**Writing Correct Code: Binary Search**

**Loop Invariants, Bound functions, Pre and Post-conditions**

Refs: <https://reprog.wordpress.com/2010/04/30/writing-correct-code-part-3-preconditions-and-postconditions-binary-search-part-4c/>

**Invariant:**

1. An invariant is a property that remains true throughout the execution of a piece of code.
2. It’s a statement about the state of a program — primarily the values of variables - that is not allowed to become false.
3. If it becomes false, then the code is wrong.
4. Choosing the correct invariant — one that properly expresses the intent of an algorithm — is a key part of the design of code; and ensuring that the invariant remains true is a key part of the actual coding.
5. **In practice, we formalize the invariant in terms of specific vairables and values that appeared in the algorithm.**

**Bound Function:**

1. The bound function of a loop is defined as an upper bound on the number of iterations still to perform.
2. More generally, we think of it as an expression whose value decreases monotonically as the loop progresses. When it reaches zero (or drops below), we exit the loop.

**Pre-Condition and Post-Condition:**

1. A way to specify the requrements for a function (about the input and output).
2. Precondtion: Some requirement about the input that have to be true.
3. More formally: A precondition is a statement about what must be true about the inputs to a function to guarantee that the function will behave as expected(which is the post condition).

**Binary Search Problem Statement:**

Given an integer X and integers A0, A1, A2, ...,AN-1, which are presorted in ascending order, find i such that Ai = X, or return -1 if X is not in the input. There might be multiple i with Ai = X.

**Invariant For Binary Search:**

1. Informally: “If the saught value ‘X’ is present in the array ‘A’ at all, then it is present in the current range.”
2. To make it useful for deriving correct code: we need to formalise that invariant in terms of specific variables and values.
3. And before that we need to decide on the representation of the range under consideration.
4. There are several candidate representions: None of them greatly better or worse than the others:
5. keep track of the highest or lowest array indexes that might hold ‘X’, or
6. the lowest index and the size of the range ; or
7. use asymmetric indexes, where we maintain to the base of the current range and the index points past the end.
8. fdf
9. fdf