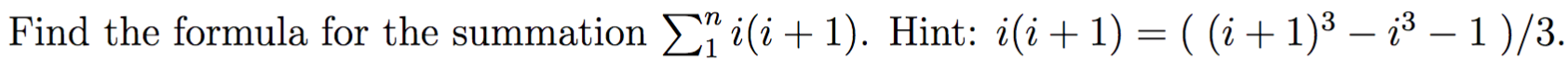
**CS3340 Assignment 1**

Question 1



= + + + + … +

= + + + + … +

=

=

=

=

=

Since then

=

=

=

=

=

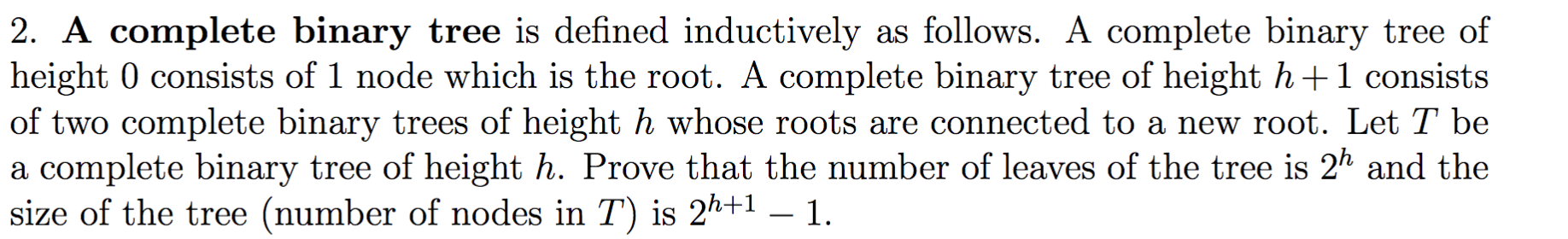
=

=

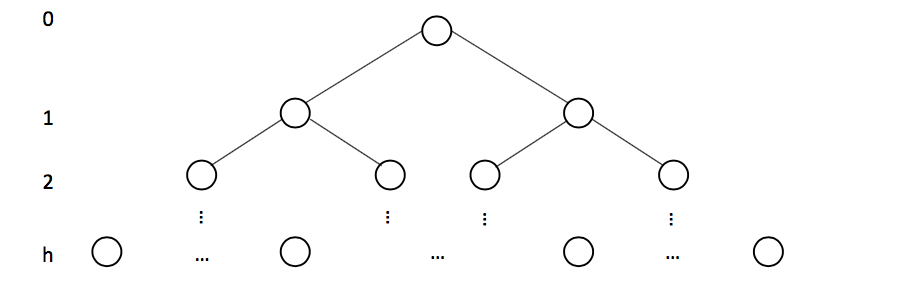
=

Therefore, =

Question 2



Height



|  |  |  |
| --- | --- | --- |
| Height | #nodes | #leaves |
| 0 | 1 | 1 |
| 1 | 3 | 2 |
| 2 | 7 | 4 |
| 3 | 15 | 8 |
| 4 | 31 | 16 |
| … | … | … |
| h | 2h+1-1 | 2h |

Proof for # nodes:

Induction base: n(0) = 1

n(0) = 20+1 - 1

n(0) = 2 - 1 = 1

Induction Hypothesis: Assume n(T) = n(T1) + n(T2) + 1

2h+1 – 1 = (2(h-1)+1 – 1) + (2(h-1)+1 – 1) + 1

2h+1 – 1 = (2h – 1) + (2h – 1) + 1

2h+1 – 1 = 2h – 1 + 2h – 1 + 1

2h+1 – 1 = 2h + 2h – 1

2h+1 – 1 = 2(2h) – 1

2h+1 – 1 = 2h+1 – 1

Therefore, the size of the tree is 2h+1 – 1

Proof for # leafs:

Induction Base: l(0) = 20 = 1

Induction Hypothesis: l(T) = l(T1) + l(T2)

2h = 2(h-1) + 2(h-1)

