cmput 304 2024 study questions 1 show your work

recall: $2^{10} = 1024$ $\lg e = 1.4...$ $\lg 10 = 3.3...$

1. (a) Explain why any steiner tree for this puzzle must include at least 4 horizontal edges.

   (b) Repeat (a) for vertical edges.

   (c) Explain why a min cost solution to this puzzle must include at least 8 edges.

   (d) Find a min cost solution to this puzzle.

   Answer:

   (a) The steiner tree for the given puzzle joins the nodes A2, B5, E1 and E3. These nodes span across columns A to E. To connect the nodes in these columns, the tree needs to pass through columns in between. Thus, the solution requires a horizontal path that connects column A to column E, moving across columns B, C, and D. Therefore, any steiner tree for this puzzle must include at least 4 horizontal edges.

   (b) Refer part (a), tree needs to vertically pass through all the rows from 1 to 5

   (c) As determined in parts (a) and (b), any steiner tree for the given puzzle includes at least 4 horizontal edges to span the columns and 4 vertical edges to span the rows. Therefore, the minimum cost solution will have at least 8 edges.

   (d) Min cost: 8

   - Connect point at (A, 3) to (E, 3) horizontally
   - Connect point at (B, 3) to (B, 5) vertically
   - Connect point at (E, 1) to (E, 3) vertically

2. A number q satisfies $\ln q = 529$. Give an arithmetic expression for $\lg q$ and estimate it roughly. (You do not have to calculate it). Repeat for $\log_{10} q$?

   Answer:

   We have $\ln q = 529$. Using properties of logarithms we have

   $$529 = \ln q = \frac{\log_2 q}{\log_2 e},$$

   and so $\log_2 q = 529 \log_2 e$. A simple estimation is given by $2 \leq e \leq 2^2$ and so we can estimate

   $$529 \leq \log_2 q \leq 1058.$$

   Sharper estimates can be obtained with sharper bounds on e. Now if $\ln q = 529$ then we can write

   $$529 = \ln q = \frac{\log_{10} q}{\log_{10} e},$$

   and so $\log_{10} q = 529 \log_{10} e$. Again another simple estimation yields $10^{1/3} \leq e \leq 10^{1/2}$ and so we may write

   $$176.333 \leq \frac{529}{3} \leq \log_{10} q \leq \frac{529}{2} = 264.5.$$

   The approximate value of $\log_2 q$ is 763.18 and the approximate value of $\log_{10} q$ is 229.74.

3. Let $x = 1,000,000$. Give an arithmetic expression for $\lg x$ and estimate it roughly. (You do not have to calculate it). Repeat for $\ln x$ and $\log_{10} x$.

   Answer:

   If $\log_2 x = a$, then $x = 2^a$. Note that $2^{20} = 1048576 \approx 10^6$, so applying logs, we obtain $\log_2 x \approx 20$. The approach is similar for $\ln x$ by writing $e \approx 3$. For $\log_{10} x$, we note that x is already a power of 10, so then $\log_{10} x = 6$.

4. Recall `http://webdocs.cs.ualberta.ca/~hayward/272/jem/collatz.html`.
   Prove or disprove (if you can):

   a) for all positive integers n, the last number printed by `collatz(n)` is 1.

   b) for all n in {1,2,...,10}, the last number printed by `collatz(n)` is 1.

   c) if the collatz conjecture fails for some integer, and if n_0 is the smallest such integer, then n_0 is odd.

   Answer:

   (a) This is an open problem! You will be famous if you solve it.

   (b) This is a simple verification of the algorithm at hand. One has the sequences:

$$1$$
$$2 \to 1$$
$$3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$4 \to 2 \to 1$$
$$5 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$6 \to 3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$7 \to 22 \to 11 \to 34 \to 17 \to 52 \to 26 \to 13 \to 40 \to 20$$
$$\to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$8 \to 4 \to 2 \to 1$$
$$9 \to 28 \to 14 \to 7 \to \ldots \text{(follow from 7)}$$
$$10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1.$$

   Hence statement b is correct.

