CSCE 222 (Carlisle), Honors Discrete Structures for Computing
Fall 2020
Homework 6

Type your name below the pledge to sign
On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work.
Sameer Hussain

Instructions:

- The exercises are from the textbook. You are encouraged to work extra problems to aid in your learning; remember, the solutions to the odd-numbered problems are in the back of the book.
- Grading will be based on correctness, clarity, and whether your solution is of the appropriate length.
- Always justify your answers.
- Don't forget to acknowledge all sources of assistance in the section below, and write up your solutions on your own.
- *Turn in .pdf file to Gradescope by the start of class on Tuesday, October 6, 2020.* It is simpler to put each problem on its own page using the LaTeX clearpage command.

Help Received:

- https://www.ntg.nl/doc/biemesderfer/ltxcrib.pdf
- https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/

Exercises for Section 3.3:

2: (1 point).
Addition: t = t + i + j
From i to n and from j to n nested loop
$n * n * 2 = 2n^2$
$= O(n^2)$

4: (1 point).
The value of i is double for every repetition of the while loop
$i = 2^n$
$= log(2n)$
$= O(log(n))$

8: (2 points).
Yes, there is a more efficient way. When you multiply it out, you need 2^k multiplications to determine x^{2^k}
It would be more efficient to find it by just squaring since $k \leq 2^k$ for all k.

10b: (2 points).
The bitwise AND operations come from: $S := S \wedge (S - 1)$
This happens on every iteration of the while loop.
It is equal to the number of 1's in the string.

12b: (2 points).
From part a, the algorithm uses $\theta(n^3) comparisons$
Outer loop executed $\frac{n}{4}$ times and and middle loop $\frac{n}{4} times$ once from $\frac{3n}{4}$ to n
$\frac{3n}{4} - \frac{n}{4} = \frac{n}{2}$
$(\frac{n}{4})(\frac{n}{4})(\frac{n}{2}) = \frac{n^3}{32}$
Thus, the complexity is $\Omega(n^3)$ times

14a: (2 points)
First y is assigned as:
$y = a_n = a_2 = 3$
Then on first iteration $i = 1$
$y = y_0 * c + a_{n-i} = 3 * 2 + a_{2-1}$
$= 7$
On the second iteration: $i = 2$
$y = y_0 * c + a_{n-i} = 7 * 2 + a_{2-2}$
$= 15$

14b: (2 points)
Multiplications $= n$
Additions $= n$

16(a-f): (4 points) One day $= 8.64x10^{15}$ possible bit operations in 86400 seconds.

a)
$n = 2^{log2n} = 2^{8.64*10^{15}}$

b)
$1000n = 8.64 * 10^{15}$
$n = 8.64 * 10^{12}$

c)
$n = \sqrt{8.64 * 10^{15}} = 9.295 * 10^7$

d)
$n = \sqrt{8.64 * 10^{12}} = 2.939 * 10^6$

e)
$n = \sqrt[3]{8.64 * 10^15}$

f)
$n = log_2(8.64 * 10^15)$

20(b,c,e,g): (2 points)

b)

$f(2n) - f(n) = log(2n) - logn$

$= log(2) + (logn - logn) = log(2)$

c)

$\frac{f(2n)}{f(n)} = \frac{100(2n)}{100n} = 2$

e)

$\frac{f(2n)}{f(n)} = \frac{(2n)^2}{n^2} = 4$

g)

$\frac{f(2n)}{f(n)} = \frac{2^{2n}}{2^n} = 2^n$

42: (2 points)

Need to find complexity of greedy algorithm that schedules the most talks adding with earliest time compatible.

Goes from i = 1 to n

if i compatible with S then

S := S and the talk i

return S

Complexity is: $O(nlogn)$