CSE 3101 Design and Analysis of Algorithms
Solutions for Practice Test for Unit 0
Relevant Math

Jeff Edmonds

Chapter 22: Existential and Universal Quantifiers

1. For each prove whether true or not when each variable is a real value. Be sure to play the correct game as to who is providing what value.

   1) $\forall x\ \exists y\ x+y=5$ 2) $\exists y\ \forall x\ x+y=5$
   3) $\forall x\ \exists y\ x\cdot y=5$ 4) $\forall x\ \exists y\ x\cdot y=0$
   5) $[\forall x\ \exists y\ P(x,y)] \Rightarrow [\exists y\ \forall x\ P(x,y)]$
   6) $[\forall x\ \exists y\ P(x,y)] \Leftarrow [\exists y\ \forall x\ P(x,y)]$
   8) $\exists a\ \forall x\ \exists y\ [x=0 \text{ or } x\cdot y=5]$

   - Answer:
     (a) $\forall x\ \exists y\ x+y=5$ is true. Let x be an arbitrary real value and let $y=5-x$. Then $x+y=5$.
     (b) $\exists y\ \forall x\ x+y=5$ is false, because $\forall y\ \exists x\ x+y\neq 5$ is true. Let y be an arbitrary real value and let $x=6-y$. Then $x+y=6\neq 5$.
     (c) $\forall x\ \exists y\ x\cdot y=5$ is false, because $\exists x\ \forall y\ x\cdot y\neq 5$ is true. Let $x=0$ and let y be an arbitrary real value. Then $x\cdot y=0\neq 5$. Note that y must be $\frac{5}{0}$, which is impossible.
     (d) $\forall x\ \exists y\ x\cdot y=0$ is true. Let x be an arbitrary real value and let $y=0$. Then $x\cdot y=0$. The odd thing about this example is that even though the value of y can depend on the value of x, it does not. $\exists y\ \forall x\ x\cdot y=0$ is a stronger statement that is also true.
     (e) $[\forall x\ \exists y\ P(x,y)] \Rightarrow [\exists y\ \forall x\ P(x,y)]$ is false. Let $P(x,y)=[x+y=5]$. Then as seen above the first is true and the second is false.
     (f) $[\forall x\ \exists y\ P(x,y)] \Leftarrow [\exists y\ \forall x\ P(x,y)]$ is true. Assume the right is true. Let y_0 the that for which $[\forall x\ P(x,y_0)]$ is true. We prove the left as follows. Let x be an arbitrary real value and let $y=y_0$. Then $P(x,y_0)$ is true.
     (g) $\exists a\ \forall x\ \exists y\ [x=0 \text{ or } x\cdot y=5]$ is true. Let $a=0$. Let x be an arbitrary real value. If $x=0$ then $[x=0 \text{ or } x\cdot y=5]$ is true because of the left. If $x\neq 0$ then let $y=\frac{5}{x}$ and $[x=0 \text{ or } x\cdot y=5]$ is true because of the right.

2. The game *Ping* has two rounds. Player-A goes first. Let m_1^A denote his first move. Player-B goes next. Let m_1^B denote his move. Then player-A goes m_2^A and player-B goes m_2^B. The relation $AWins(m_1^A, m_1^B, m_2^A, m_2^B)$ is true iff player-A wins with these moves.

   (a) Use universal and existential quantifiers to express the fact that player-A has a strategy in which he wins no matter what player-B does. Use $m_1^A, m_1^B, m_2^A, m_2^B$ as variables.
   (b) What steps are required in the Prover/Adversary technique to prove this statement?
   (c) What is the negation of the above statement in standard form?
   (d) What steps are required in the Prover/Adversary technique to prove this negated statement?

   - Answer: Regarding the game *Ping*.
     (a) The statement that player-A has a strategy in which he wins no matter what player-B does is $\exists m_1^A \forall m_1^B \exists m_2^A \forall m_2^B\ AWins(m_1^A, m_1^B, m_2^A, m_2^B)$. His strategy specifies his first move m_1^A. Then for each move m_1^B his opponent makes, he must specify his next move m_2^A. This must lead to a win no matter what his opponents next move is.
     (b) The Prover/Adversary technique to prove this statement is a strategy for the prover to win the following game. The prover gives m_1^A, the adversary give m_1^B, the prover gives m_2^A, the adversary give m_2^B, and the prover wins if $AWins(m_1^A, m_1^B, m_2^A, m_2^B)$ is true.

(c) The negation of the above statement is $\forall m_1^A \exists m_1^B \forall m_2^A \exists m_2^B \neg AWins(m_1^A, m_1^B, m_2^A, m_2^B)$.

(d) The Prover/Adversary technique to prove this negated statement is a strategy for the prover to win the game when he takes the role of player-B.