   (c) If the Collatz conjecture fails, then this algorithm will not halt. Hence, if n_0 is the smallest such number in an infinite sequence of outputs where the conjecture fails it must be odd, for if it was even, then the next element in the sequence would be $n_0/2 < n_0$, a contradiction to the minimality of n_0.

5. Partition these functions into same-theta equivalence classes, in increasing order (so if $\lim_{n\to\infty} f(n)/g(n) = 0$ then put $f(n)$ before $g(n)$. Also, for each class, give the simplest function in that class. For example, the simplest function in the class $\Theta(3.5n^2 + 6\lg n)$ is $f(n) = n^2$.

$a(n) = 2^n$ $b(n) = 3^n$ $c(n) = 1.5^{2n}$ $d(n) = n^2 + \lg n$ $e(n) = n^3 + (\lg n)^4$

$f(n) = \ln n + 2\lg n + 16\log_{10} n$ $g(n) = n^{2.5} + (\lg n)^{90}$

$h(n) = \ln n^2 + 0.5n^2 \lg n$ $k(n) = \ln n^2 + 0.5(n \lg n)^2$

Answer:

Same-theta equivalence classes and their simplest function:

$a(n) = 2^n$:

Same-theta equivalence: $\Theta(2^n)$ Simplest function: $a(n) = 2^n$

$b(n) = 3^n$:

Same-theta equivalence: $\Theta(3^n)$ Simplest function: $b(n) = 3^n$

$c(n) = 1.5^{2n}$:

Same-theta equivalence: $\Theta(1.5^{2n})$ Simplest function: $c(n) = 2.25^n$

$d(n) = n^2 + \lg n$:

Same-theta equivalence: $\Theta(n^2)$ Simplest function: $d(n) = n^2$

$e(n) = n^3 + (\lg n)^4$:

Same-theta equivalence: $\Theta(n^3)$ Simplest function: $e(n) = n^3$

$f(n) = \ln n + 2\ \lg n + 16\log n$:

Same-theta equivalence: $\Theta(\log n)$ Simplest function: $f(n) = \log n$

$g(n) = n^{2.5} + (\lg n)^{90}$:

Same-theta equivalence: $\Theta(n^{2.5})$ Simplest function: $g(n) = n^{2.5}$

$h(n) = \ln n^2 + 0.5n^2 \lg n$:

Same-theta equivalence: $\Theta(n^2 \lg n)$ Simplest function: $h(n) = n^2 \lg n$

$k(n) = \ln n^2 + 0.5(n \lg n)^2$:

Same-theta equivalence: $\Theta((n \lg n)^2)$ Simplest function: $k(n) = (n \lg n)^2$

In their increasing order:

$\log n < n^2 < n^2 \lg n < (n \lg n)^2 < n^{2.5} < n^3 < 2^n < 1.5^{2n} < 3^n$

Bonus: Proof of $\lim_{n\to\infty} \frac{n^{2.5}}{(n \lg n)^2} = \infty$

$$\lim_{n\to\infty} \frac{\sqrt{n}}{\lg^2 n} = \lim_{n\to\infty} \frac{\frac{d(n^{0.5})}{dn}}{\frac{d(\lg^2 n)}{dn}} = \lim_{n\to\infty} \frac{\frac{1}{2n^{0.5}}}{2 \lg n \frac{1}{n}} = \lim_{n\to\infty} \frac{\sqrt{n}}{4 \lg n}$$

$$= \lim_{n\to\infty} \frac{1}{4 \times \frac{1}{n} \times 2\sqrt{n}} = \lim_{n\to\infty} \sqrt{n} = \infty$$

$$\Rightarrow \lim_{n\to\infty} \frac{n^{2.5}}{(n \lg n)^2} = \infty$$

Notice, we applied L'Hôpital's rule twice. We can prove $\lim_{n\to\infty} \frac{n^b}{\lg^k n} = \infty$ for any $b > 0$ and any integer k by applying the rule k times.

