Math Concepts in Computing II CIS 2166: Hw 1

Hai Dang

September 2, 2016

3.1.14) List all the steps used to search for 7 in the sequence given in Exercise 13 for both a linear search and a binary search.
1, 3, 4, 5, 6, 8, 9, 11

(a) Linear Search:
$1 = 7$
1. Set counter to 1.
2. Compare counter 1 to size of list 8.
3. Then compare 7 to 1, the value in the array.
4. Since counter $1 \leq 8$ and values $7 \neq 1$ then increment counter by 1.
5. Set counter to 2.
6. Compare counter 2 to size of list 8.
7. Then compare 7 to 3, the value in the array.
8. Since counter $2 \leq 8$ and values $7 \neq 3$ then increment counter by 1.
9. Set counter to 3.
10. Compare counter 3 to size of list 8.
11. Then compare 7 to 4, the value in the array.
12. Since counter $3 \leq 8$ and values $7 \neq 4$ then increment counter by 1.
13. Set counter to 4.
14. Compare counter 4 to size of list 8.
15. Then compare 7 to 5, the value in the array.
16. Since counter $4 \leq 8$ and values $7 \neq 5$ then increment counter by 1.
17. Set counter to 5.
18. Compare counter 5 to size of list 8.
19. Then compare 7 to 6, the value in the array.
20. Since counter $5 \leq 8$ and values $7 \neq 6$ then increment counter by 1.
21. Set counter to 6.
22. Compare counter 6 to size of list 8.
23. Then compare 7 to 8, the value in the array.
24. Since counter $6 \leq 8$ and values $7 \neq 8$ then increment counter by 1.
25. Set counter to 7.
26. Compare counter 7 to size of list 8.
27. Then compare 7 to 9, the value in the array.
28. Since counter $7 \leq 8$ and values $7 \neq 9$ then increment counter by 1.
29. Set counter to 8.
30. Compare counter 8 to size of list 8.
31. Then compare 7 to 11, the value in the array.
32. Since counter $8 \leq 8$ and values $7 \neq 11$ then increment counter by 1.
33. Since counter 9 ¿ size of list 8. Exit the loop.
34. Compare the counter 9 to the size of list 8.
35. Since $9 > 8$, number not found in array: return 0

(b) Binary Search:

1. Set counter i to 1 for the lowest index.
2. Set counter j to 8 being the size of the largest array index.
3. Compare the counters with values 1 and 8.
4. Since for the counters $1 < 8$, enter the loop.
5. Compute midpoint as the floor of $(1+8)/2$ which is 4
6. Then compare the value 7 to 5, the value in the array.
7. Since $7 > 5$, set the i counter to 5.
8. Compare the counters with values 5 and 8.
9. Since for the counters $5 < 8$, enter the loop.
10. Compute midpoint as the floor of $(5 + 8)/2$ which is 6.
11. Then compare the value 7 to 8, the value in the array.
12. Since $7 < 8$, set the j counter to 6.
13. Compare the counters with values 6 and 8.
14. Since for the counters $6 < 8$, enter the loop.
15. Compute midpoint as the floor of $(6 + 8)/2$ which is 7.
16. Then compare the value 7 to 9, the value in the array.
17. Since $7 < 9$, set the j counter to 7.
18. Compare the counters with values 7 and 7.
19. Since 7 is not < 7 exit the loop.
20. Compare 7 with array value 9 at location 7.
21. Since $7 \neq 9$, number not found in array: return 0.

Algorithm 1 Quaternary Search

3.1.28) Specify the steps of an algorithm that locates an element in a list of increasing integers by successively splitting the list into four sublists of equal (or as close to equal as possible) size, and restricting the search to the appropriate piece.

```
procedure (x : integer, a_1, a_2, ..., a_n : increasing integers)
    i ← 1
    j ← n
    while i < j do
        lq = ⌊(i + j)/4⌋                    ▷ Lower Quarter
        mid = ⌊(i + j)/2⌋                   ▷ Middle
        uq = ⌊3(i + j)/4⌋                   ▷ Upper Quarter

        if x > a[mid] then                  ▷ Upper half, greater than the midpt
            if x > a[uq] then
                i ← uq + 1                  ▷ range is from i = uq+1 to j = rightmost
            else
                i ← m + 1                   ▷ range is from i = m+1 to j = uq
                j ← uq
            end if
        else                                ▷ Lower half, less than and equal to midpt
            if x > a[lq] then
                i ← lq + 1                  ▷ range is from i = lq+1 to j = m
                j ← m
            else
                j = lq                      ▷ range is from i = leftmost to j = lq
            end if
        end if

    end while

    if x = a_i then
        location ← i                        ▷ Set location to index i
    else
        location ← −1                       ▷ x is not found in list
    end if
return location
end procedure
```

Algorithm 2 Binary Search Insert

3.1.44) Describe an algorithm based on the binary search for determining the correct position in which to insert a new element in an already sorted list.

```
procedure (x : integer, a_1, a_2, ..., a_n : increasing integers)
    i ← 1
    j ← n
    while i < j do
        m = ⌊(i + j)/2⌋                                  ▷ Middle
        if x > a_m then
            i ← m + 1
        else
            j ← m
        end if
    end while

    if x = a_i then
        location ← i                                     ▷ Set location to index i
    else
        location ← i                                     ▷ Get the position where x should be inserted
        for i ← n + 1 to location, decrement i do
            arr_i ← arr_{i−1}                            ▷ Copy over values at higher index to lower index
                                                         ▷ to occupy i+1 to n+1
        end for
        arr_location ← x                                 ▷ Insert value x into the empty
    end if
end procedure
```