3. Let $Works(P, A, I)$ to true if algorithm A halts and correctly solves problem P on input instance I. Let $P = Halting$ be the Halting problem which takes a Java program I as input and tells you whether or not it halts on the empty string. Let $P = Sorting$ be the sorting problem which takes a list of numbers I as input and sorts them. For each part, explain the meaning of what you are doing and why you don't do it another way.

   Extra:
   Let $A_{insertionsort}$ be the sorting algorithm which we learned in class.
   Let A_{yes} be the algorithm that on input I ignores the input and simply halts and says "yes".
   Let A_{∞} be the algorithm that on input I ignores the input and simply runs for ever.

   (a) Recall that a problem is *computable* if and only if there is an algorithm that halts and returns the correct solution on every valid input. Express in first order logic that $Sorting$ is computable.

   (b) Express in first order logic that $Halting$ is not computable.

   (c) Express in first order logic that there are uncomputable problems.

   (d) What does the following mean and either prove or disprove it: $\forall I, \exists A, Works(Halting, A, I)$. (Not simply by saying the same in words "For all I, exists A, $Works(Halting, A, I)$")

   (e) What does the following mean and either prove or disprove it $\forall A, \exists P, \forall I, Works(P, A, I)$. Hint: An algorithm A on an input I can either halt and give the correct answer, halt and give the wrong answer, or run for ever.

   - Answer:
     (a) $\exists A, \forall I, Works(Sorting, A, I)$. We know that there at least one algorithm, eg. $A = mergesort$, that works for every input instance I.
     (b) $\forall A, \exists I, \neg Works(Halting, A, I)$ We know that opposite statement is true. Every algorithm fails to work for at least one input instance I.
     (c) $\exists P, \forall A, \exists I, \neg Works(P, A, I)$
     (d) It says that every input has some algorithm that happens to output the right answer. It is true. Consider an arbitrary instance I. If on instance I, $Halting$ happens to say yes, then let A be the algorithm that simply halts and says "yes". Otherwise, let A be the algorithm that simply halts and says "no". Either way A "works" for this instance I.
     (e) It says that every algorithm correctly solves some problem. This is not true because some algorithm do not halt on some input instances. We prove the complement $\exists A, \forall P, \exists I, \neg Works(P, A, I)$ as follows. Let A be an algorithm that runs for ever on some instance I'. Let P be an arbitrary problem. Let I be an instance I' on which A does not halt. Note $Works(P, A, I)$ is not true.

4. Quantifiers:

   (a) $\exists y \ \forall x \ x \cdot y = 0$
   Prove whether true or not when each variable is a real value. Be sure to play the correct game as to who is providing what value.

   - Answer: $\exists y \ \forall x \ x \cdot y = 0$ is true. Let $y = 0$ and let x be an arbitrary real value. Then $x \cdot y = 0$.

   (b) $\exists y \ \forall x \ x \cdot y = 5$

   - Answer: $\exists y \ \forall x \ x \cdot y = 5$ is false because $\forall y \ \exists x \ x \cdot y \neq 5$ is true. Let y be an arbitrary real value and let $x = 0$. Then $x \cdot y \neq 5$.

(c) A computational problem is a function P from inputs I to required output $P(I)$. An algorithm is a function A from inputs I to actual output $A(I)$. $Time(A, I)$ gives the running time for algorithm A on input I. Let $T(n)$ be some function. Use universal and existential quantifiers to express "Problem P is computable in time $T(n)$". (No BigOh)

- Answer: $\exists A,\ \forall I,\ A(I) = P(I)$ and $Time(A, I) \leq T(n)$

(d) Express "Problem P is not computable in time $T(n)$".

- Answer: $\forall A,\ \exists I,\ A(I) \neq P(I)$ or $Time(A, I) > T(n)$

Chapter 23: Time Complexity

5. This problem compares the running times of the following to algorithms for multiplying.

algorithm $KindergartenAdd(a, b)$

$\langle$***pre−cond***$\rangle$**:** a and b are integers.

$\langle$***post−cond***$\rangle$**:** Outputs $a \times b$.

```
begin
    c = 0
    loop i = 1 ... a
        c = c + b
    end loop
    return(c)
end algorithm
```

algorithm $GradeSchoolAdd(a, b)$

$\langle$***pre−cond***$\rangle$**:** a and b are integers.

$\langle$***post−cond***$\rangle$**:** Outputs $a \times b$.

```
begin
    Let a_{s-1} ... a_3 a_2 a_1 a_0 be the decimal digits of a, so that a = sum_{i=0}^{s-1} a_i · 10^i.
    Let b_{t-1} ... b_3 b_2 b_1 b_0 be the decimal digits of b, so that b = sum_{j=0}^{t-1} b_j · 10^j.
    c = 0
    loop i = 0 ... s - 1
        loop j = 0 ... t - 1
            c = c + a_i · b_j · 10^{i+j}.
        end loop
    end loop
    return(c)
end algorithm
```

For each of these algorithms, do the following questions.