6. Let $f(n) = n + \ln n$ and let $t(n) = n$. Find a positive rational number c and a positive integer n_0 such that, for all $n \geq n_0$, $f(n) \leq ct(n)$.

   Answer:

   To find the positive rational number c:

$$f(n) \leq ct(n)$$

$$n + \ln n \leq cn$$

$$\ln n \leq (c-1)n$$

$$1 + \frac{\ln n}{n} \leq c$$

   So, if n goes to ∞, then $\frac{\ln n}{n}$ goes to 0. That is, $1 + \frac{\ln n}{n}$ is closer to 1.

   $\Rightarrow$ we can choose $c = 2$.

   To find the value of n_0:

$$n + \ln n \leq cn$$

$$n + \ln n \leq 2n$$

$$\ln n \leq n$$

   $\Rightarrow$ we have $n_0 = 1$.

   NOTE: You can verify it by plugging $n = 1$ in the equality:

$$\ln 1 \leq 1$$

$$0 \leq 1$$

   Thus, the inequality holds.

   $\Rightarrow c = 2$ and $n_0 = 1$.

7. Let $f(n) = 2^n$. Let $g(n) = 1.7^n$. Using the defintion of $O(g(n))$, prove or disprove: $f(n) \in O(g(n))$.

Answer:

In order to disprove that $f(n) \in O(g(n))$, consider a constant c as:

$$f(n) \leq cg(n)$$

$$2^n \leq c \cdot 1.7^n$$

$$\frac{2^n}{1.7^n} \leq c$$

$$(\frac{2}{1.7})^n \leq c$$

$$n \log_c (\frac{2}{1.7}) \leq \log_c c$$

$$nlog_c(\frac{2}{1.7}) \leq 1$$

$$n \leq \frac{1}{log_c(\frac{2}{1.7})}$$

$\Rightarrow$ for all values of $n > \frac{1}{log_c \frac{2}{1.7}}$, $f(n) \leq cg(n)$ fails.

OR

We can also disprove this by using limit theorem:

It states that: If $lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, then $f(n) \in O(g(n))$.

So, we have:

$$lim_{n\to\infty} \frac{2^n}{1.7^n} = \infty$$

$$\Rightarrow f(n) \notin O(g(n))$$

.

8. Let $f(n) = n^2 + 100 \lg n$.

   (a) When $n = 2$, is $f(n)$ closer to n^2 or to $100 \lg n$?

   (b) Roughly what is the smallest n such that $n^2 \geq 100 \lg n$?

   (c) What is the simplest function $g(n)$ such that $f(n) \in \Theta(g(n))$?

   Answer:

   (a) 100 lg(n)

   (b) Notice $100 \lg n$ is only defined on $n > 0$, and $\lim_{n \to 0^+} 100 \lg n = -\infty$, while $n^2 > 0, \forall n > 0$. So there doesn't exist a smallest n such that $n^2 > 100 \lg n$, as there's always an n' closer to 0 that also suffices. If we consider only $n > 1$, then the smallest n that suffices is ≈ 21.

   (c) As n increases, n^2 grows much faster than $100 \log(n)$. Thus, the simplest function $g(n)$ such that $f(n) \in \Theta(g(n))$ is n^2.

9. a) For an input of size (number of bits) n, an algorithm has runtime in the set $\Theta(n^{2.5})$. For $n = 3.7e6$ (scientific notation: 3.7×10^6), the runtime is 11s. What is your best guess for the runtime when $n = 7.4e6$? Explain.

   b) Is it possible that your guess in a) will be 10 times too small? Explain.

