CS-UY 2413: Design&Analysis of Algorithms—Spring 2017 Prof. Boris Aronov
Department of Computer Science and Engineering Prof. Lisa Hellerstein

Homework 1

Due Jan. 31 at the start of lecture (Hellerstein's sections).
Due Feb. 1 at the start of lecture (Aronov's section).
This homework should be handed in on paper, not electronically. No late homeworks accepted. Contact your professor for special circumstances.

Policy on collaboration on this homework: Feel free to discuss this homework and write the answers in groups. Each group must have no more than THREE (3) people in it; all from the same section. Hand in ONLY ONE homework per group. On that homework, put the names, netIDs, and sections of ALL the people in the group. Your name should appear on only one homework that is handed in. Homeworks handed in separately with identical or similar answers will be considered academic dishonesty.

Any homework submitted is expected to be the work of the students whose names are on it. If there is evidence that the work is not your own (such as copying from others, from the Internet, paying a third party to carry out the work, etc.), it will be treated as academic dishonesty and will be reported to the department and the Dean of Student Affairs. You will receive a zero on the homework, and if it happens again, you will receive an F in the course. It does not matter who copied from whom.

Format: Please type or write very clearly. Mathematical formulas, if not clearly handwritten, must be properly typeset (no n2 instead of n^2 etc). Indecipherable answers will be returned ungraded.

Useful Fact #1: It takes approximately $\log_b N$ b-ary digits to represent the number N in base b. More precisely, it takes $\lceil \log_b(N+1) \rceil$.

Useful Fact #2: $1 + 2 + 3 + \ldots + N = \frac{N(N+1)}{2}$

Useful Fact #3: For $r \neq 1$, $r^0 + r^1 + \ldots + r^n = \frac{r^{n+1}-1}{r-1}$

Review: You are encouraged to read "Bases and Logs," on page 12 of the textbook, especially if you don't know these topics well.

---

1. Answer the following:

   (a) What is the largest number that can be represented in binary using only 4 bits?
   Solution: The largest number that can be represented is $2^4 - 1 = 15$.

(b) As a function of n, what is the largest number that can be represented in binary using only n bits? (Make sure your answer is consistent with the answer you gave for the previous question.)

Solution: The largest number that can be represented in binary using only n bits is $2^n - 1$.

2. According to Forbes magazine, Bill Gates's net worth on Jan. 24, 2017 was \$84.4 Billion (that is, \$84,400,000,000).

   (a) If you start with a dollar and double your money every day after that, it will take you 3 days until you have \$8.

   If you start with a dollar, and you double your money every day after that, how many days will it take for you to have at least \$84.4 Billion? Your answer should be a number of days. You may use a calculator or program to calculate your answer. You should try to use logs in your calculations.

   Solution: If we double our money every day, then the amount of money we have after n days is 2^n. If we want to find how many days it will take to reach a certain value, then, we should use the base-2 logarithm. Applying this to \$84.4 Billion, we have $\log_2 84,400,000,000 \approx 36.30$. So, after $\lceil \log_2 84,400,000,000 \rceil = 37$ days, you will have at least \$84.4 Billion.

   (b) How many bits are in the binary representation of the number \$84,400,000,000?

   Solution: Using Useful Fact #1, we have that it takes $\lceil \log_2(84,400,000,001) \rceil = 37$ bits to represent the number 84,400,000,000 in binary.

3. Let's approximate the number of bits required to write down the 200th Fibonacci number, F_{200}, in binary.

   Recall that F_N is approximately equal to $2^{0.694N}$. Therefore, the number of bits required to write down the 200th Fibonacci number is approximately $\log_2 2^{0.694N}$, for $N = 200$. What is this value?

   You should be able to answer this question without a calculator.

   Solution: Recalling that $\log_2 2^x = x$, we have that $\log_2 2^{0.694N} = 0.694N$. For $N = 200$, we get 138.8.

4. Suppose you have a long straight path, forming a rectangle with dimensions $1 \times k$ (measured in meters) for some integer $k > 0$. You would like to cover that path with tiles. (Note that the tiles cannot overlap, and they must cover the entire path.) You have two types of tiles you can use: tiles of dimension 1×1 and tiles of dimension 2×1.

   For $k = 1$, there is only 1 way to tile the path. You must use a single tile that is 1×1.

   For $k = 2$, there are 2 ways to tile the path. You can either use two 1×1 tiles, or one 2×1 tile.

If $k > 2$, how many ways can you tile the path? Express your answer as a Fibonacci number, with a subscript that depends on k, and explain why your answer is correct.

Solution: When we wish to lay down tiles to form a $k \times 1$ path, we have two choices:

- Create a path of length $k-1$, and then lay down a 1×1 tile
- Create a path of length $k-2$, and then lay down a single 2×1 tile

Thus the total number of ways to tile a path of length k is equal to the sum of the number of ways to tile a path of length $k-1$ and the number of ways to tile a path of length $k-2$.

