CSIS385 Algorithms

Lab #5 : Recurrence Formulas and Binary Search

To understand what recursion is, you must first understand recursion.
– Unknown

Names: ____

Learning goals:

- to be able to write a recurrence formula describing the running time of an algorithm
- to be able to solve recurrence formulas using the method of backward substitutions
- to be able to count the worst and average number of comparisons made by binary search

1. (10 Points) For each of the following, give a recurrence relation that describes its running time. For array computations, express your answer in terms of n, the number of array items between first and last inclusive. You do not need to get a closed form.

```
compute1( A[ first … last ] )

    if ( first = last ) return

    compute( A[ first … last - 1] )

    A[last] = A[last]*2
```

Recurrence Relation

T(1) =

T(n) =

```
compute2( A[ first … last ] )

    if ( first = last ) return

    compute( A[ first … last - 1] )

    for k ← first to last
        A[k] = A[k]* 2
```

Recurrence Relation

T(1) =

T(n) =

```
compute3( A[ first … last ] )

    if ( first = last ) return

    compute( A[ first … last - 1] )

    for k ← first to last
        for j ← first to last
            A[k] = A[k] + A[j] * 2
```

Recurrence Relation

T(1) =

T(n) =

```
compute4( A[ first … last ] )

    if ( first = last ) return

    compute( A[ first … last - 1] )
    compute( A[ first + 1 … last] )

    A[last] = A[last]*2
```

Recurrence Relation

T(1) =

T(n) =

```
cubes( n )
  if ( n = 1 ) return 1
```

Recurrence Relation:

T(1) =

T(n) =

```
else return cubes( n – 1) + n*n*n
```

2. (6 Points) For each of the following methods that print "Hello!", write a recurrence relation for the EXACT number of times it prints "Hello!". You may assume that n is a power of 2 and thus is always evenly divisible by 2.

```
Greetings( int n )
    if ( n = 1 )
        print Hello!
        print Hello!
    else
        for i ← 1 to n
            print Hello!

        Greetings( n/2 )
```

Recurrence Relation:

T(1) =

T(n) =

```
Greetings( int n )
    if ( n = 1 )
        print Hello!
        print Hello!
    else
        Greetings( n/2 )

        for i ← 1 to n
            print Hello!

        Greetings( n/2 )
```

Recurrence Relation:

T(1) =

T(n) =

```
Greetings( int n )
    if ( n = 1 ) return

    else
        Greetings( n/2 )

        print Hello!

        Greetings( n/2 )
```

Recurrence Relation:

T(1) =

T(n) =

3. Consider the recurrence formula below.

$T(1) = 3$

$T(n) = T(n-1) + 3n$

a. (5 Points) Compute the value of T(4): ____________________

b. (10 Points) Get a closed form for the recurrence using backwards substitution. Show your work for each of the three steps.

Step 1: Perform repeated substitutions and find pattern

Step 2: Determine the value of i that results in the base case

Step 3: Write the pattern for the last substitution step and simplify to get a nice looking formula.

c. (3 Points) Check your work by substituting n=8 into your formula and seeing if the results match what you got in question 3a.

4. Study the pseudocode below.

a. (3 Points) Trace the algorithm and determine how many times it prints “hello” when it is invoked with input 8? ________________

```
greetings( n )
    if ( n = 1 )
        print “hello”
        print “hello”
    else
        greetings( n/2 )
        greetings( n/2 )
        print “hello”
```

b. (5 Points) Write a recurrence formula describing exactly how many times the algorithm prints “hello.” Assume n is a power of two.

T(1) =

T(n) =

c. (10 Points) Get a closed form for your recurrence using backwards substitution. Show your work for each of the three steps.

Step 1: Perform repeated substitutions and find pattern

Step 2: Determine the value of i that results in the base case

Step 3: Write the pattern for the last substitution step and simplify to get a nice looking formula.

d. (3 Points) Check your work by substituting n=8 into your formula and seeing if the results match what you got in question 1a.

5. Consider the recurrence formula below.

$T(1) = 1$

$T(n) = 2\,T(\,n/2\,) + n$

a. (5 Points) Compute the value of T(8): _____________________

b. (10 Points) Get a closed form for the recurrence using backwards substitution. Show your work for each of the three steps. Assume n is a power of 2.

Step 1: Perform repeated substitutions and find pattern

Step 2: Determine the value of i that results in the base case

Step 3: Write the pattern for the last substitution step and simplify to get a nice looking formula.

c. (3 Points) Check your work by substituting n=8 into your formula and seeing if the results match what you got in question 2a.

6. (5 Points) Below is pseudocode for an iterative version of binary search found in your text book. Binary search is an example of a decrease and conquer algorithm.

```
ALGORITHM IterBinarySearch(A[0…n-1], X)  // Iterative version

    // Input: An array A[0…n-1] sorted in ascending order and a search key X

    // Output: index of the element that is equal to X, or -1 if there is no such element

    left ← 0

    rite ← n-1

    while left <= rite do

            mid ← ⌊(left + rite)/2⌋

            if X = A[mid] then return mid

            else if X < A[mid] then rite ← mid - 1

            else left ← mid + 1

    return -1
```

For the array below, trace the execution of IterBinarySearch(A[0…12], 39). Show the values of mid, left and rite at the bottom of each iteration of the while loop using the chart below.

3	14	27	31	39	42	55	70	74	81	85	93	98

	mid	left	rite
Starting values	----	0	12
End of 1st iteration			
End of 2nd iteration			
End of 3rd iteration			

7. Consider the array of values below.

3	14	27	31	39	42	55	70	74	81	85	93	98

a. (5 Points) What is the largest number of times the comparison X = A[mid] is performed by iterative binary search in searching for a key in the array above?

b. (5 Points) List all the keys of this array that will require the largest number of X = A[mid] comparisons when searched for by iterative binary search.

c. (6 Points) Find the average number of X = A[mid] comparisons made by iterative binary search in a successful search of this array. Assume that each key is searched for with the same probability.

d. (6 Points) Find the average number of X = A[mid] comparisons made by iterative binary search in an unsuccessful search in this array. Assume that searches for keys in each of the 14 intervals formed by the array's elements are equally likely.