CS112 Data Structures

Recitation 05

Yu Yang
yy388@cs.rutgers.edu
Office: CoRE 331
Office Hour: 3-5pm, Wed.

1. Given the following sequence of integers:

```
3, 9, 2, 15, -5, 18, 7, 5, 8
```

   1. What is the average number of comparisons for a successful search assuming all entries are searched with equal probability? Show your work.
   2. Suppose the search probabilities for the elements of this list are, respectively:

      ```
      0.1, 0.3, 0.05, 0.2, 0.05, 0.1, 0.05, 0.1, 0.05
      ```

      What is the average number of comparisons for successful search with these search probabilities? Show your work.
   3. Rearrange the list entries in a way that would result in the lowest number of comparisons on the average for successful search, given the above probabilities of search. What is this lowest number of comparisons? Show your work.

RUTGERS

- In a sequential search of a list, it takes one comparison to succeed at the first element, two comparisons to succeed the second element, and so on. In general it takes *i* comparisons to succeed at the *i*-th element. With the assumption that all entries are searched with equal probability, the formula is:

```
        (1 + 2 + 3 + 4 + ... + n) / n = n*(n + 1)/2*n = (n + 1)/2 = (9+1)/2 = 5
```

- If the search probabilities are changed according to the given example, the average # of comparisons should be computed as follows:

```
0.1 x 1 + 0.3 x 2 + 0.05 x 3 + 0.2 x 4 +
0.05 x 5 + 0.1 x 6 + 0.05 x 7  + 0.1 x 8 + 0.05 x9 = 4.1
```

- We should rearrange the entries such that entries with high search probabilities come first in the list. For example the entry "9" should be the first item since it has the highest search probability. Following this procedure, the new list should be arranged like this:

```
   9 15 {3,5,18} {2,-5,7,8}
```

  The entries in the brackets can be arranged in an arbitrary order since they have the same search probabilities.

3. Draw the comparison tree for binary search on a sorted array of length 11.
    1. What is the worst case number of comparisons for success? For failure?
    2. What is the average number of comparisons for success, assuming equal likelihood of success at any spot?
    3. What can you say about the average number of comparisons for failure? Can you approximate it within a small range? Does it depend on the distribition of the probabilities of failing at the various failure spots?

SOLUTION

For the comparison tree, the nodes have the index positions where comparisons are made:

```
              (5) =
       < /           \ >
    = (2)             (8) =
   < /   \ >       < /   \ >
= (0)     (3) =  = (6)     (9) =

< / \ > < / \ >  < / \ > < / \ >
 F (1)=  F (4)=   F (7)=  F (10)=
  < /\ >  < /\ >  < /\ >    < /\ >
   F  F    F  F    F  F     F  F
```

1. Worst case #comparisons for success = 7 (for positions 1, 4, 7, 10), worst case #comparisons for failure = 8 (for failures off the positions 1, 4, 7, 10)

2. Average #c for success = (1*1 + 2*3 + 4*5 + 4*7)/11 = 5

3. It would take 6 comparisons to get to any of the failure nodes at the last but one level, and 8 comparisons for the last. So the average MUST be between 6 and 8, no matter what the probability distribution is over all of these possibilities.

4. * A variant of binary search, called *lazy* binary search, works as described in the following algorithm, where t is the target to search, and n is the size of the array:

```
left <-- 0
right <-- n-1
while (left < right) do
   mid <-- (left + right)/2
   if (t > A[mid]) then
      left <-- mid + 1
   else
      right <-- mid
   endif
endwhile
if (t == A[left]) then
   display "found at position", left
else
   display "not found"
endif
```

1. Trace this algorithm on the following array, with 46 as the search target:

```
10   15   25   30   45   46   48   72   76   80   93
```

   How many comparisons are made by the time a match is found? How does your answer compare with that for regular binary search?

2. Repeat with 40 as the target. How many comparisons are made until failure is detected? How does your answer compare with that for regular binary search?

3. Draw the comparison tree for lazy binary search on an array of length 11 (same length as the example array above).
   1. What is the worst case number of comparisons for success? For failure?
   2. What can you say about the range of values for the average number of comparisons for success? For failure?
   3. Under what conditions is it preferable to use lazy binary search over the regular binary search?

