CSC 349-07: Design and Analysis of Algorithms **Winter 2023**

P vs NP: March 7, 2023

Instructor: Rodrigo Canaan *Scribe: Ishaan Sathaye*

1 Announcements

Make sure to write all announcements here.

- Dr. Migler reminded everyone to be healthy and kind and to sign up for a scribe day.
- Second bullet point
- Third bullet point

2 Searching

2.1 Problem Statement

Search
Input: A list of numbers, sorted in increasing order, $A = [a_0, a_1, a_2, a_3, \ldots a_{n-1}]$ and a number x.
Goal: Return a position of x in A if $x \in A$ or return a statement that x is not in the list.

2.2 Pseudocode for BinarySearch

BinarySearch(A, x, min, max)

Input: An array of n sorted numbers A, number x, integers min, and max. (Initially $min = 0$ and $max = n - 1$)
Output: A position of x in A if $x \in A$, or -1 if $x \notin A$.

```
1: if max ¡ min then
       return −1
2: else
3:     if x < A[⌊(max + min)/2⌋] then
         return BinarySearch(A, x, min, ⌊(max + min)/2⌋ − 1)
4:     else if x > A[⌊(max + min)/2⌋] then
         return BinarySearch(A, x , ⌊(max + min)/2⌋ + 1, max)
5:     else
         return ⌊(max + min)/2⌋
```

2.3 Correctness of BinarySearch

Notice that in the output statement for BinarySearch we said that **A** position of x is returned because we don't know which position of x will be returned (if x occurs multiple times). For example if we call BinarySearch($\{2, 2, 3, 5, 5, 7\}$, 5, min, max) then the second occurrence of 5 is returned.

Also notice that for BinarySearch(A, x, min, max) we aren't "throwing away" half of the list at every iteration. Rather, we are reducing the size of the interval under consideration ((min, max)).

For the base case: if $max < min$ then the interval that we are considering is empty, thus x couldn't possibly be in that interval.

Again, we must prove that BinarySearch(A, x, min, max) is correct. We will prove the following lemmas:

Lemma 1 *If $x \in A$, then* BinarySearch(A, x, min, max) *returns a position of x in A.*

Lemma 2 *If $x \notin A$, then* BinarySearch(A, x, min, max) *returns -1.*

We will prove Lemma 2 by contradiction:

Proof: Suppose $x \notin A$, further suppose for a contradiction that BinarySearch(A, x, min, max) returns something other than -1. Note that the only time that BinarySearch(A, x, min, max) returns something other than -1 is in line 5 of the algorithm. The only way to enter line 5 of the algorithm is if $x = A[\lfloor (max + min)/2 \rfloor]$. However, if $x \notin A$ then this will never occur. We have reached our contradiction. Of course we should also point out that our algorithm terminates... *Why?* ■

Now for the proof of Lemma 1. Divide and conquer is a recursive technique and therefore induction is usually the proof technique for the proofs of correctness for these types of algorithms. We will restate our lemma to make it more suitable for a proof by induction. Note that we are going to use induction over the size of the interval that we are considering ((min, max)).

Lemma 3 *If $x \in A(min, max)$ and $max - min = n$, then* BinarySearch(A, x, min, max) *returns a position of x in A.*

Proof: We proceed by induction of the size of the interval under consideration.

Base Case: Suppose that $x \in A(min, max)$ and $max - min = 0$. Then $min = max = \lfloor (max + min)/2 \rfloor$. Also, since $x \in A(min, max)$, it must be that $x = A[min] = A[max] = A[\lfloor (max + min)/2 \rfloor]$. So line 5 of BinarySearch(A, x, min, max) returns $min = max = \lfloor (max + min)/2 \rfloor$.

(Strong) Inductive Hypothesis: If $x \in A(min, max)$ and $max - min \leq k$, then BinarySearch(A, x, min, max) returns a position of x in A.

Inductive Step: Suppose $x \in A(min, max)$ and $max - min = k + 1$. There are three possibilities:

1. $x < A[\lfloor (max + min)/2 \rfloor]$
2. $x > A[\lfloor (max + min)/2 \rfloor]$
3. $x = A[\lfloor (max + min)/2 \rfloor]$

Case 1: Because the list is sorted and x is in the interval $A[(min, max)]$, it must be that x is in the first half of the interval, $A[(min, \lfloor (max + min)/2 \rfloor - 1)]$.

What does BinarySearch(A, x, min, max) do in this case?

In this case, in line 3, BinarySearch(A, x, min, max) returns BinarySearch(A, x, min, $\lfloor (max + min)/2 \rfloor - 1$).

Note that $x \in A[(min, \lfloor (max + min)/2 \rfloor - 1)]$ and $(\lfloor (max + min)/2 \rfloor - 1) - min \leq k$, so our inductive hypothesis is met and BinarySearch(A, x, min, max) returns a position of x in A.

Case 2: Symmetric to Case 1: because the list is sorted and x is in the interval from $A[(min, max)]$, it must be that x is in the second half of the interval, $A[(\lfloor (max + min)/2 \rfloor + 1, max)]$.

What does BinarySearch(A, x, min, max) do in this case?

In this case, in line 4, BinarySearch(A, x, min, max) returns BinarySearch(A, x, $\lfloor (max + min)/2 \rfloor + 1, max$).

Note that $x \in A[(\lfloor (max + min)/2 \rfloor + 1, max)]$ and $max - (\lfloor (max + min)/2 \rfloor + 1) \leq k$, so our inductive hypothesis is met and BinarySearch(A, x, min, max) returns a position of x in A.

Case 3: If $x = A[\lfloor (max + min)/2 \rfloor]$, then line 5 of BinarySearch(A, x, min, max) correctly returns $\lfloor (max + min)/2 \rfloor$.

Therefore by the principle of mathematical induction, we have the result.

■

Notice that we used *strong induction* in the above proof. This is because the size of the problem is being reduced by about $\frac{n}{2}$ at each stage, so having information about a problem of size $n - 1$ (which you would get from weak or regular induction) would not be useful.

We have proven that BinarySearch(A, x, min, max) is correct. The next step is to evaluate the running time.