2h = (2-1) 2h + (2-1)2h

2h = 2h/2 + 2h/2

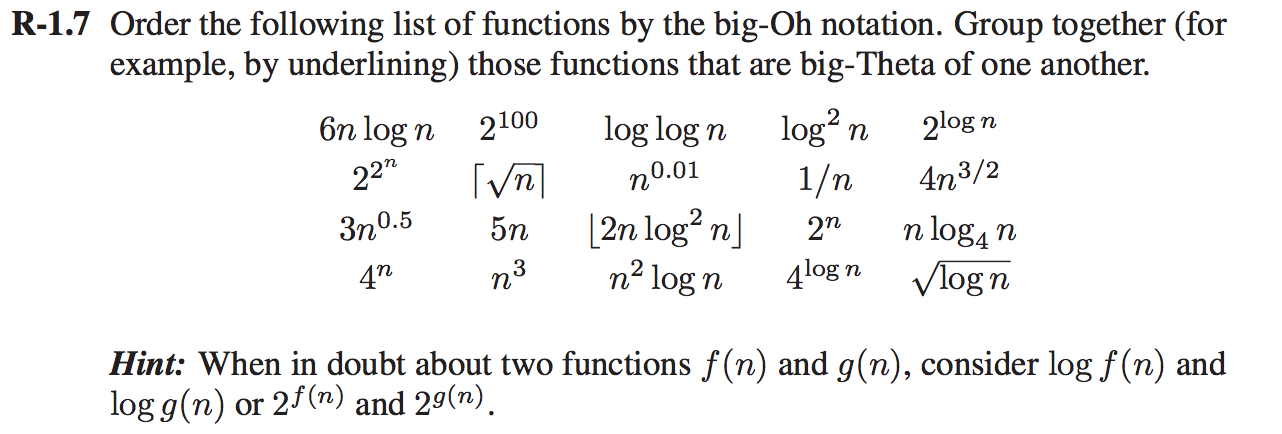
2h = (2) 2h/2

2h = 2h

Therefore, the number of the leafs in the tree is 2h

Question 3

https://i.gyazo.com/1719ed9184ea5333a1ac9477ab883240.png

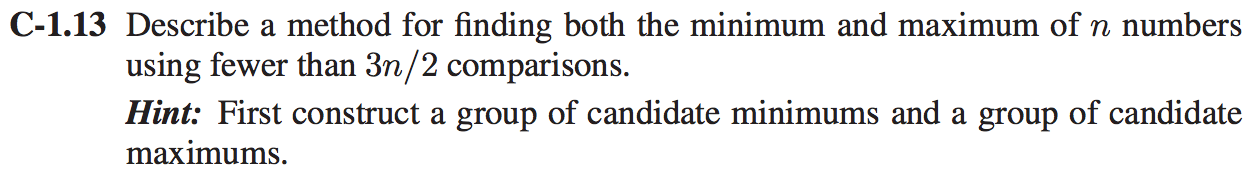


For those that are grouped together by big-theta, they will appear on the same line separated by space

1. 1/*n* Fastest
2. 2100
3. log log *n*
4. *√* log *n*
5. log2 *n*
6. *n*0.01
7. ⎡*√n*⎤, 3*n*0.5
8. 2log *n*, 5*n*
9. *n* log4 *n*, 6*n* log *n*
10. ⎣2*n* log2 *n*⎦
11. 4*n*3/2
12. 4log *n*
13. *n*2 log *n*
14. *n*3
15. 2*n*
16. 4*n*
17. 22*n* Slowest

Question 4

https://i.gyazo.com/e7a7aba601307cb3ffd0259b537ecf74.png



Start by comparing each consecutive pair of numbers, collecting the lesser number into one group and the greater number in another group.

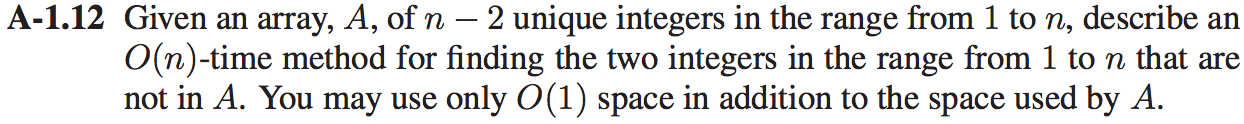
This step takes n/2 comparisons.

Next, use n/2 comparisons to find the smallest number from the group of lesser numbers, and n/2 comparisons to find the biggest number from the group of greater numbers.

Total comparisons would be 3n/2 altogether.

Question 5

https://i.gyazo.com/b6a673ab6ae99fa3f4c09c3d3a63fdb0.png



We start by creating 6 variables:

* sum (The sum of integers from 1 to n)
* sumT (The sum of elements from the array A)
* product (The product of integers from 1 to n)
* productT (The product of the elements from the array A)
* differenceS (The difference between sum and sumT)
* differenceP (The difference between product and productT)

Next, we will loop through the integers in the range from 1 to n.

