CMSC351-2023-09-25

Monday, September 25, 2023 8:28 AM

B I N A R Y   S E A R C H

① **Intro**: Suppose A is a sorted list of length n containing distinct elements. Given a specific target element (which may or may not be in the list), how could we find that element (or fail)?

- Linearly - scan the entire list? $\Theta(n)$
- something better?! YES! :)

② **Idea**: Start by checking the elt. at the middle of A.
if that's our target, rejoice! Done!
if not then:
    if target > middle elt, look in the right half of A
    if target < " ", " " " left " " A.
Repeat w/ the chosen half, until we find it, or fail.

ex $A = [1, 3, 5, 7, 10, 40, 50]$
if target = 7 we check middle of A, find 7, rejoice!
if target = 40 we note $40 > 7 =$ middle, so look right!
now we look at [10, 40, 50] etc.

③ **Pseudocode**:

```
\\ PRE: A is a sorted list of length n.
\\ PRE: TARGET is a target element.
function binarysearch(A,TARGET)
    L = 0                                   } c1
    R = n-1
    while L <= R                            ??? iterations
        C = floor((L+R)/2)                  }
        if A[C] == TARGET                   }
            return C                        }
        elif TARGET < A[C]                  } c2
            R = C-1                         }
        elif TARGET > A[C]                  }
            L = C+1                         }
        end                                 }
    end while
    return FAIL                             c3
end
\\ POST: Value returned is either the index or FAIL.
```

Found!

Look left!

Look right!

Hmm!

if we reach L==R and don't find TARGET, after that iteration we will have L>R and the loop will fail then we RETURN FAIL!

Note: This is decrease and conquer b/c the list size decreases!

Note: This is decrease and conquer b/c the list size decreases!

④ TIME!

(A) Best-Case: If TARGET is the first elt we check time is:

$$T(n) = c_1 + c_2 = \Theta(1)$$

(B) Worst-Case: If TARGET is not in the list!

Before while loop, list has length n.
After 1 iteration, length is $n/2$

2	$n/4$
⋮	
k	$n/2^k$

The list will have length 1 when $\frac{n}{2^k} = 1$ so $n = 2^k$ so $k = \lg n$
So after lg n iterations we have length 1. i.e. L == R
But, then it does one more iteration! So total = $1 + \lg n$ iterations!
So

$$T(n) = c_1 + (1 + \lg n)c_2 + c_3 = \Theta(\lg n)$$

(C) Avg Case!

Q: what do we mean by avg. case?
A: We'll say: Imagine a sorted list of length n.
We pick one elt. at random, uniformly (each = likely)
run alg. searching for that elt.
Take the expected value over all elts.

To ease cals, focus on lists w/ length $n = 2^N - 1$ w/ $N \in \mathbb{Z}^+$.

i.e. $N=1$: $n=1$
$N=2$: $n=3$
$N=3$: $n=7$
etc.

- Okay, so look at N=1 case: while loop runs 1 time.
- Now, look at N=2 case: there is one middle elt. Where while loop iterates once.

- Okay, so look at N=1 case: while loop runs 1 time.
- Now, look at N=2 case: there is one middle elt. Where while loop iterates once. If it's not that one then we look left or right. Each sublist is a copy of the N=1 case so twice as many elts as N=1 case w/ an add'l iteration each!

#elts	#iterations
1	1
2	2

*

N=3

- Again! One middle elt w/ 1 iteration. If not that then we have two copies of the N=2 case w/ 1 more iteration each!

N=3

#e	#it
1	1
2	2
4	3

* w/ #elts doubled
#its incremented

- Again! N=4

#e	#it
1	1
2	2
4	3
8	4

In general: N

#e	#it
1	1
2	2
4	3
⋮	⋮
2^{N-1}	N

now then, if i iterations are required then time = $c_1 + ic_2$

okay, in addition, the prob. of picking an elt. which takes some # of iterations is #elts/n.

Thus:

prob	#its	time
$\frac{1}{n}$	1	c_1+1c_2
$\frac{2}{n}$	2	c_1+2c_2
$\frac{4}{n}$	3	c_1+3c_2

$\frac{2}{n}$	2	C_1+2C_2
$\frac{4}{n}$	3	C_1+3C_2
$\vdots$	$\vdots$	$\vdots$
$\frac{2^{N-1}}{n}$	N	C_1+NC_2

Exp value is then $\sum_{i=0}^{N-1}\left(\frac{2^i}{n}\right)(C_1+iC_2) = \ldots = \Theta(\lg n)$