   Answer:

   (a) Let $f(n) = \alpha n^{2.5}$ be our guess for the runtime of the algorithm on input size n, with α a constant scalar (to scale $f(3.7e6)$ to 11 seconds). Clearly, $f(n) \in \Theta(n^{2.5})$. Then we have,

$$\frac{f(7.4e6)}{f(3.7e6)} = \frac{\alpha(7.4e6)^{2.5}}{\alpha(3.7e6)^{2.5}} = \frac{7.4^{2.5}}{3.7^{2.5}} = 2^{2.5} \approx 5.5 \Rightarrow f(7.4e6) \approx 5.5 \times 11 \approx 60$$

   Our guess is 60 seconds.

   (b) Consider $g(n) = \beta(n^{2.5} + (\ln n)^{90})$, with β a constant scalar (to scale $g(3.7e6)$ to 11s). $g(n) \in \Theta(n^{2.5})$. Using a calculator, such as this python script (`https://github.com/ryanbhayward/algs/blob/master/thetademo.py`), we see that $g(7.4e6)/g(3.7e6) \approx 55$, which means, the actual runtime could possibly be at 600 seconds—our estimate of $f(7.4e6)$ might be 10 times too small.

10. For non-negative integers x, y with $x < y$, explain why the runtime of the usual addition algorithm is in $O(\lg y)$.

    Answer:

    The addition operation of x and y is done by adding their binary representation (which have size $\lg x \in \mathcal{O}(\lg x)$ and $\lg y \in \mathcal{O}(\lg y)$ respectively): digit by digit, starting from the least significant. For each digit addition: there are at most 3 operations: the addition itself, at most 1 operation to generate a carry over to the next, and at most 1 operation to incorporate any carry over from the previous—each of them takes $\mathcal{O}(1)$, so total $\mathcal{O}(1)$. There are $\mathcal{O}(\lg y)$ digits (we assumed $y > x$). So in total, $\mathcal{O}(\lg y)$.

11. Repeat the previous question for time $\Omega(\lg y)$.

Answer:

The binary representation of y has size $\lg y \in \Omega(\lg y)$. In the best case: if there is no carry over, the algorithm still has to iterate over $\Omega(\lg y)$ digits of y. So in total, $\Omega(\lg y)$.

12. Recall iterative Fibonacci:

`https://webdocs.cs.ualberta.ca/~hayward/304/jem/warmup.html#ifib`

a) Justify each step in the runtime analysis:

`https://webdocs.cs.ualberta.ca/~hayward/304/asn/itfib.pdf`

b) Explain why the runtime of `ifib(n)` is proportional to $\sum_{j=1}^{n} \lg(\text{fib}(j))$.

c) Explain why the above sum is in $O(n^2)$.

d) Let $r(n)$ be the runtime for `ifib(n)`. For a positive integer t, explain why we expect that $r(2^{t+1})/r(2^t)$ will be around 4.

Answer:

Answer (a-c). Note the main function in iterative Fibonacci:

```
def ifib(n):
  a,b = 0,1
  for _ in range(n):
    a, b = b, a+b
  return a
```

To calculate $Fib(n)$ using this function, the loop needs to execute n times and preform the addition n times. Before the start of the loop, a, b are set to the start of the Fibonacci sequence. As the loop continues, by induction, $a+b$ will produce the rest of the sequence. So after the first iteration, $f(2)$ is calculated, and so on. The cost of this addition is proportional to the size of the result, which is $\Theta(\log(f(2)))$ in the first loop (because it takes $\Theta(\log(f(2)))$ to store $f(2)$) and then the second loop will take $\Theta(\log(f(3)))$ to calculate $f(3)$ and so on. Therefore, you can represent the time it takes for ifib(n) to calculate $f(n)$ like this:

$$r(n) \in \Theta(\log f(2) + \log f(3) + \cdots + \log f(n))$$
$$r(n) \in \Theta(\sum_{j=2}^{n} \log f(j))$$

