SIENAcollege
Computer Science

Computer Science 385
Design and Analysis of Algorithms
Siena College
Spring 2018

Lab 6: Working With Recurrences

Due: Start of your next lab session

You will be assigned a partner to work with on this lab. Only one submission per group is needed.

Group members: _____

Learning goals:

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

---

Submitting

Once all written items are initialed to indicate completion, turn in one copy of this handout. Be sure names of all group members are clearly on the first page.

Grading

Score: [ ] / 100

Question 1:

For each of the following, give a recurrence relation that describes the number of multiplication operations it performs. Express your answer in terms of n, which is the number of array items between $first$ and $last$, inclusive. You do not need to get a closed form. (8 points, 2 each)

ALGORITHM COMPUTE1(A)
//Input: an array $A[first..last]$
if $first = last$ **then**
return
$Compute1(A[first..(last-1)])$
$A[last] \leftarrow A[last] * 2$

Recurrence Relation

$M(1) =$

$M(n) =$

ALGORITHM COMPUTE2(A)
//Input: an array $A[first..last]$
if $first = last$ **then**
return
$Compute2(A[first..(last-1)])$
for $k \leftarrow first..last$ **do**
$A[k] \leftarrow A[k] * 2$

Recurrence Relation

$M(1) =$

$M(n) =$

ALGORITHM COMPUTE3(A)
//Input: an array $A[first..last]$
if $first = last$ **then**
return
$Compute3(A[first..(last-1)])$
for $k \leftarrow first..last$ **do**
for $f \leftarrow first..last$ **do**
$A[k] \leftarrow A[k] + A[j] * 2$

Recurrence Relation

$M(1) =$

$M(n) =$

ALGORITHM COMPUTE4(A)
//Input: an array $A[first..last]$
if $first = last$ **then**
return
$Compute4(A[first..(last-1)])$
$Compute4(A[(first+1)..last)])$
$A[last] \leftarrow A[last] * 2$

Recurrence Relation

$M(1) =$

$M(n) =$

Question 2:

Give a recurrence relation that describes the number of multiplication operations performed by the algorithm below for a given value of n. You do not need to get a closed form. (2 points)

ALGORITHM CUBES(n)
  if $n = 1$ **then**
    return 1
  else
    return $Cubes(n-1) + n * n * n$

Recurrence Relation

$M(1) =$

$M(n) =$

Question 3:

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

ALGORITHM F1(n)
  if $n = 1$ **then**
    print Recursion!
    print Recursion!
  else
    for $i \leftarrow 1..n$ **do**
      print Recursion!
    $F1(n/2)$

Recurrence Relation

$R(1) =$

$R(n) =$

ALGORITHM F2(n)
  if $n = 1$ **then**
    print Recursion!
    print Recursion!
  else
    $F2(n/2)$
    for $i \leftarrow 1..n$ **do**
      print Recursion!
    $F2(n/2)$

Recurrence Relation

$R(1) =$

$R(n) =$

ALGORITHM F3(n)
  if $n = 1$ **then**
    return
  else
    $F3(n/2)$
    print Recursion!
    $F3(n/2)$

Recurrence Relation

$R(1) =$

$R(n) =$

For the next three questions, consider the recurrence formula below.

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

Question 4:
Compute the value of T(4) (5 points)

Question 5:
Get a closed form for the recurrence using backwards substitution. Show your work for each of the three steps. (10 points)

Step 1: Perform repeated substitutions until you find a pattern and can write a general formula in terms of i.

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.

Question 6:
Check your work by substituting $n = 4$ into your formula and seeing if the results match your answer when you computed $T(4)$ previously. (3 points)

For the next four questions, consider the algorithm below.

ALGORITHM GREETINGS(n)
  if $n = 1$ **then**
    print 385
    print 385
  else
    $Greetings(n/2)$
    $Greetings(n/2)$
    print 385

Question 7:
Trace the algorithm and determine how many times it prints "385" when it is invoked with input 8. (3 points)

Question 8:
Write a recurrence formula describing exactly how many times the algorithm prints "385". Assume n is a power of two. (5 points)

$T(1) =$

$T(n) =$

Question 9:
Get a closed form for your recurrence using backwards substitution. Show your work for each of the three steps. (10 points)

Step 1: Perform repeated substitutions until you find a pattern and can write a general formula in terms of i.

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.

Question 10:
Check your work by substituting $n = 8$ into your formula and seeing if the results match what you got when you traced that result above. (3 points)

For the next three questions, consider the recurrence below.

$$T(1) = 1$$
$$T(n) = 2T(n/2) + n$$

Question 11:
Compute $T(8)$. (5 points)

Question 12:
Get a closed form for this recurrence using backwards substitution. Show your work for each of the three steps. Assume n is a power of 2. (10 points)

Step 1: Perform repeated substitutions until you find a pattern and can write a general formula in terms of i.

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.

Question 13:
Check your work by substituting $n = 8$ into your formula and seeing if the results match what you got for $T(8)$ above. (3 points)

For the remaining questions, consider the iterative binary search psuedocode below. Binary search is an example of a decrease and conquer algorithm.

ALGORITHM BINARYSEARCH(A, K)
//Input: a sorted array $A[0..n-1]$
//Input: search key K
$l \leftarrow 0$
$r \leftarrow n-1$
while $l \leq r$ **do**
  $m \leftarrow \lfloor \frac{l+r}{2} \rfloor$
  if $K = A[m]$ **then**
    return m
  else if $K < A[m]$ **then**
    $r \leftarrow m-1$
  else
    $l \leftarrow m+1$
return -1

Question 14:
For the array below, trace the execution of `BinarySearch(A[0..12], 39)`. Show the values of m, l and r at the bottom of each iteration of the while loop, using the chart below. (5 points)

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

	m	l	r
Starting values	—	0	12
End of 1^{st} iteration			
End of 2^{nd} iteration			
End of 3^{rd} iteration			

Here is the array again from the previous page:

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

Question 15:

What is the largest number of times the comparison $K = A[m]$ is performed by iterative binary search in searching for any key in the array above? (5 points)

Question 16:

List all the keys of this array that will require the largest number of $K = A[m]$ comparisons when searched for by iterative binary search. (5 points)

Here is the array again:

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

Question 17:
Find the average number of $K = A[m]$ comparisons made by iterative binary search in an successful search in this array. Assume that a search for each key is equally likely. (6 points)

Question 18:
Find the average number of $K = A[m]$ 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. (6 points)