(a) Suppose that each addition to c requires time 10 seconds and every other operation (for example multiplying two single digits 9×8 and shifting by zero) is free. What is the running time of each of these algorithms either as a function of a and b or as a function of s and t? Give everything for this entire question exactly, i.e. not bigOh.

(b) Let $a = 9,168,391$ and $b = 502$. (Without handing it in trace the algorithm.) With 10 seconds per addition, how much time (seconds,minutes,...) does the computation require?

(c) The formal "size" of an input instance is the number of bits n to write it down. What is n as a function of our instance $\langle a, b \rangle$?

(d) Suppose your job is to choose the worst case instance $\langle a, b\rangle$ (i.e. that which maximizes the running time), but you are limited that you can only use n bits to represent your instance. Do you set a big and b small, a small and b big, or a and b to be the same size? Give the worst case a and b or s and t as a function of n.

(e) The "Running Time" of an algorithm is formally defined to be a function $T(n)$ from n to the time required for the computation on the worst case instance of size n. Give $T(n)$ for each of these algorithms. Is this polynomial time?

- Answer:
  (a) *KindergartenAdd*: The computation requires a additions for a total of $T(a, b) = 10a$ seconds.
  GradeSchoolAdd: The computation requires $s{\cdot}t$ additions for a total of $T(s, t) = 10st$ seconds.
  (b) *KindergartenAdd* requires $10a = 9,168,391\ additions\ \times 10\frac{sec}{additions} \times \frac{1}{60}\frac{min}{sec} \times \frac{1}{60}\frac{hours}{min} \times \frac{1}{24}\frac{days}{hours} \times \frac{1}{365}\frac{years}{days} = 3\ years.$
  GradeSchoolAdd requires $10st = 7 \cdot 3\ additions\ \times 10\frac{sec}{additions} \times \frac{1}{60}\frac{min}{sec} = 3.5$ minutes.
  (c) $n = \log_2 a + \log_2 b$.
  (d) *KindergartenAdd*: The time depends on a and not on b. Hence, we set $a = 2^n$ to be its maximum value and $b = 1$ to be as small as possible so it is not using up bits of the instance.
  GradeSchoolAdd: Because the time depends on the product of info about a and b, the worst case is when $a = b$. Hence each uses $\frac{1}{2}n$ bits, giving $a = b = 2^{\frac{1}{2}n}$. We have that $s = t = log_{10}(a) = log_{10}(2^{\frac{1}{2}n}) = \frac{1}{2}n \cdot log_{10}(2) = 0.15n$.
  (e) *KindergartenAdd*: $T(n) = T(a, b) = 10a = 10 \cdot 2^n$. This is not polynomial but exponential time.
  GradeSchoolAdd: $T(n) = T(s, t) = 10st = 10 \cdot (0.15n) \cdot (0.15n) = 0.225n^2$. This is quadratic time, which is polynomial time.

6. The input is an n bit integer x. The output is $\pi(x)$ which is the x^{th} digit of real number $\pi = 3.14\ldots$ written in decimal. What is the time complexity of the following algorithms?

   (a) It could take $Time\ 1/\epsilon$ to get π to accuracy within ϵ.
   - Answer: Then it takes $2^x = 2^{(2^n)}$ time to find the x^{th} bit. Here the size of the input is n, because x is an n bit integer.

   (b) It could take $\Theta(1)$ time to get the next bit of accuracy of π.
   - Answer: Hence it takes $Time = \Theta(x) = \Theta(2^n)$ to get to the x^{th} bit.

   (c) I think I heard that they can now find the x^{th} bit without first finding all the earlier bits.
   - Answer: This might take $poly(n)$ time.

7. Time Complexity: Consider the two simple algorithms, summation and factoring.

   Summation: The task is to sum the N entries of an array, that is, $A(1) + A(2) + A(3) + \ldots + A(N)$.

   Factoring: The task is to find divisors of an integer N. For example, on input $N = 5917$ we output that $N = 97 \times 61$. (This problem is central to cryptography.) The algorithm checks whether N is divisible by 2, by 3, by 4, ... by N.

   Give the time complexity of each.

   - Answer: Both algorithms take $T = \Theta(N)$ time to complete. For Summation, the instance size is $n = \Theta(N)$ bits. Hence, its time complexity is $T = \Theta(N) = \Theta(n)$. This is linear time. For Factoring, the instance size is $n = \Theta(\log N)$ bits. Hence, its time complexity is $T = \Theta(N) = 2^{\Theta(n)}$. This is exponential time.

