

CSE 202: Loop Invariants

Binary Search specification

Write a binary search algorithm in pseudo-code. Provide formal proof of its correctness. Provide formal proof of its running time.

Given:

function BinarySearch($A[1..n]$, key)

Pre-condition:

- A is a sorted array indexed from 1 to n .
- key is an element to be searched for

Post-condition:

- If key is an element of the array, then returns i , where $A[i] = key$
- If key is not an element of the array, returns "no".

BINARYSEARCH($A[1..n]$, key)

```
1   $i \leftarrow 1$ 
2   $j \leftarrow n$ 
3  loop
4      precondition: if  $key$  is in original list,  $key$  is in  $A[i..j]$ 
5      exit when  $j \leq i$ 
6       $mid = \lceil \frac{i+j}{2} \rceil$ 
7      if  $key < A[mid]$ 
8          then
9               $j \leftarrow mid - 1$ 
10         else
11              $i = mid$ 
12 end loop
13 if  $i = j \& key = A[i]$ 
14     then
15         return  $i$ ;
16     else
17         return "no";
```

Necessary proofs

Precondition & code_{preloop} → LoopInvariant

If *key* is in $A[1..n]$, *key* is in $A[1..n]$ (by substitution of equals)

LoopInvariant & ExitCondition & code_{postloop} → postcondition

Because of the exit condition ($j \leq i$), there are two cases:

- $j < i$

By *LoopInvariant*, if *key* is in the original list *key* is in empty list but by the definition of empty list, *key* is not in it. *code_{postloop}* returns "no" in this case, satisfying *postcondition*.

- $j = i$

By *LoopInvariant*, if *key* is in original array, *key* is in $A[i..j]$. By substitution, if *key* is in original array, *key* is in $A[i..i]$. *code_{postloop}* returns *i* in this case, which satisfies the postcondition. If *key* is not in the array, then $key \neq A[i]$ and the code returns "no";

LoopInvariant' & not(ExitCondition) & code_{loop} → LoopInvariant''

LoopInvariant': if *key* is in original list, *key* is in $A[i..j]$.

not(ExitCondition): $not(j \leq i) \rightarrow i < j$

Since $i < j$, $\lceil \frac{i+j}{2} \rceil \leq j$ (because $\lceil \frac{j+j}{2} \rceil = j$). Similarly, $\lceil \frac{i+j}{2} \rceil > i$

If the *key* is in $A[i..j]$, then *key* must be in either $A[i..mid-1]$, or $A[mid, j]$.

Case 1: If *key* is in original list, *key* is in $A[i..mid-1]$

In this case, $key < A[mid]$ (because *A* is sorted).

Code is of form

$i'' \leftarrow i'$

$j'' \leftarrow mid - 1$

Thus, if *key* is in original list, *key* is in $A[i'', j'']$.

Case 2: If *key* is in original list, *key* is in $A[mid..j]$

In this case, $key \geq A[mid]$ (because *A* is sorted). Code is of the form:

$i'' \leftarrow mid$

$j'' \leftarrow j'$

Thus, if *key* is in original list, *key* is in $A[i'', j'']$

Loop terminates

Every time through the loop, $j'' - i'' < j' - i'$, and i and j are integers. The loop terminates if $j = i$.