（1）（2）（3）均满足。

假设在某次循环过程中，三个循环不变量都成立，则：

$0 \leqslant lower \leqslant upper < n$

lower=0 或 A[lower-1]$<$x

upper=n-1 或 A[upper]$\geqslant$x

mid=lower+(upper-lower)/2

若 A[mid]=x 或 A[upper]=x，则函数返回，lower 与 upper 不变，（1）（2）（3）均满足；

若 A[upper]$\neq$x 且 A[mid]$<$x，则 lower'=mid+1，upper 不变，（3）满足，此时 0$\leqslant$lower'$\leqslant$upper'<n 仍然成立，A[lower'-1]=A[mid]$<$x，（1）（2）（3）均满足；

若 A[upper]$\neq$x 且 A[mid]$>$x，则 upper'=mid，lower 不变，（2）满足，此

时 0≤lower'≤upper'<n 仍然成立，A[upper']=A[mid]>x，（1）（2）（3）均满足

综上所述，循环不变量（1）（2）（3）成立，由此证明了循环不变量的正确性。

2.

循环不变量改为

//@loop_invariant 0<=lower && lower<=upper && upper<=n;

//@loop_invariant is_in(x,A,lower,mid) || is_in(x,A,mid,upper) || !is_in(x,A,lower,upper)

证明：

若 x 在数组 A 中，则有 is_in(x,A,lower,mid) || is_in(x,A,mid,upper) = True，满足循环不变量。此时后置条件中((0<=\result && \result<n) && A[\result]==x) = True，满足后置条件。

若 x 不在数组 A 中，则有 !is_in(x,A,lower,upper) = True，满足循环不变量。此时后置条件中(-1 == \result && !is_in(x,A,0,n)) = True，满足后置条件。

3.

```
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 && A[mid-1] < x} return mid;
      else if (A[mid] < x) lower = mid+1;
      else /*@assert(A[mid] > x);@*/
        upper = mid;
    }
   return -1;
}
```

循环不变量为：

（1）//@loop_invariant 0 <= lower && lower <= upper && upper <= n;

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

（3）//@loop_invariant (upper == n || A[upper] >= x);
证明：

初始条件下，lower=0，upper=n，循环不变量（1）（2）（3）均满足。

假设在某次循环过程中，三个循环不变量都成立，则：

mid=lower+(upper-lower)/2

若 A[mid]==x && A[mid-1]＜x，函数返回，lower 与 upper 均不变，（1）（2）（3）均满足。

若 A[mid]＜x，则 lower’=mid+1，upper 不变，（3）满足，此时 0≤lower’≤upper’＜n 成立，A[lower’-1]=A[mid]＜x，（2）满足。

若 A[mid]＞x || (A[mid]==x && A[mid-1]==x)，则 lower 不变，upper’=mid，此时 0≤lower’≤upper’＜n 成立，A[upper’]=A[mid]≥x，（3）满足。

综上所述，循环不变量（1）（2）（3）成立，由此证明了循环不变量的正确性。

4.

当 lower 与 upper 的值非常大时，lower+upper 会出现溢出的情况，导致 mid 的值为负数，造成数组越界，造成这种情况下约定失败。

5.

这样做存在的问题是，将循环体改变后循环无法终止，形成死循环，因此这个循环体不能正确地实现二分查找。

Recitation 7

Checkpoint 0：

	$O(\log 2n)$					
$O(4)$	$O(\log(\log(n)))$	$O(4n+3)$		$O(n^2+200000n+3)$		
$O(1)$	$O(\log(n))$	$O(n)$	$O(n\log(n))$	$O(n^2)$	$O(2^n)$	$O(n!)$

Checkpoint 1：

证明：

$n^3+300n^2-cn^3\leqslant 0$

$(1-c)n^3+300n^2\leqslant 0$

当 $c=2$ 时，$300n^2-n^3\leqslant 0$，$n^2(300-n)\leqslant 0$

$\therefore \exists c=2$，$\forall x>300$，$n^3+300n^2\leqslant 2n^3$

$\therefore n^3+300n^2\in O(n^3)$

Checkpoint 2：

证明：

$\because f(n)\in O(g(n))$

$\therefore \exists c$，$\forall n>n_0$，$f(n)\leqslant g(n)$

不等式两边同时乘上 k，得

$k\cdot f(n)\leqslant c\cdot k\cdot g(n)$

$\therefore \exists c'=c\cdot k$，$\forall n>n_0$，$k\cdot f(n)\leqslant c'\cdot g(n)$

$\therefore k\cdot f(n)\in O(g(n))$