Chapter 24: Logarithms and Exponentials

8. Simplify the following exponentials: $a^3 \times a^5$, $3^a \times 5^a$, $3^a + 5^a$, $2^{6log_4 n+7}$, $n^{\frac{3}{log_2 n}}$.

   - Answer: $a^3 \times a^5 = a^8$, $3^a \times 5^a = 15^a$, $3^a + 5^a =?$, $2^{5log_4 n+7} = [4^{0.5}]^{6log_4 n} \cdot 2^7 = [4^{log_4 n}]^3 \cdot 128 = 128n^3$, $n^{\frac{3}{log_2 n}} = [2^{\log_2 n}]^{\frac{3}{log_2 n}} = 2^3 = 8$.

9. Logarithms and Exponentials: Simplify the following exponentials: $a^3 \times a^5$, $3^a \times 5^a$, $3^a + 5^a$, $2^{6log_4 n+7}$, $n^{\frac{3}{log_2 n}}$.

   - Answer: $a^3 \times a^5 = a^8$, $3^a \times 5^a = 15^a$, $3^a + 5^a =?$, $2^{5log_4 n+7} = [4^{0.5}]^{6log_4 n} \cdot 2^7 = [4^{log_4 n}]^3 \cdot 128 = 128n^3$, $n^{\frac{3}{log_2 n}} = [2^{\log_2 n}]^{\frac{3}{log_2 n}} = 2^3 = 8$.

Chapter 25: Asymptotic Growth

10. For each of the following functions sort its terms by growth rate. Get the big picture growth by Classifying it into $2^{\Theta(n)}$, $n^{\Theta(1)}$, $\log^{\Theta(1)}(n)$, $\Theta(1)$ or into a similar and appropriate class. Also give its Theta approximation.

    (a) $f(n) = 5n^3 - 17n^2 + 4$
    (b) $f(n) = 5n^3 \log n + 8n^3$
    (c) $f(n) = 2^{2^{5n}}$
    (d) $f(n) = 7^{3\log_2 n}$
    (e) $f(n) = \{$ 1 if n is odd, 2 if n is even $\}$
    (f) $f(n) = 2 \cdot 2^n \cdot n^2 \log_2 n - 7n^8 + 7\frac{3^n}{n^2}$
    (g) $f(n) = 100n^{100} + 3^{4n} + \log^{1,000}(n) + 4^{3n}$
    (h) $f(n) = 6\frac{n^4}{\log^3 n} + 8n^{100}2^{-5n} + 17$
    (i) $f(n) = \frac{1}{n^2} + \frac{5\log n}{n}$
    (j) $f(n) = 7\sqrt[5]{n} + 6\sqrt[3]{n}$
    (k) $f(n) = \frac{6n^{5.2}+7n^{7.5}}{2n^{3.1}+7n^{2.4}}$
    (l) $f(n) = -2n$
    (m) $f(n) = 5n^{\log^3 n}$

    - Answer:
      (a) The terms are $5n^3 >> 17n^2 >> 4$ for sufficiently large n. Hence, $f(n) \in \Theta(n^3)$ and in $n^{\Theta(1)}$. Here $n^{\Theta(1)}$ consists of the function with the asymptotics of polynomials.
      (b) The terms are $5n^3 \log n >> 8n^3$. Hence, $f(n) \in \Theta(n^3 \log n)$. Note $f(n) \notin \Theta(n^3)$ because $\log n$ grows faster than a constant. On the other hand, $f(n) \in n^{\Theta(1)}$ because for sufficiently large n, it is bounded between n^3 and n^4. This is a little confusing, because $f(n)$ is not itself a polynomial.
      (c) $f(n) = 2^{2^{5n}}$ is double exponential $2^{2^{\Theta(n)}}$ which grows far too quickly too be exponential. $f(n) = \Theta(2^{2^{5n}})$.
      (d) $f(n) = 7^{3\log_2 n} = (2^{\log_2 7})^{(3\log_2 n)} = 2^{\log_2 7 \cdot 3\log_2 n} = (2^{\log_2 n})^{3\log_2 7} = n^{2.5719\ldots}$ which is in $\Theta(n^{2.5719\ldots})$ and in $n^{\Theta(1)}$.
      (e) $f(n) = \{$ 1 if n is odd, 2 if n is even $\}$ is not "constant", but because it is is bound between 1 and 2 it is in $\Theta(1)$. It is a little confusing, but $\Theta(1)$ is referred to as the set set of constant functions.
      (f) Note the exponential 3^n is so much bigger than 2^n for large n that the first dominates even when divided by the small polynomial n^2. Hence, the terms are $7\frac{3^n}{n^2} >> 2 \cdot 2^n \cdot n^2 \log_2 n >> 7n^8$. This gives that $f(n) \in \Theta(\frac{3^n}{n^2})$. Because $f(n)$ is bounded between 2^n and 3^n, we have that $f(n) \in 2^{\Theta(n)}$.
      (g) The terms are $3^{4n} = 81^n >> 4^{3n} = 64^n >> 100n^{100} >> \log^{1,000}(n)$, the first two of which are exponential, the third polynomial, and the last poly-log. Hence, $f(n) \in \Theta(81^n)$. Because $81^n = 2^{6.339\ldots n}$, we have that $f(n) \in 2^{\Theta(n)}$.
      (h) The term $8n^{100}2^{-5n}$ exponentially decreasing. Dividing by the exponential 2^{5n} just kills the poor polynomial n^{100}. Thought of another way $b^a = 2^{-5} = \frac{1}{32}$. On the other hand, the logarithm $\log^3 n$ has little effect on the polynomial n^4. Hence, the terms are $6\frac{n^4}{\log^3 n} >> 17 >> 8n^{100}2^{-5n} + 17$. Hence, $f(n) \in \Theta(\frac{n^4}{\log^3 n})$ and in $n^{\Theta(1)}$.