* Add each item to sum.
  + sum += item
* Multiply each item to product
  + product \*= item

Once we’re done the first loop we will loop through the elements in array A

* Add each element to sumT
  + sumT += A[i]
* Multiply each element to productT
  + productT \*= A[i]

Now that both loops are done, we calculate differenceS and differenceP

* differenceS is the difference between sum and sumT
  + differenceS = sum – sumT
* differenceP is the difference between product and productT
  + differenceP = difference – differenceT

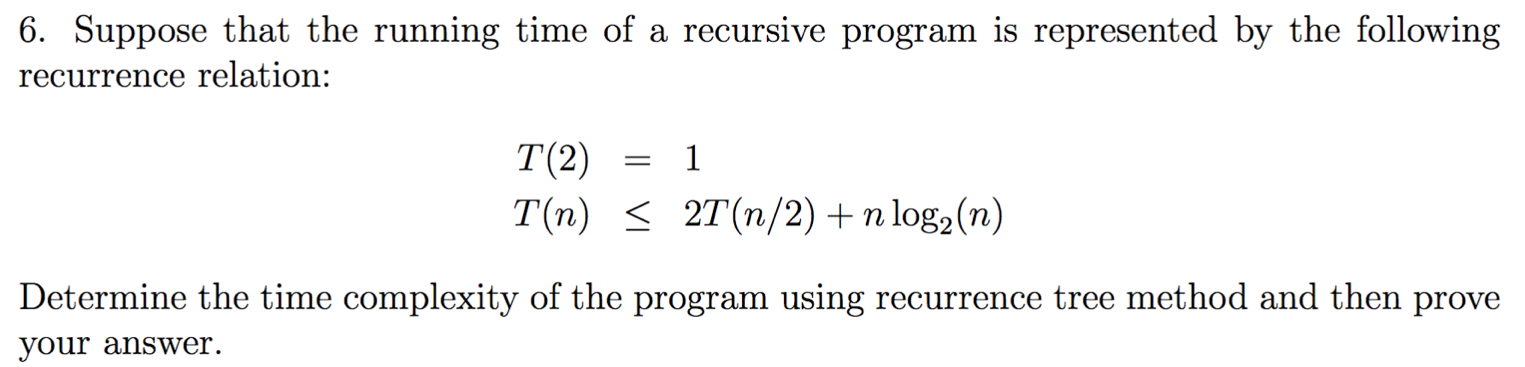
Because we have the differences, we now loop through array A going 1 pair at a time.

* For each pair(x,y) in the set of integers from 1 to n, we compare x+y with differenceS and x\*y with differenceP
* If both x+y and x\*y are equal to differenceS and differenceP then check the array A if x and y are elements of the array.
* If they are, then go back to comparing integers. Otherwise return the pair(x,y) as they are the numbers not in A.

Time complexity: O(n) + O(n-2) + O(n) + O(n-2) = O(n)

Only O(1) space is used, in addition to the space used by A.

