Questions from Dr. Yuepeng Wang

Jingqian Liu

2. Given a sorted array of integers A and an integer value V, write a procedure/method that performs a binary search to find an index i such that $A[i] = V$. If V does not exist in the array, return a negative integer. Can you formally specify the correctness of this procedure? Can you prove your implementation is correct?

Listing 1 shows an implementation of the binary search algorithm. I think the correctness of the function can be shown as follows:

Input: An array of integer A, a target integer V.

Precondition: A is sorted; that is, for integers i, j in $[0, A.length - 1]$ and $i < j, A[i] \leq A[j]$.

Postcondition: Return k such that $A[k] = V$ if V in A else return -1.

```
1  def binary_search(A: list[int], V: int) -> int:
2      lower = 0
3      upper = len(A) - 1
4      while upper >= lower:
5          mid = (lower + upper) // 2
6          if V == A[mid]:
7              return mid
8          elif V < A[mid]:
9              upper = mid - 1
10         else:
11             lower = mid + 1
12     return -1
```

Listing 1: A Python implementation of binary search

Proof of correctness:

Claim 1: The value of $upper - lower$ at the ith time line 4 is executed is less than the value of $upper - lower$ at the $(i-1)$th time line 4 is executed for $i > 1$.

Proof: By line 5 we know that $lower \leq mid \leq upper$ is always true. Then the execution of line 9 will strictly decrease the value of $upper$, and the execution of line 11 will strictly increase the value of $lower$. Since line 9 and line 11 are the only two lines that can jump back to line 4 from the loop body, the claim is true.

Claim 2: When line 4 is executed, if V is in the array, its index is in the range $[lower, upper]$.

Proof: Proof by induction. When line 4 is reached for the first time, $[lower, upper]$ covers all the array indices. The statement is trivially true. When line 4 is reached for the ith time, either $A[mid] = V$ or $A[mid] \neq V$. If $A[mid] = V$, mid is returned, and there will be no more execution of line 4. If $A[mid] \neq V$, because A is sorted, the index of V can only be in $[mid + 1, upper]$ if $V > A[mid]$ or in $[lower, mid - 1]$ if $V < A[mid]$. The variables $lower$ and $upper$ are then updated to $mid + 1$ and $mid - 1$ in the two corresponding cases, and the statement holds when line 4 is reached for the $(i + 1)$th time.

If the function is returned from line 7, $A[mid] = V$ and the postcondition of the function holds. Because of claim 1, the function cannot loop forever. If the loop condition at line 4 is evaluated to false, $upper < lower$. Then $[lower, upper]$ is an empty range. By claim 2, since the index of V cannot be in this range, V is not in the array. This is the only case where line 12 can be reached, and this is the only case where -1 is returned. The postcondition of the function holds in this case as well. $\square$