We can express this as a recurrence relation. If we let $T(k)$ represent the number of ways to tile a path of length k, then we have:

$$T(k) = \begin{cases} T(k-1) + T(k-2) & \text{if } k > 2 \\ 2 & \text{if } k = 2 \\ 1 & \text{if } k = 1 \end{cases}$$

This is just like the definition for the Fibonacci numbers, except offset by one. That is, $T(2) = 2$, but 2 is the *third* Fibonacci number. In general, $T(k) = F_{k+1}$.

5. For each example, indicate whether $f = O(g)$ or $f = \Omega(g)$ or $f = \Theta(g)$. No justification is necessary.

   (a) $f = 2^n$, $g = n^2$
   Solution: $f = \Omega(g)$

   (b) $f = 2^{\log_2 n}$, $g = 3n$
   Solution: $f = \Theta(g)$

   (c) $f = 3n^{1/3}$, $g = \frac{1}{3}n$
   Solution: $f = O(g)$

   (d) $f = 2n^2$, $g = n \log_2 n$
   Solution: $f = \Omega(g)$

   (e) $f = 5n^4 + 4n^3 + 3n^2 + 2n + 1$, $g = 15n^3$
   Solution: $f = \Omega(g)$

   (f) $f = 4 \log_2 n$, $g = \log_4^2 n$
   Solution: $f = O(g)$

   (g) $f = \log_{10} n$, $g = \log_5 n + \log_2 n$
   Solution: $f = \Theta(g)$

   (h) $f = 3^n$, $g = n!$
   Solution: $f = O(g)$

(i) $f = 2^{(\log_2 n)}$, $g = n^2$
Solution: $f = O(g)$

(j) $f = 16^{(\log_4 n)}$, $g = n^2$
Solution: $f = \Theta(g)$

6. For each program below, write down (a) the exact value of the variable `result` at the end of execution, in summation notation, (b) how this value grows with `N`, in the Θ-notation, in simplest form. Justify your answer. Assume that `result` is initialized to `0` and `N` has positive value at the start of execution. For example, for the code

```
for i = 0 ... N+1
   result=result + 2
```

the correct answers would be $\sum_{i=0}^{N+1} 2$ and $\Theta(N)$, respectively. Notes: (i) Writing $\Theta(2N)$ would not be counted as a correct answer, as it can be simplified. (ii) The question is about the *value* of a variable, not about the running time. (iii) "=" denotes assignment. (iv) Your asymptotic estimate has to be a Θ, not just O or just Ω. There are in general two ways to achieve this: you can compute the exact expression, such as $n(n+4)/5+3n$ and then observe that it is $\Theta(n^2)$. Alternatively, you estimate an upper and a lower bound separately, if you get both $O(n^2)$ and $\Omega(n^2)$, you can conclude that the answer is $\Theta(n^2)$.

(a)
```
for i = 1 ... N
    for j = 1 ... i
          result = result + 2
```
Solution: First, we get that `result` $= \sum_{i=1}^{N} \sum_{j=1}^{i} 2$. This is equivalent to $2\sum_{i=1}^{N} \sum_{j=1}^{i} 1$. Next, $\sum_{j=1}^{i} 1 = i$, so we get `result` $= 2\sum_{i=1}^{N} i = 2(1 + 2 + 3 + \cdots + N) = N(N + 1)$, using the Useful Fact #2 for the final step. Thus, `result` $= \Theta(N^2)$.

(b)
```
j = 1
while j < √N
  result = result + j
  j = j + 1
```
Solution: We find that `result` $= 1 + 2 + \cdots + \lfloor\sqrt{N-1}\rfloor$. Again, we can apply Useful Fact #2. In this case, we get that `result` $= \frac{\lfloor\sqrt{N-1}\rfloor(\lfloor\sqrt{N-1}\rfloor+1)}{2}$. Note that the floor operation has a negligible effect: $x - 1 < \lfloor x \rfloor \leq x$. So this does not affect the asymptotic growth of `result`. Then we can say that `result` $= \frac{\lfloor\sqrt{N-1}\rfloor(\lfloor\sqrt{N-1}\rfloor+1)}{2} = \Theta(\sqrt{N} \times \sqrt{N}) = \Theta(N)$

7. If N is a power of 2, what is the value of `result` after the execution of the code below? Give your answer as a function of N. (Hint: Test that your answer is correct by checking what the code does for $N = 4$.)

```
result = 0
i = 1
while i < N
   i = 2 * i
   result = result + 1
```

Solution: `result` $= \log_2 N$. Note that when $N = 4$, `result` $= 2$ and when $N = 8$, `result` $= 3$.