COMP 3760: Algorithm Analysis and Design

Lesson 7: Binary Search

Rob Neilson

rneilson@bcit.ca

Homework and Reading

- This stuff is due in lab in the week of Sept29-Oct3…
  - Read chapters 7.1, 7.2, 7.3
- Homework…
  - Chapter 7.1, page 253, questions 2, 3, 4
  - Chapter 7.2, page 264, questions 2
  - Chapter 7.3, page 270, questions 1, 7

Searching (Review of Search Topics Covered To Date)

- So far we have considered brute force search algos for
  - linear search
  - string matching
  - exhaustive search
    - and in looking at this we considered a couple algos to generate permutations
- Now we want to consider ***Binary Search***
  - this is not a brute force algorithm, rather, it makes efficiency gains by *ignoring* part of the input
  - it works by repeatedly dividing the input in half, and just searching in the half where the target should exist

we call this type of algorithm a **Divide and Conquer Algorithm**

PS: I am sure you remember this, but binary search only works on sorted arrays

- *it doesn’t work on unsorted data*
- *it doesn’t work on linked lists or sets or maps*

Binary Search Exercise 1

- See attached handout for question

- Soln for w=81
  - iteration 1: x,y,z = 11,20,10
  - iteration 2: x,y,z = 11,20,15 ** target found

- Soln for w=22
  - iteration 1: x,y,z = 0,9,10
  - iteration 2: x,y,z = 5,9,4
  - iteration 3: x,y,z = 5,6,7
  - iteration 4: x,y,z = 6,6,5
  - iteration 5: x,y,z = 6,6,6 ** target found

- Question: largest number of iterations is 6, when the target is not found in the array being searched

Binary Search Exercise 2

- See attached handout for question

- Soln for w=90

Call trace:

1. binarySearch(a, 90, 0, 20)
1.1 binarySearch(a, 90, 11, 20)
1.1.1 binarySearch(a, 90, 16, 20)
1.1.1.1 binarySearch(a, 90, 16, 17)
*1.1.1.1.1 binarySearch(a, 90, 17, 17) **target found, returns*

Sample Exam Question

If you have a solution, please post it to the class discussion forum.

a. Provide pseudocode for an algorithm to rearrange elements of a given array of ***n*** real numbers such that all the negative numbers precede all the positive numbers. Marks will be awarded based on the efficiency of your algorithm (ie: the more efficient it is, the higher your grade).

b. Does your algorithm use brute force, divide and conquer, decrease and conquer, transform and conquer, or something else? Explain.

c. What is the efficiency class of your algorithm? Justify your answer.

The End