Topic 2a, Linear Search: Solutions

Searching Problem

Input:

A sequence of *n* numbers $A = \langle a_1, a_2, ... a_n \rangle$ (in <u>no particular order)</u> and a value *v*.

Output:

An index *i* such that *v* = A[*i*] or the special value NIL if *v* does not appear in *A*.

(a) Write pseudocode for **linearSearch(A,v)**, an algorithm that scans through a sequence represented as an array *A* looking for *v*, and provides the desired output (*i* or NIL). Use Java style 0-based indexing, A[i] to access elements, and A.length to get the number of elements *n*. Number the lines for reference, as started below.

```
linearSearch(A, v)
1 for i = 0 to A.length - 1
2   if A[i] == v
3     return i
4 return NIL
```

(b) Let ***p*** be the number of array positions checked. (*p will be replaced with a function of n: p = f(n). For example, when the element is not present, p = n.*) Let c_1 be the cost of executing line 1, c_2 the cost of executing line 2, etc. As was done in the lecture notes:

- Write the expression for the cost to execute each line in terms of *p* and the c_i.
- Sum these to get the total cost *T*(*n*).
- Then collect the constants and rename them "a" and "b".

```
linearSearch(A,v)                  //cost     times
1 for i = 0 to A.length - 1         // $c_1$     p+1 // +1 for exit
2   if A[i] == v                    // $c_2$     p
3     return i                      // $c_3$     1 or 0
4 return NIL                        // $c_4$     0 or 1
```

Only one of c_3 and c_4 will be paid, so let c_5 = max(c_3, c_4)

```
T(n) = c_1*(p+1) + c_2*p + c_5
     = c_1*p + c_1 + c_2*p + c_5
     = (c_1+c_2)*p + (c_1+c_5)

Let a = (c_1+c_2) and b = (c_1+c_5)

T(n) = a*p + b
```

(c) As a function of *n*, what is *p* for the worst case: precisely how many elements of the input sequence need to be checked?

- Rewrite your last expression for this *p*.
- In what Θ (Theta) complexity class is this? (Write as Θ(____).)

Justify your answers!

In the worst case p = n, so an+b = Θ(n)

(d) As a function of *n*, what is *p* for the average case: precisely how many elements of the input sequence need to be checked on average, assuming that the element being searched for is present and equally likely to be any element in the array?

- Rewrite your expression for this *p*.
- In what Θ (Theta) complexity class is this? (Write as Θ(____).)

Justify your answers!

The possible runtimes are
1 step if v is the first element,
2 steps if v is the second element,
3 steps if v is the third element,
.
.
.
n steps if v is the nth **element**

Since each step is equally likely to happen, the average runtime is

$$\frac{1+2+3+\ldots(n-1)+n}{n} = \frac{(n+1)*n}{2n} = \frac{n+1}{2}$$

This is Θ(n).

Clarifications:

- We are not assuming any ordering of the numbers
- Return any location (the first instance) where the key is found, not all the locations
- Use "Java-like" pseudocode, but don't worry about details of syntax
- Analytic variable "*p*" is number of positions checked: replace it with functions of n in the answers to (c) and (d).
- Example: In the best case, the value *v* you are looking for is in the first array element, so *p*=1.