Question 6



T(n) n log n

/ \

T(n/2) T(n/2) n log n

k / \ / \

T(n/4) T(n/4) T(n/4) T(n/4) n log n

/ \ / \ / \ / \

…

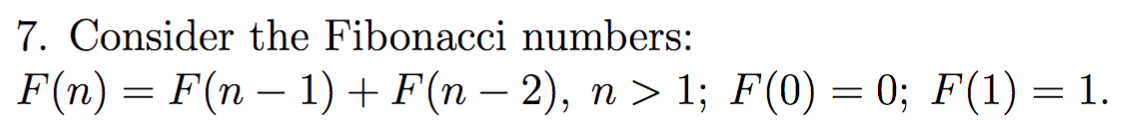
| | | | | | | |

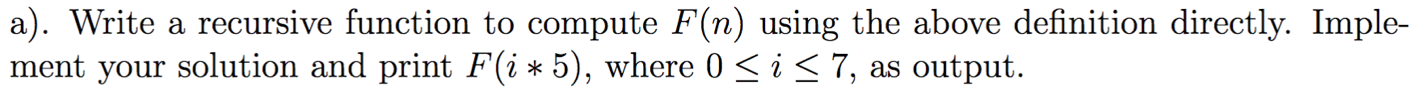
1 1 1 1 1 1 1 1 n

Total: n log n + n

In total we get n log n by summing the amount of work done at each level

Question 7



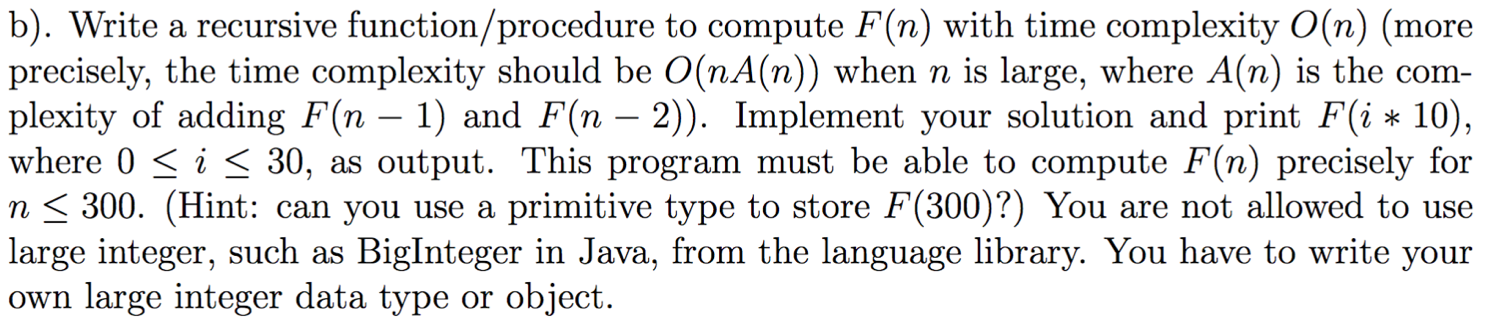


To run the program, type the following into the command prompt

“./asn1\_a.sh” which will compile and run the program

Make sure the .java file is in the same directory as the shell script

Time complexity is O(2n)



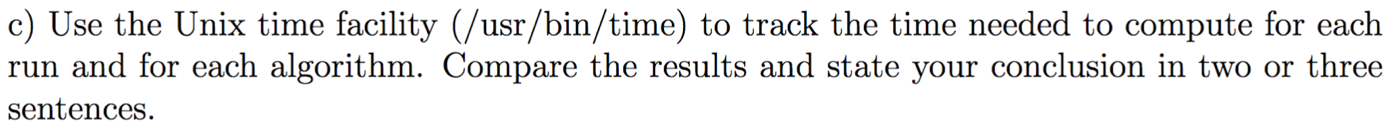
To run the program, type the following into the command prompt

“./asn1\_b.sh” which will compile and run the program

Make sure the .java file is in the same directory as the shell script

Time complexity is O(nA(n)) where A(n) is the complexity to add f(n-1) and f(n-2).

A(n) is O(1), the O(n) comes from the loop that adds f(n-1) and f(n-2) as the loop iterates n times. So the time complexity is O(n(1)) which is O(n).



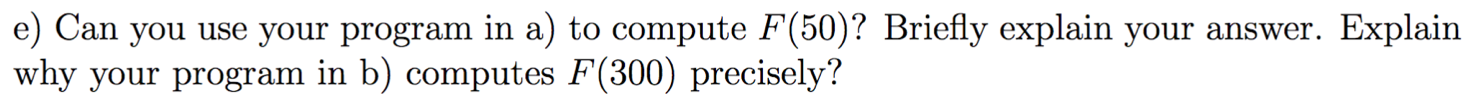
The Fibonacci recursive algorithm in “a”, is expensive and requires a larger amount of time to calculate when compared to the algorithm used for “b”. The algorithm used for “b”, has a time complexity of O(n), compared to the algorithm in “a” which used time complexity of O(2n).

The difference between the time complexities are given through the test results.

a took 0.8 s

b took 0.5 s

d) \*\*The assignment doesn’t have 7d but has 7e I’m assuming it’s a typo\*\*



Program “a” cannot be used to compute F(50), as the 50th term of the Fibonacci sequence is larger than 232, which is the max that an integer object can store.

Program “b” is able to compute F(300), as the numbers are stored and calculated using an array of integers. Each digit in the number are stored in an index of the array. When the arrays containing n-1 and n-2 are added together, each index of the arrays are added together, and stored into a separate array. At the end of the calculation, the array that contained F(300) looked like the following:

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 3, 2, 2, 4, 4, 6, 2, 9, 4, 2, 0, 4, 4, 5, 5, 2, 9, 7, 3, 9, 8, 9, 3, 4, 6, 1, 9, 0, 9, 9, 6, 7, 2, 0, 6, 6, 6, 6, 9, 3, 9, 0, 9, 6, 4, 9, 9, 7, 6, 4, 9, 9, 0, 9, 7, 9, 6, 0, 0

When printed, the leading zeroes were cut, leaving the following elements to be printed:

2, 2, 2, 2, 3, 2, 2, 4, 4, 6, 2, 9, 4, 2, 0, 4, 4, 5, 5, 2, 9, 7, 3, 9, 8, 9, 3, 4, 6, 1, 9, 0, 9, 9, 6, 7, 2, 0, 6, 6, 6, 6, 9, 3, 9, 0, 9, 6, 4, 9, 9, 7, 6, 4, 9, 9, 0, 9, 7, 9, 6, 0, 0

When printed out without any commas, spaces, or new lines gives the following number:

222232244629420445529739893461909967206666939096499764990979600

This number is equal to F(300) which is calculated properly.