The second line is just the short form of that summation. Now to simplify this sum, we can use Binet's formula. As a reminder, Binet's formula proposes a closed form expression for the Fibonacci sequence:

$$f(n) = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}}$$

As $(\frac{1-\sqrt{5}}{2})^n$ is small enough, we can rewrite that expression and say that $f(n)$ is in:

$$\Theta(\frac{(\frac{1+\sqrt{5}}{2})^n}{\sqrt{5}}).$$

Now using this, we can simplify the runtime:

$$r(n) \in \Theta(\sum_{j=2}^{n} \log f(j)) \tag{1}$$

$$r(n) \in \Theta(\sum_{j=2}^{n} \log \frac{(\frac{1+\sqrt{5}}{2})^j}{\sqrt{5}}) \tag{2}$$

$$r(n) \in \Theta(\sum_{j=2}^{n} j \log \frac{\frac{1+\sqrt{5}}{2}}{\sqrt{5}}) \tag{3}$$

$$r(n) \in \Theta(\sum_{j=2}^{n} j) \tag{4}$$

$$r(n) \in \Theta(n^2) \tag{5}$$

Line (3) follows from the rule that $\log n^c = c \log n$, Line (4) follows from the fact that the log is now a fixed number under Θ and the last line is the closed form of the sum.

Answer (d)
We already know that $r(2^t) \in \Theta((2^t)^2) = \Theta(2^{2t})$ and $r(2^{t+1}) \in \Theta((2^{t+1})^2) = \Theta(2^{2t+2})$. Then it's clear that the ratio between them is 4.

13. Recall recursive `fib(n)`:

```
def fib(n):
  if (n<=1): return n
  return fib(n-1) + fib(n-2)
```

Complete the proof of the following claim.

Claim. For all integers $n \geq 0$, `fib(n)` returns $f(n)$, where $f(n)$ is defined as 0 if n is 0, 1 if n is 1, and the sum of $f(n-1)$ and $f(n-2)$ when n is at least 2.

Proof. Argue by induction on n. The claim holds when n is 0 or 1 (why?).

Let x be an integer greater than or equal to 2. Assume that the claim holds for all values of n in the set $\{0, 1, \ldots, x-1\}$. In order to complete the proof, we now want to show that the claim holds when n is x, i.e. that `fib(x)` returns $f(x)$.

So what happens when `fib(x)` executes? Well, $x \geq 2$, so the `if` test evaluates to false, so the program returns `fib(x-1)+fib(x-2)`.

(now finish the proof . . .)

Answer:

Base Case: The claim holds when $n = 0$ and $n = 1$ (if statement evaluates to true):

- For $n = 0$: $\text{fib}(0) = 0 = f(0)$.
- For $n = 1$: $\text{fib}(1) = 1 = f(1)$.

Let x be an integer such that $x \geq 2$. Assume the claim holds for all values of n in the set $\{0, 1, \ldots, x-1\}$. Therefore, for all k where $0 \leq k < x$, we have:

$$\text{fib}(k) = f(k).$$

Inductive Step: We now want to show that the claim holds when $n = x$, i.e., that $\text{fib}(x)$ returns $f(x)$.

When $\text{fib}(x)$ executes, since $x \geq 2$, the condition in the if statement evaluates to false. Thus, the function returns:

$$\text{fib}(x) = \text{fib}(x-1) + \text{fib}(x-2).$$

From our above assumption,

$$\text{fib}(x-1) = f(x-1) \text{ and } \text{fib}(x-2) = f(x-2).$$

Substituting these into our expression:

$$\text{fib}(x) = \text{fib}(x-1) + \text{fib}(x-2) = f(x-1) + f(x-2).$$

By the definition of $f(n)$:

$$f(x) = f(x-1) + f(x-2) \quad \text{for } n \geq 2.$$

Thus, we have shown that:

$$\text{fib}(x) = f(x).$$

We have verified the base cases for $n = 0$ and $n = 1$, and shown that if the claim holds for all integers less than x, it also holds for x. Therefore, we can conclude fib(n) returns f(n) for all integers $n \geq 0$ by mathematical induction.