Complexity Analysis II

Parinya Suwansrikham
Software Engineering
College of Arts, Media and Technology
Chiang Mai University

Agenda

- Algorithms
  - Binary Search
- Complexity of Algorithms

Algorithm

- Binary Search Algorithm is used when the list has terms occurring in order of increasing size (list of number from smallest to largest)
- It proceeds by comparing the element to be located to the middle term of the list.
- The list is then split into two smaller sublists of the same size, or where one of these smaller lists has one fewer term than the other.

Binary Search Algorithm

Example 3. To search for 19 in the list

1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

Solution:

1. Split this list into 2 smaller lists

   1 2 3 5 6 7 8 10   12 13 15 16 18 19 20 22
2. Compare 19 and the largest term in the first list (10). Since 10 < 19, the search is limited to second sublist.
3. Split this list, which has 8 terms, into the 2 smaller lists of 4 terms each.

   12 13 15 16   18 19 20 22

Binary Search Algorithm

4. Compare 16 and 19. Since 16 < 19 the search is restricted to the second of these lists.
5. Split the list 18 19 20 22 into 2 lists

   18 19 20 22
6. Since 19 is not greater than the largest term of the first of 2 lists, the search is restricted to the first list: 18 19,
7. Split the list 18 19 into 2 lists

   18 19
8. Compare 18 and 19. Since 18 < 19, the search is restricted to the second list.

Binary Search Algorithm

ALGORITHM 1. The Binary Search Algorithm
METHOD binary_search(x: integer, a_1, a_2, … a_{n-1}, a_n : increasing integers)
SET i = 1 // i is left endpoint of search interval
SET j = n // j is right endpoint of search interval
WHILE i < j
    $m = \lfloor (i + j)/2 \rfloor$
    IF x > a_m
        THEN i = m + 1
      ELSE
        j = m
    ENDIF
ENDWHILE
IF x = a_i
    THEN location = i
ELSE
    location = 0
//

Complexity of Algorithm

- How can the efficiency of algorithm be analyzed?
  - Measure in time
  - Measure amount of computer memory
- This question involves **computational complexity** of the algorithm.
  - Time complexity: analysis of the time required to solve a problem.
  - Space complexity: analysis of the computer memory required to solve a problem.

Complexity of Algorithm

- Space complexity is tied with particular data structures used to implement the algorithm.
- Time complexity can be expressed in term of the number of operations.
  - Comparison of integers
  - The addition of integers
  - The multiplication of integers
  - The division of integers
  - or any other basic operations

Complexity of Algorithm

Example 2. Describe the time complexity of finding the maximum in a set

Solution:

ALGORITHM 1. Finding the Maximum Element in a Finite Sequence.

method *findmax*(a_1, a_2, … a_{n-1}, a_n : integers)

$max = a_1$;

for $i = 1$ to n

    if $max < a_i$ **then** $max = a_i$

Complexity of Algorithm

1. The number of comparison will be used as the measure of the time complexity of the algorithm.
2. Set temporary maximum to the first term in the list.
3. Comparison to determine the end of the list.
4. Comparison between temporary maximum and second term. Update the temporary maximum to the value of the second term if it is larger.

Complexity of Algorithm

5. This procedure is continued, using 2 comparisons for each term of the list

5.1 Comparison to determine that the end of the list

5.2 Comparison to determine whether to update the temporary maximum

For each of the 2nd through the nth elements

Then the number of comparison is $2(n-1) + 1 = 2n - 1$.

The finding maximum algorithm is $O(n)$.

Complexity of Algorithm

Example 3. Describe the time complexity of the linear search algorithm

Solution:

ALGORITHM 2. The Linear Search Algorithm

METHOD linear_search(x: integer, a_1, a_2, … a_{n-1}, a_n: distinct integers)

SET $i = 1$

WHILE $i \le n$ and $x \ne a_i$

SET $i = i + 1$

ENDWHILE

IF $i \le n$

location $= i$

ELSE

location $= 0$

ENDIF

Complexity of Algorithm

1. The number of comparison will be used as the measure of the time complexity of the algorithm.
2. Each step 2 comparisons are performed

   2.1 Determine the ending of the list

   2.2 Compare the element x with a term of the list

   2.3 One more comparison is made outside the loop

3. If $x = a_i$, there is $2i + 1$ comparison.

The most comparison is $2n + 2$, when the element is not in the list. This is call **worst-case analysis**.

Complexity of Algorithm

Example 4. Describe the time complexity of the binary search algorithm

Solution:

1. Assume $n = 2^k$ elements in the list $a_1, a_2, \ldots a_{n-1}, a_n$ where k is a positive integer.
2. At the first stage the search is restricted to a list with 2^{k-1} terms. Two comparisons have been used.
3. At most $2k + 2 = 2\log n + 2$ comparisons are required to perform a binary search.

The time complexity of the binary search algorithm is $O(\log n)$

Complexity of Algorithm

TABLE 1 Commonly Used Terminology for the Complexity of Algorithms.

Complexity	*Terminology*
$O(1)$	Constant complexity
$O(\log n)$	Logarithmic complexity
$O(n)$	Linear complexity
$O(n \log n)$	$n \log n$ complexity
$O(n^b)$	Polynomial complexity
$O(b^n)$, where $b > 1$	Exponential complexity
$O(n!)$	Factorial complexity

Complexity of Algorithm

TABLE 2 The Computer Time Used by Algorithms.

Problem Size	*Bit Operations Used*					
n	$\log n$	n	$n \log n$	n^2	2^n	$n!$
10	3×10^{-9} s	10^{-8} s	3×10^{-8} s	10^{-7} s	10^{-6} s	3×10^{-3} s
10^2	7×10^{-9} s	10^{-7} s	7×10^{-7} s	10^{-5} s	4×10^{13} yr	*
10^3	1.0×10^{-8} s	10^{-6} s	1×10^{-5} s	10^{-3} s	*	*
10^4	1.3×10^{-8} s	10^{-5} s	1×10^{-4} s	10^{-1} s	*	*
10^5	1.7×10^{-8} s	10^{-4} s	2×10^{-3} s	10 s	*	*
10^6	2×10^{-8} s	10^{-3} s	2×10^{-2} s	17 min	*	*