(i) The terms are $\frac{5\log n}{n} >> \frac{1}{n^2}$. The n^2 increases faster n and hence $\frac{1}{n^2}$ shrink faster than $\frac{1}{n}$. Thought of another way, the $d = -1$ for the first is bigger than the $d = -2$ for the second. Hence, $f(n) \in \Theta(\frac{5\log n}{n})$. This shrinks and hence is smaller than constant giving that $f(n) \in \mathcal{O}(1)$, but $f(n) \notin \Theta(1)$. It is $\frac{1}{n^{\Theta(1)}}$.

(j) The terms are $6\sqrt[3]{n} = 6n^{\frac{1}{3}} >> 7\sqrt[5]{n} = 7n^{\frac{1}{5}}$. The $d = \frac{1}{3}$ for the first is bigger than the $d = \frac{1}{5}$ for the second. Hence, $f(n) \in \Theta(n^{\frac{1}{3}})$ and in $n^{\Theta(1)}$.

(k) $f(n) = \frac{6n^{5.2}+7n^{7.5}}{2n^{3.1}+7n^{2.4}} \approx \frac{7n^{7.5}}{2n^{3.1}} = 3.5n^{4.4}$ Hence, $f(n) \in \Theta(n^{4.4})$ and in $n^{\Theta(1)}$.

(l) The function $f(n) = -2n$ is in none or our classes because $c < 0$. One could say, however, that it is in $-\Theta(n)$.

(m) The function $f(n) = 5n^{\log^3 n}$ is too big to be in $n^{\Theta(1)}$ because $\log^3 n$ is bigger than a constant. It is also too small to be in $2^{\Theta(n)}$ because $f(n) = 5n^{\log^3 n} = 5\left[2^{\log n}\right]^{\log^3 n} = 5 \cdot 2^{\log^4 n}$ and $\log^4 n$ is smaller than $\Theta(n)$.

11. Suppose that $y = \Theta(\log x)$. Which of the following are true: $x = \Theta(2^y)$ and $x = 2^{\Theta(y)}$? Why?

- Answer: Suppose $y = \Theta(\log x)$. Effectively, this means that $y = c \cdot \log x$ for some unknown constant $c \in \Theta(1)$. This gives $\log x = \frac{1}{c}y$ and $x = 2^{\frac{1}{c}y} \in 2^{\Theta(y)}$. It also gives that $x = \left(2^{\frac{1}{c}}\right)^y \neq \Theta(2^y)$, unless $c = 1$.

Chapter 26: Adding-Made-Easy Approximations

12. Give the Θ approximation of the following sums. Indicate which rule you use and show your work.

(a) $\sum_{i=0}^{n} 7i^3 - 300i^2 + 16$

(b) $\sum_{i=0}^{n} i^8 + \frac{2^{3i}}{i^2}$

(c) $\sum_{i=0}^{n} \frac{1}{i^{1.1}}$

(d) $\sum_{i=0}^{n} \frac{1}{i^{0.9}}$

(e) $\sum_{i=0}^{n} 7\frac{i^{3.72}}{\log^2 i} - 300i^2 \log^9 i$

(f) $\sum_{i=1}^{n} \frac{\log^e i}{i}$

(g) $\sum_{i=1}^{\log n} n \cdot i^2$

(h) $\sum_{i=0}^{n} \sum_{j=0}^{m} \frac{j}{i}$

(i) $\sum_{i=1}^{n} \sum_{j=1}^{i} j^i$

(j) $\sum_{i=1}^{n} \sum_{j=1}^{i^2} ij \log(i)$

- Answer:

(a) The function $f(i) = 7i^3 - 300i^2 + 16 = n^{\Theta(1)}$ is a polynomial. Note that $b^a = 1$ and $d = 3 > -1$. Hence, half the terms in the sum $\sum_{i=0}^{n} 7i^3 - 300i^2 + 16$ are approximately equal. Hence, the sum evaluates to being within a constant of the number of terms n times the largest term $f(n) = 7n^3 - 300n^2 + 16$, giving $\Theta(n^4)$. We call such a sum *arithmetic-like*.