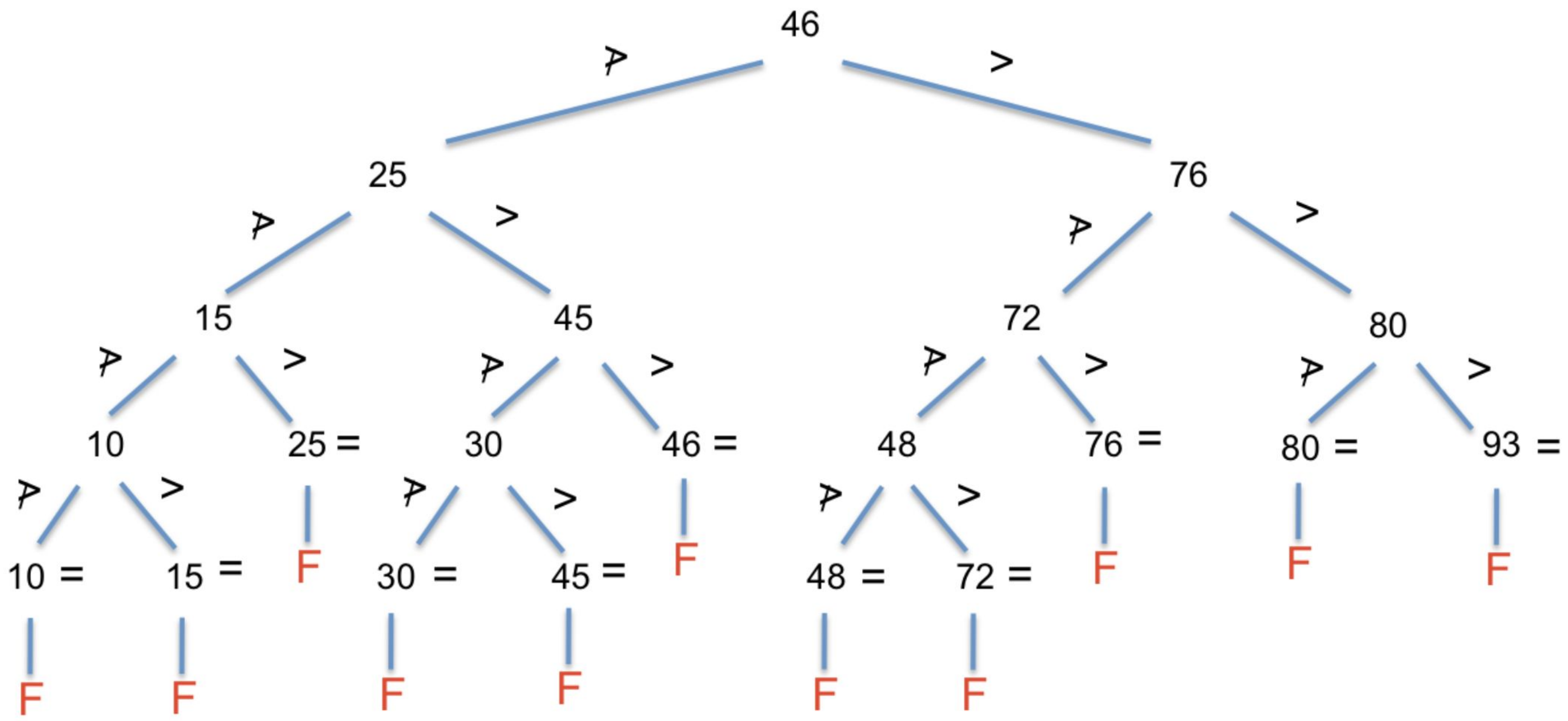
1. Trace this algorithm on the following array, with 46 as the search target:

```
10    15    25    30    45    46    48    72    76    80    93
```

How many comparisons are made by the time a match is found? How does your answer compare with that for regular binary search?

2. Repeat with 40 as the target. How many comparisons are made until failure is detected? How does your answer compare with that for regular binary search?

3. Draw the comparison tree for lazy binary search on an array of length 11 (same length as the example array above).
    1. What is the worst case number of comparisons for success? For failure?
    2. What can you say about the range of values for the average number of comparisons for success? For failure?
    3. Under what conditions is it preferable to use lazy binary search over the regular binary search?



RUTGERS

First Midterm Review