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?

I think the correctness of the function performing the binary search can be as follows:

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

Precondition: A is sorted; i.e. for two integers i, j in the range of $[0, A.length - 1]$ 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: If line 4 is executed more than once, than $upper - lower$ at the ith time line 4 is executed is less than $upper - lower$ 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, the claim is true.

Claim 2: When line 4 is executed, either $upper \geq lower$ or V is not in the array.

Proof: Proof by induction. Base case: when line 4 is reached for the first time, $lower$ and $upper$ are initialized by line 2 and line 3. The statement is trivially true. Inductive step: when line 4 is reached for the ith time, $i > 1$, either $A[mid] = V$ or $A[mid] \neq V$. If $A[mid] = V$, there will not be another execution of the loop, and the function returns mid. If $A[mid] \neq V$, by the inductive hypothesis, $lower \leq upper$. If $lower < upper$, after $upper$ is updated by line 9 or $lower$ is updated by line 11, $lower \leq upper$ still holds. After the bound update, because the array is sorted, V must have an index in the $[lower, upper]$ range if it is in A. If $lower = upper$, $lower = upper = mid$. Then after *upper* is updated by line 9 or *lower* is updated by line 11, $lower \le upper$ no longer holds. In this case, the range is an empty set, so V is not in A.

Because line 12 is only reachable when $lower > upper$, so -1 is only returned when V is not in A by Claim 2. By Claim 1, the loop cannot be executed infinitely many times. So it will terminate and return *mid* when it finds a *mid* such that $A[mid] = V$ if V is in A or return -1 at line 12. Thus the postcondition of the function holds. $\square$