(b) The function $f(i) = i^8 + \frac{2^{3i}}{i^2} = 2^{\Theta(i)}$ is exponential. Note that $b^a = 2^3 = 8 > 1$. Hence, the sum $\sum_{i=0}^{n} i^8 + \frac{2^{3i}}{i^2}$ is dominated by the last term giving that it evaluates to $\Theta(f(n)) = \Theta(\frac{2^{3n}}{n^2})$. We call such a sum *geometric*.

(c) The sum $\sum_{i=0}^{n} \frac{1}{i^{1.1}}$ has $d = -1.1 < -1$. Hence, the sum is dominated by the first term, evaluating to $\Theta(1)$. We call such a sum *bounded tail*.

(d) The sum $\sum_{i=0}^{n} \frac{1}{i^{0.9}}$ has $d = -0.9 > -1$. Hence even though the terms are decreasing, the sum still evaluates asymptotically to the number of terms n times the largest term $f(n) = \frac{1}{n^{0.9}}$, giving $\Theta(n^{1-0.9}) = \Theta(n^{0.1})$ which grows with n. We still call such a sum *arithmetic-like*.

(e) $f(i) = \frac{i^{3.72}}{\log^2 i} - 300i^2 \log^9 i + 16 = i^{\Theta(1)}$, with $d = 3.72 > -1$. Hence the sum is arithmetic-like and evaluates to $\Theta(n \cdot f(n)) = \Theta(\frac{n^{4.72}}{\log^2 n})$. Note you can't ignore the $\log^2 n$ when computing the Theta.

(f) Consider $\sum_{i=1}^{n} \frac{\log^e i}{i}$. If $e = 0$, then $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\ln n)$, which is the classic *harmonic sum*. The chart does not cover this for the other values of e. However, for $e > -1$, $\sum_{i=1}^{n} \frac{\log^e i}{i} = \Theta(\ln^{e+1} n)$. This is can best be understood as $\approx \log^e n \cdot \sum_{i=1}^{n} \frac{1}{i} = \log^e n \cdot \Theta(\ln n)$. For $e = -1$, $\sum_{i=1}^{n} \frac{1}{i \log i} = \Theta(\ln \ln n)$. For $e < -1$, $\sum_{i=1}^{n} \frac{1}{i \log^{-e} i} = \Theta(1)$.

(g) $\sum_{i=1}^{\log n} n \cdot i^2 = n \cdot \sum_{i=1}^{N} i^2 = n \cdot \Theta(N^3) = \Theta(n \log^3 n)$. Even easier, because the function $f(i) = n \cdot i^2$ is polynomial, the sum is arithmetic-like, giving that it evaluates to the number of terms $\log n$ times the largest term $n \log^2 n$, giving $\Theta(n \log^3 n)$.

(h) $\sum_{i=0}^{n} \sum_{j=0}^{m} \frac{j}{i} = \sum_{i=0}^{n} \frac{1}{i} \Theta(m^2) = \Theta(m^2 \log(n))$, Arithmetic & Harmonic.

(i) For every constant i, $f(j) = j^i \in j^{\Theta(1)}$ giving that $\sum_{j=1}^{i} j^i = \Theta(\text{number of terms times the last term}) = \Theta(i \cdot f(i)) = \Theta(i \cdot i^i)$. This sum is arithmetic-like. This function $g(i) = \Theta(i \cdot i^i)$ is exponential giving that the sum geometric and is dominated by the last term, i.e. $\sum_{i=1}^{n} \sum_{j=1}^{i} j^i = \sum_{i=1}^{n} \Theta(i \cdot i^i) = \Theta(g(n)) = \Theta(n \cdot n^n)$.

(j) $\sum_{i=1}^{n} \sum_{j=1}^{i^2} ij \log(i) = \sum_{i=1}^{n} i \log(i) \sum_{j=1}^{N} j = \sum_{i=1}^{n} i \log(i) \Theta(N^2) = \sum_{i=1}^{n} \Theta(i^5 \log(i)) = \Theta(n^6 \log(n))$, arithmetic & arithmetic

13. Zeno's classic "paradox" is that Achilles is traveling $1 \frac{km}{hr}$ and has $1km$ to travel. First he must cover half his distance, then half of his remaining distance, then half of this remaining distance, ... He never arrives there. "The Story of Philosophy" by Bryan Magee states "People have found it terribly disconcerting. There must be a fault in the logic, they have said. But no one has yet been fully successful in demonstrating what it is." Resolve this ancient paradox by adding up the time required for each step.

