ADw2

Name : **Peter Garamvoelgyi** Student ID: **2018280070**

1. **Write pseudocode for linear search, which scans through the sequence, looking for *v*. Using a loop invariant, prove that your algorithm is correct. Make sure that your loop invariant fulfills the three necessary properties. (CLRS 2.1-3)**

**Input**: A sequence ofnumbers and a value

**Output**: An indexs.t.or the special value nil ifdoes not appear in.

|  |  |
| --- | --- |
| Linear-search:  1 nil  2 **for**  **to**  3 **if** **then**  4  5 **return** | **The loop invariant I for this algorithm:**  At the start of every iteration of the **for** loop, exactly one of the following conditions must be true:   1. nil anddoes not contain, 2. nil and. |

Initialization:

At first, andnil.is the empty array.

The empty array does not contain.

a) is fulfilled, thus **I** is fulfilled.

Maintenance:

We know that at the beginning of every iteration **I** is fulfilled for. Let us consider four cases:

1. a) is fulfilled, i.e.nil anddoes not contain.
   1. Ifthen. b) is fulfilled at the end of this iteration.
   2. Ifthendoes not containandnil. a) is fulfilled at the end of this iteration.
2. b) is fulfilled, i.e.nil and.
   1. Ifthenand b) is fulfilled at the end of this iteration.
   2. Ifthen b) still remains fulfilled at the end of the iteration.

In all these possibilities, **I** remains fulfilled.

Termination:

At the end of the **for** loop we must have.

The invariant holds, i.e. eitherdoes not containandnil, ornil and. This proves that the algorithm is correct.

1. **Consider linear search again (see Exercise 2.1-3). How many elements of the input sequence need to be checked on the average, assuming that the element being searched for is equally likely to be any element in the array? How about in the worst case? What are the average-case and worst-case running times of linear search in-notation? Justify your answers. (CLRS 2.2-3)**

Let us have the following assumptions:

1. Each element in the array is unique.
2. The array contains
3. Our algorithm terminates right after finding.

Letbe a random variable representing the number of elements we have to check until we find the first matching one. The expected value (corresponding to the average-case running time) is

To prove that this is indeed, let us show the corresponding constants:

The worst-case is of course the case where the array does not contain, in which case the number of steps is.

Again, we show the validity of the-constraint by showing the constants:

(Of course, these constants are just examples, there are infinitely many valid solutions.)

We can make our analysis more general by dropping assumption 2., i.e. we also regard the case when the array does not contain. Letdenote the probability that the array contains. This way, the expected value becomes:

… which corresponds to our previous solution when.

Let us regard the case when we drop assumption 1. as well, i.e. the array elements do not have to be unique. Let Y be a random variable representing the position in the array that contains the value. According to the problem statement

… i.e. each position in the array contains the valuewith equal probability. As the array can containmultiple times, these events are independent. E.g.

It is easy to see that

Using this, the distribution ofcorresponds to the geometric distribution

The expected value is given by

1. **Explain why the statement, “The running time of algorithm A is at least” is meaningless. (CLRS 3.1-3)**

**Def. :**

Intuitively,gives an (asymptotic) upper bound. A function cannot be larger than its upper bound.

Letdenote the running time of algorithm A for input size.

means

At the same time,is *at least*suggests:

These two constraints can only be fulfilled simultaneously if

This is, however, a very special case (and this is just one interpretation of an otherwise rather ambiguous sentence). We can conclude that saying that an algorithm’s running time is at leastis meaningless.

1. **Show that .**

**Def. :**

In our case:

Our goal is to show three constantsfor which the definition above holds. Let us consider the inequalities

Rearranging, we get

Let and . The first inequality then becomes

… i.e.is fulfilled for any positive value. The second inequality becomes

… i.e. we get a quadratic function with roots and . The inequality is fulfilled when or . For us, this means that our constraint is fulfilled for all . From the above derivation, we get the constants

… for which

**Thus, holds.**