Homework 4

ACM1501　　U201514716　　罗海旻

Exercise 1. Rewrite the binary search function so that both lower and upper bounds of the interval are inclusive. Make sure to rewrite the loop invariants and the loop body appropriately, and prove the correctness of the new loop invariants. Also explicitly prove termination by giving a measure that strictly decreases each time around the loop and is bounded from below.

解：修改后代码如下

```
int search(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 && upper < n;
    //@loop_invariant lower == 0     || A[lower-1] < x;
    //@loop_invariant upper == n-1 || A[upper+1] > 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-1;
      }
    }
  //@assert lower == upper;
  return -1;
}
```

证明如下：

根据循环不变量以及循环条件有如下不等式：

0<=lo<=hi<n

lo=0 or A[lo-1]<x

hi=n-1 or A[hi+1]>x

lo<hi

如下情况中：

A[mid]=x:成立

A[mid]<x: lo’ = mid+1;
A[lo’- 1] = A[mid+1 - 1]=A[mid] < x;
lo’ = mid+1 = lo + (hi-lo)/2 + 1 = (hi + lo)/2 + 1 < hi + 1
即 lo’<=hi

即循环不变量不发生改变。

A[mid]>x:hi’ = mid -1;
A[hi’+1]=A[mid-1 + 1] = A[mid] > x;
Hi’ = mid - 1 = (hi+lo)/2 - 1> lo - 1
即 hi’ >=lo

即循环不变量不发生改变。

综上所述，循环不变量保持不变。

下面证明循环可以终止：

由于循环条件为 lo<hi，且随着循环的推进，lo 严格增大而 hi 严格减小，这样的话必然导致 lo==hi 的情况存在，那么退出循环。从而即使未找到匹配的下标，循环也会终止。

Exercise 2. Rewrite the invariants of the binary search function to use is_in(x, A, l, u) which returns true if and only if there is an i such that x = A[i] for l ≤ i < u. is_in assumes that 0 ≤ l ≤ u ≤ n where n is the length of the array.

Then prove the new loop invariants, and verify that they are strong enough to imply the function’s post-condition.

解：

```
int search(int x, int[] A, int n)
//@requires n == \length(A);
//@requires is_sorted(A, 0, n);
/*@ensures (\result == -1 && !is_in(x, A, 0, n))
        || (0 <= \result && \result < n && A[\result] == x); @*/
{
  int lo = 0;
  int hi = n;

  while (lo < hi)
  //@loop_invariant 0 <= lo && lo <= hi && hi <= n;
  //@loop_invariant !is_in(x, A,0,lo);
  //@loop_invariant !is_in(x, A,hi,n);
  {
    int mid = lo + (hi - lo)/2;
    //@assert lo <= mid && mid < hi;
    if (A[mid] == x) return mid;
    if (A[mid] < x) {
      //@assert mid + 1 <= hi;
      //@assert !is_in(x, A,0,mid+1);
      lo = mid+1;
    } else { //@assert A[mid] > x;
      //@assert !is_in(x, A,mid,n);
      hi = mid;
    }
  }
  //@assert lo == hi;
  //@assert !is_in(x,A,0,n);
  return -1;
}
```

证明如下：

已证明对于该循环，下列不等式成立:

0<=lo<=hi<n
lo=0 or A[lo-1]<x
hi=n or A[hi]>x

那么由于 A 已经排序，那么 A[lo-1]以及之前的元素均小于 x，x 不可能为其中任何一个。即 !is_in(x, A,0,lo);成立。

同理，A[hi]以及之后的元素均大于 x，则 x 不可能为其中任何一个元素，即 !is_in(x, A,hi,n);成立。

综上所述，得证。

Exercise 3. Binary search as presented here may not find the leftmost occurrence of x in the array in case the occurrences are not unique. Given an example demonstrating this.

Now change the binary search function and its loop invariants so that it will always find the leftmost occurrence of x in the given array (if it is actually in the array, −1 as before if it is not).

Prove the loop invariants and the post-conditions for this new version, and verify termination.

解：

```
int search(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) {
          if (mid == 0 || A[mid - 1] < x)
              return mid;
          else
              hi = mid;
          }
        } else if (A[mid] < x) {
          lower = mid+1;
        } else {
          //@assert A[mid] > x;
          upper = mid;
        }
      }
    //@assert lower == upper;
    return -1;
  }
```

证明如下:

已有:

0<=lo<=hi<n

lo=0 or A[lo-1]<x

hi=n or A[hi]>=x

lo<hi

如下情况中:

A[mid]=x 且(mid=0 or A[mid-1] < x):成立

A[mid]=x 且(mid !=0 or A[mid-1] = x):

hi’ = mid

A[hi’] = A[mid] = x >= x;

即循环不变量不改变

A[mid]<x: lo’ = mid+1;

A[lo’- 1] = A[mid+1 - 1]=A[mid] < x;

lo’ = mid+1 = lo + (hi-lo)/2 + 1 = (hi + lo)/2 + 1 < hi + 1

即 lo’<=hi

即循环不变量不发生改变。

A[mid]>x:hi’ = mid ;

A[hi’] = A[mid] > x;
Hi’ = mid = (hi+lo)/2 > lo
即 hi’ >= lo

即循环不变量不发生改变。

综上所述，循环不变量保持不变。

下面证明循环可以终止：

由于循环条件为 lo<hi，且随着循环的推进，lo 严格增大而 hi 严格减小，这样的话必然导致 lo==hi 的情况存在，那么退出循环。从而即使未找到匹配的下标，循环也会终止。

Exercise 4. If you were to replace the midpoint computation by
int mid = (lo + hi)/2;then which part of the contract will alert you to a flaw in your thinking? Why?Give an example showing how the contracts can fail in that case.

解：

lo+hi 可能引发溢出。

例如 0x1+0x7fffffff = 0x80000000 发生溢出。

Exercise 5. In lecture, we used design-by-invariant to construct the loop body implementation from the loop invariant that we have identified before. We could also have maintained the loop invariant by replacing the whole loop body just with

```
// .... loop_invariant elided ....
{
lo = lo;
hi = hi;
}
```

Prove the loop invariants for this loop body. What is wrong with this choice? Which part of our proofs fail, thereby indicating why this loop body would not implement binary search correctly?

解:

在此循环中，lo 和 hi 的值始终保持不变，即本身即为循环不变量。

这个循环永远不会终止，不会得到需要的结果。