- Answer: His first such step takes him half an hour, his second a quarter, his third an eight, ... for a total of only $\sum_{i=1}^{\infty} \frac{1}{2^i} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \ldots = 1$ hour. Given that he travels one kilometer at one kilometer an hour, this is reasonable.

14. Consider the following functions.

- **Classification:** Provide a Θ-approximation, if possible classifying the function into $\Theta(1)$, $2^{\Theta(n)}$, $\log^{\Theta(1)}(n)$, or $n^{\Theta(1)}$. What is this class called?
- **Summation:** Approximate the sum $\Theta(\sum_{i=1}^{n} f(i))$. Which result did you use and how did you use it? What type of sum is it?

(a) $f(n) = 5 \cdot n^{5.5} + n^{2.5} \log^{100}(n)$

(b) $f(n) = 5 \cdot 3^{4n} + n^{100}$

(c) $f(n) = \frac{5}{n^{1.01}} + \frac{7}{n^{0.99}}$

- Answer:

(a) **Class:** $f(n) = 5 \cdot n^{5.5} + n^{2.5} \log^{100}(n)$. The $n^{2.5} \log^{100}(n)$ is smaller than $n^{2.5001}$ which is clearly smaller that $n^{5.5}$. Hence, $f(n) \in \Theta(n^{5.5}) \subseteq n^{\Theta(1)}$. This is a polynomial function.
Sum: $b^a = 1$ and $d = 5.5 > -1$. Hence, the sum is arithmetic and half the terms are approximately equal. Hence, $\sum_{i=1}^{n} f(i) = \Theta(n \cdot f(n)) = \Theta(n^{6.5})$.

(b) **Class:** $f(n) = 5 \cdot 3^{4n} + n^{100}$ is dominated by the exponential term not the polynomial term. $f(n) \in \Theta(3^{4n}) \subseteq 2^{\Theta(n)}$ is an exponential function.
Sum: $\sum_{i=1}^{n} f(i) = \sum_{i=1}^{n} \Theta(3^{4i}) = \sum_{i=1}^{n} \Theta(81^i)$. Since $81 > 1$, the sum is geometric increasing and dominated by the last term. Hence $\sum_{i=1}^{n} f(i) = \Theta(f(n)) = \Theta(3^{4n})$.

(c) **Class:** $f(n) = \frac{5}{n^{1.01}} + \frac{7}{n^{0.99}} = 5 \cdot n^{-1.01} + 7 \cdot n^{-0.99}$. Because $-0.99 > -1.01$, the second term dominates and $f(n) \in \Theta(\frac{1}{n^{0.99}})$. This decreases with n so is to small to even be considered constant, $\Theta(1)$.
Sum: $b^a = 1$ and $d = -0.99 > -1$. Hence, the sum is still arithmetic and half the terms are approximately equal. Hence, $\sum_{i=1}^{n} f(i) = \Theta(n \cdot f(n)) = \Theta(n^{0.01})$.
Sum: (not asked for) If $f(n) = \frac{5}{n^{1.01}}$, then $b^a = 1$ and $d = -1.01 < -1$. Hence, the sum is bounded tail and the sum is dominated by the first term. Hence, $\sum_{i=1}^{n} f(i) = \Theta(1)$.

Chapter 27: Recurrence Relations

15. Give solutions for the following examples:

(a) $T(n) = 2T(\frac{n}{2}) + n$

(b) $T(n) = 2T(\frac{n}{2}) + 1$

(c) $T(n) = 4T(\frac{n}{2}) + \Theta(\frac{n^3}{\log^3 n})$

(d) $T(n) = 32T(\frac{n}{4}) + \Theta(\log n)$

(e) $T(n) = 27T(\frac{n}{3}) + \Theta(n^3 \log^4 n)$

(f) $T(n) = 8T(\frac{n}{4}) + \Theta((\frac{n}{\log n})^{1.5})$

(g) $T(n) = 4T(\frac{n}{2}) + \Theta(\frac{n^2}{\log n})$

- Answer: Examples:

