University of Nevada, Las Vegas Computer Science 477/677 Fall 2022

Assignment 1: Answers

1. True or False

   (a) **F** $n^2 = \Theta(n^3)$.

   (b) **T** $\log n^2 = \Theta(\log n^3)$.

   (c) **F** $\log^2 n = \Theta(\log^3 n)$.

   (d) **F** $n^{1.0001} = O(100 \log^{100} n)$

2. This problem is based on problem 0.3(c) on page 9 of the textbook. I have rewritten the problem, and the answer differs from the answer to the problem in the textbook.

   The Fibonacci numbers $F_1, F_2, \ldots$ are defined by the rules
   $F_1 = 1$
   $F_2 = 1$
   $F_n = F_{n-2} + F_{n-1}$ for $n \geq 3$

   (a) Compute F_n for $n = 3, 4, 5, 6, 7, 8$.

   (b) Find a constant C such that $F_n = \Theta(C^n)$?

   Solution: Assume that $F_n = C^n$. This is false, but it's close enough. Then

$$
\begin{aligned}
C^n &= C^{n-2} + C^{n-1} \\
\text{Divide by } C^{n-2}: & \\
C^2 &= C + 1 \\
C^2 - C - 1 &= 0 \\
\text{By the quadratic formula:} & \\
C &= \frac{1 \pm \sqrt{5}}{2} \\
\text{Since } C \text{ must be positive:} & \\
C &= \frac{1 + \sqrt{5}}{2} \approx 1.6180339
\end{aligned}
$$

3. The following code fragment takes $\Theta(n^2)$ time to execute. (We assume that n is given.)

```
for(int i = 0; i < n; i++)
  for(int j = 0; j < n; j++)
```

Here is a pseudocode version. It is humanly readable, but not in any particular programming language.

```
Read n
initialize count
increment count
i = 0
while i < n
   increment i
   increment count
   j = 0
   while j < n
      increment j
      increment count
Print count
```

(a) Write this program in C++.

```
int n;
int main()
 {
  int kounter = 0;
  cout << "Enter a value of n: ";
  cin >> n;
  cout << " n = " << n << endl;
  kounter++;
  int i = 0;
  while(i < n)
   {
    i++;
    kounter++;
    int j = 0;
    while(j < n)
     {
      j++;
      kounter++;
     }
   }
  cout << "counter = " << kounter << endl;
  return 1;
 }
```

(b) Run this program, entering various values of n, such as 1, 4, 10, 30, 100.

Enter a value of n: 1 n = 1 counter = 3

Enter a value of n: 4 n = 4 counter = 21

Enter a value of n: 10 n = 10 counter = 111

Enter a value of n: 30 n = 30 counter = 931

Enter a value of n: 100 n = 100 counter = 10101

(c) Looking at the output, is it clear to you that the time complexity of the program is $\Theta(n^2)$?

Programming tip: never use an English word as an identifier in a program. Use a misspelled word, or a foreign word, instead. Why?

4. For each of these fragments:
   Expand the fragment to a C++ program with a counter.
   Run the code for $n = 1, 10, 100, 1000, 1000000$.
   For each, compute the final value of the counter for each n. Show the results in a table. (You do not have to turn in your code.)

   Guess the asymptotic time complexity, which will be $\Theta(1)$, $\Theta(\log n)$, $\Theta(n)$, $\Theta(n \log n)$, $\Theta(\sqrt{n})$, or $\Theta(n^2)$.

	1	10	100	1000	1000000	complexity
(a)	1	4	7	10	20	$\Theta(\log n)$
(b)	1	3	24	2984	2999973	$\Theta(n)$
(c)	2	4	11	33	1001	$\Theta(\sqrt{n})$
(d)	1	35	673	9977	19951425	$\Theta(n \log n)$

(a)
```
for(int i = n; i > 0; i=i/2)
    i = i/2;
```

(b)
```
for(int i = 1; i < n; i++)
    for(int j = n; j > i; j=j/2);
```

(c)
```
for(int i = 0; i*i < n; i++)
```

(d)
```
for(int i = 1; i < n; i++)
    for(int j = i; j > 0; j=j/2);
```

5. The recursive algorithm implemented below as the C++ function `mystery` computes a well-known algebraic operation. What is that operation?

```
1  float squre(float x)
2   {
3    return x*x;
4   }
5
6  float mystery(float x, int k)
7   {
8    if (k == 0) return 1.0;
9    else if(x == 0.0) return 0.0;
10   else if (k < 0) return 1/mystery(x,-k);
11   else if (k%2) return x*mystery(x,k-1);
12   else return mystery(squre(x),k/2);
13  }
```

It returns x^k.

It is clear that `squre(x)` returns x^2.

The code fails if $x = 0$ and $k \leq 0$, because division by zero is impossible and because 0^0 is undefined. So we can only prove correctness if $x \neq 0$ or $k > 0$.

Proof of Correctness: By induction on the depth of the recursion.

If $k = 0$, then the execution at line 8 returns 1, the correct value. If $x = 0$, then the execution at line 9 return 0, the correct value. Recursion will occur if the first line executed in 10, 11, or 12. If $k < 0$, then $x^k = \dfrac{1}{x^{-k}}$, and the correct value is returned at line 10. Suppose that $k > 0$ and $x \neq 0$. If k is even, the line 12 is executed. Since $x^k = (x^2)^{\frac{k}{2}}$, the correct value will be returned. If k is odd, then $k - 1$ is even. Line 11 returns $x * (x^{k-1}) = x^k$, which is correct.