

# Exercise answers - Chapter 2 - Getting Started

Christian Duffau-Rasmussen

April 18, 2018

## 2.1-3 Linear Search

```
1: procedure LINEAR-SEARCH( $A, v$ )
2:   for  $i \leftarrow 1$  to  $\text{length}[A]$  do
3:     if  $A[i] == v$  then
4:       return  $i$ 
5:   return NILL
```

**Proof of correctness** To prove linear search is correct, I formulate a *loop invariant* that need to hold true at *initialization*, during *maintainance* and at *termination*.

**Loop invariant** At the start of loop iteration  $i$ , the value  $v$  is not in the sub-array  $A[1..i-1]$ . In formal terms,  $v \notin \{A[j] | j \in \{1, \dots, i-1\}\}$

**Initialization** At initialization  $i = 0$ , so the sub-array  $A[1..i-1]$  is empty. So  $v$  is trivially not a member of that array, and the loop invariant holds.

**Maintenance** Assuming that  $v$  is not in the sub-array  $A[1..i-1]$  at the start of iteration  $i$ , then two outcomes are possible during the iteration. Either,  $A[i] == v$  and the loop terminates, or  $A[i] \neq v$  and  $v$  is not an element of the array  $A[1..i]$  which is the loop invariance condition for iteration  $i+1$ .

**Termination** The algorithm terminates in two different ways. The first happens when  $A[i] == v$ , assuming  $v \notin A[1..i-1]$  before termination, then the loop invariant still holds. Again, assuming  $v \notin A[1..i-1]$  before termination. In the second case we have that  $i = n+1$ , in that case  $v \notin A[1..i-1] = A[1..n]$ , which is the entire array, and the procedure returns NILL.

## 2.2-3 Running time of Linear search

Assuming the index of correct element is uniformly distributed from 1 to  $n$ , the expected index is  $E[i] = n/2$ . So the average case running time must be  $c_1 \frac{n}{2} + c_2$ , where  $c_1$  is the running time of each comparison  $A[i] == v$  and the loop increment, and  $c_2$  is the running time of the return statement. In the worst case

the algorithm needs to run through all  $n$  elements as well as returning NIL, so the running time is  $c_1n + c_2$ . Both the average and worst case running times of linear search are  $\Theta(n)$ .

### 2.3-3 Linear Recurrence

To show that the recursion

$$T(n) = \begin{cases} 2 & \text{if } n = 2 \\ 2T(n/2) + n & \text{if } n = 2^k, \text{ for } k > 1 \end{cases}$$

is  $T(n) = n \lg n$ . Assuming that  $n$  is a power of two we have  $n = 2^k$  for  $k \in 1, 2, \dots$ . For  $k = 1$  we have that  $T(n) = T(2) = 2$  by definition of  $T$ .

$$2 \lg(2) = 2 = T(2).$$

Hence the claim holds for  $k = 1$ .

Next, assuming that  $T(2^{k-1}) = 2^{k-1} \lg 2^{k-1} = (k-1)2^{k-1}$  we have that,

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2T(2^k/2) + 2^k \\ &= 2T(2^{k-1}) + 2^k \\ &= 2((k-1)2^{k-1}) + 2^k \\ &= k2^k \\ &= 2^k \lg(2^k) = n \lg n \end{aligned}$$

so the claim also holds for  $k$  given it holds for  $k-1$ , at since it holds true for  $k=1$  it holds for any  $k \geq 1$ .  $\square$

### 2.3-5 Binary search

```

1: procedure BINARY-SEARCH(A, p, r, v)
2:   if  $p \leq r$  then
3:      $m = p + \lfloor \frac{r-p}{2} \rfloor$ 
4:     if  $A[m] == v$  then
5:       return m
6:     else if  $A[m] > v$  then
7:       return BINARY-SEARCH(A, p, m - 1, v)
8:     else if  $A[m] < v$  then
9:       return BINARY-SEARCH(A, m + 1, r, v)
10:  return NIL
```

In the worst case, binary search needs to split the array in two, until the last split goes from two elements to a single one. A way of assessing the running time is to understand how many possible splits can be made. Assuming the length of the array is  $n = 2^k$ , the length after the first split is simply  $n/2$ . Finding

the length of the consequent halved arrays, amounts to raising the *half* to the number of splits and multiplying by the initial size,  $n$ . So, to find the number of splits which result in the length of 1, we need to solve the following,

$$1 = \frac{n}{2^x} \quad \Leftrightarrow \quad 2^x = n \quad \Leftrightarrow \quad x = \lg n.$$

So we can conclude, that the recursion calls itself at most  $\lg n$  times, this is an indication of the worst case running time being  $\Theta(\lg n)$ .