$T(n)$	$\frac{\log a}{\log b}$	c	d vs -1	Dom.	Rule	Solution
$2T(\frac{n}{2}) + n$	$\frac{\log_2 2}{\log_2 2} = 1$	1	$0 >$	all	$\Theta(f(n) \log n)$	$\Theta(n \log n)$
$2T(\frac{n}{2}) + 1$	$\frac{\log_2 2}{\log_2 2} = 1$	0		base	$\Theta(n^{\log a / \log b})$	$\Theta(n)$
$4T(\frac{n}{2}) + \Theta(\frac{n^3}{\log^3 n})$	$\frac{\log_2 4}{\log_2 2} = 2$	3		top	$\Theta(f(n))$	$\Theta(\frac{n^3}{\log^3 n})$
$32T(\frac{n}{4}) + \Theta(\log n)$	$\frac{\log_2 32}{\log_2 4} = \frac{5}{2}$	0		base	$\Theta(n^{\log a / \log b})$	$\Theta(n^{2.5})$
$27T(\frac{n}{3}) + \Theta(n^3 \log^4 n)$	$\frac{\log_3 27}{\log_3 3} = \frac{3}{1}$	3	$4 >$	all	$\Theta(f(n) \log(n))$	$\Theta(n^3 \log^5 n)$
$8T(\frac{n}{4}) + \Theta((\frac{n}{\log n})^{1.5})$	$\frac{\log_2 8}{\log 4} = \frac{3}{2}$	$\frac{3}{2}$	$-1.5 <$	base	$\Theta(n^{\log a / \log b})$	$\Theta(n^{1.5})$
$4T(\frac{n}{2}) + \Theta(\frac{n^2}{\log n})$	$\frac{\log_2 4}{\log_2 2} = 2$	2	$-1 =$	Ex 27.2.4 gives $\Theta(n^c \cdot \log \log n) = \Theta(n^2 \log \log n)$		

16. Solve the famous Fibonacci recurrence relation $Fib(0) = 0$, $Fib(1) = 1$, and $Fib(n) = Fib(n-1) + Fib(n-2)$ by plugging in $Fib(n) = \alpha^n$ and solving for α.

- Answer: Plugging $Fib(n) = \alpha^n$ into $Fib(n) = Fib(n-1) + Fib(n-2)$ gives that $\alpha^n = \alpha^{n-1} + \alpha^{n-2}$. Dividing through by α^{n-2} gives $\alpha^2 = \alpha + 1$. Solving this gives that either $\alpha = \frac{1+\sqrt{5}}{2}$ or $\alpha = \frac{1-\sqrt{5}}{2}$. Any linear combination of these two solutions will also be a valid solution, namely $Fib(n) = c_1 \cdot (\alpha_1)^n + c_2 \cdot (\alpha_2)^n$. Using the fact that $Fib(0) = 0$ and $Fib(1) = 1$ and solving for c_1 and c_2 gives that $Fib(n) = \frac{1}{\sqrt{5}} \left[\left[\frac{1+\sqrt{5}}{2} \right]^n - \left[\frac{1-\sqrt{5}}{2} \right]^n \right]$.

17. Solve the following by unwinding them.

(a) $T(n) = T(n-1) + n$

(b) $T(n) = 2 \cdot T(n-1) + 1$

- Answer: Unwinding:
  (a) $T(n) = T(n-1) + n$: Because $a = 1$ and $c > 1$, the table give $T(n) = \Theta(n \cdot f(n)) = \Theta(n^2)$. Unwinding gives $T(n) = n + T(n-1) = n + (n-1) + T(n-2) = n + (n-1) + (n-2) + T(n-3) = n + (n-1) + (n-2) + \ldots + (n-i+1) + T(n-i) = n + (n-1) + (n-2) + \ldots 1 = \Theta(n^2)$.
  (b) $T(n) = 2 \cdot T(n-1) + 1$: Because $a = 2$, the table give $T(n) = \Theta(a^{\frac{n}{b}}) = \Theta(2^n)$. Unwinding gives $T(n) = 1 + 2T(n-1) = 1 + 2 + 4T(n-2) = 1 + 2 + 4 + \ldots + 2^{i-1} + 2^i T(n-i) = 1 + 2 + 4 + \ldots + 2^{n-1} + 2^n = \Theta(2^n)$.

18. Recurrence Relations: Strassen's algorithm for matrix multiplication is given two $n \times n$ matrices A and B. In $\mathcal{O}(n^2)$ time, it finds seven pairs of $\frac{n}{2} \times \frac{n}{2}$ matrices. It recursively multiplies each pair. Then in time $\mathcal{O}(n^2)$, it combines the answers to the seven multiplication into the answer for $A \times B$. Let $T(n)$ be the total time for the algorithm given two $n \times n$ matrices. Give the **recurrence relation** for $T(n)$. Solve for it.

- Answer: $T(n) = 7T(n/2) + \Theta(n^2)$. The number of base cases is $\Theta(n^{\frac{\log a}{\log b}}) = \Theta(n^{\frac{\log 7}{\log 2}}) = \Theta(n^{2.8073})$. The work at the top is $\Theta(n^c) = \Theta(n^2)$. Note $2.8073 > 2$. Hence, the time is dominated by the base cases and $T(n) = \Theta(n^{2.8073})$.