CS 341, Winter 2015
Andrew Arnold, Timothy Chan, Doug Stinson

Solutions to Assignment 1 (due January 27 Tuesday 4pm)

Please read `https://www.student.cs.uwaterloo.ca/~cs341/policies.html` for general instructions.

1. Give a complete proof from the definition of ω (not using limits) that

$$n^{3.41} - 2015n^2 + 4 \in \omega(n^3).$$

Answer:

$$\begin{aligned} n^{3.41} - 2015n^2 + 4 &\geq n^{3.41} - 2015n^2 \\ &\geq 0.5n^{3.41} && \text{as long as } 0.5n^{3.41} \geq 2015n^2, \text{ i.e., } n \geq 4030^{1/1.41} \\ &\geq cn^3 && \text{as long as } n \geq (2c)^{1/0.41}. \end{aligned}$$

So, we can set $n_0 = \max\{4030^{1/1.41}, (2c)^{1/0.41}\}$, for example, and the definition of ω is fulfilled.

2. For each pair of functions $f(n)$ and $g(n)$, fill in the correct asymptotic notation among Θ, o, and ω in the statement $f(n) \in \sqcup(g(n))$. Formal proofs are not necessary, but provide brief justifications for all of your answers. (The default base in logarithms is 2.)

   (a) $f(n) = n^3(\log n)^2$ vs. $g(n) = n^2(\log n)^3$.
   Answer: ω. $f(n)/g(n) = n/\log n \to \infty$ as $n \to \infty$.

   (b) $f(n) = n^{341} + 2013^n$ vs. $g(n) = n^{240} + 2014^n$.
   Answer: o. $n^{341} \in o(2013^n)$ so $f(n) \in \Theta(2013^n)$. $n^{240} \in o(2014^n)$ so $g(n) \in \Theta(2014^n)$. Finally $2013^n/2014^n = (2013/2014)^n \to 0$ as $n \to \infty$ because $2013/2014 < 1$.

   (c) $f(n) = 3^{\log_9 n}$ vs. $g(n) = n^{1/4} + \sqrt{n} + \log n$.
   Answer: Θ. $f(n) = 3^{\log_9 n} = n^{\log_9 3} = n^{1/2}$. $g(n) = n^{1/4} + \sqrt{n} + \log n = n^{1/4} + n^{1/2} + \log n \in \Theta(n^{1/2})$ because $n^{1/4} \in o(n^{1/2})$ and $\log n \in o(n^{1/2})$.

   (d) $f(n) = (\log n)^{\log n}$ vs. $g(n) = n^2$.
   Answer: ω. $f(n) = 2^{\log n \log\log n}$ and $g(n) = 2^{2\log n}$. We have $\log n \log\log n \in \omega(2\log n)$ from which it follows that $\log n \log\log n - 2\log n \to \infty$. Then $f(n)/g(n) = 2^{\log n \log\log n - 2\log n} \to \infty$ as $n \to \infty$.

3. Analyze the following pseudocode and give a tight Θ bound on the running time as a function of n. Carefully show your work. (Here, "..." refers to some constant-time operations that do not change the values of i, j, k, and n.)

   (a)
   ```
   1.  for i = 1 to n do {
   2.      for j = 4n down to n do
   3.          ...
   4.      for j = 1 to i do
   5.          for k = 2 to j + 5 do
   6.              ...
       }
   ```

Answer: The complexity of the loop in 5–6 is $\Theta(j+4) = \Theta(j)$. The complexity of the loop in 4–6 is $\sum_{j=1}^{i} \Theta(j) = \Theta(\sum_{j=1}^{i} j) = \Theta(i^2)$. The complexity of the loop in 2–3 is $\Theta(3n) = \Theta(n)$. The complexity of the loop in 1–6 is

$$\sum_{i=1}^{n} (\Theta(n) + \Theta(i^2)) = \Theta\left(\sum_{i=1}^{n} n + \sum_{i=1}^{n} i^2\right) = \Theta(n^2 + n^3) = \Theta(n^3).$$

(b)
```
1.  k = 1
2.  for i = 1 to n do {
3.      for j = 1 to k + 5 do
4.          ...
5.      k = 3k
    }
```

Answer: The complexity of the loop in 3–4 is $\Theta(k+5) = \Theta(k)$. In the loop from 2–5, k takes on the values $1, 3, 3^2, \ldots, 3^{n-1}$. So the complexity is

$$\Theta(1) + \Theta(3) + \Theta(3^2) + \cdots + \Theta(3^{n-1}) = \Theta\left(\sum_{i=1}^{n-1} 3^i\right) = \Theta(3^n).$$

(c)
```
1.  for i = 1 to n do {
2.      j = n
3.      while j > 0 do {
4.          ...
5.          j = j − i^2
        }
    }
```

Answer: The complexity of the loop in 2–5 is $\Theta(n/i^2)$. The complexity of the loop in 1–5 is

$$\sum_{i=1}^{n} \Theta(n/i^2) = \Theta\left(\sum_{i=1}^{n} n/i^2\right) = \Theta\left(n \sum_{i=1}^{n} 1/i^2\right) = \Theta(n)$$

since $\sum_{i=1}^{\infty} 1/i^2 = \pi^2/6$ which implies $\sum_{i=1}^{n} 1/i^2 \in \Theta(1)$.

4. Consider the following problem: Given two sequences of numbers $\langle a_1, \ldots, a_n \rangle$ and $\langle b_1, \ldots, b_n \rangle$, we want to determine whether there exist i, j, k (with $0 \le i, j \le n - k$ and $1 \le k \le n$) such that $a_{i+1} + a_{i+2} + \cdots + a_{i+k} = b_{j+1} + b_{j+2} + \cdots + b_{j+k}$. (In other words, we want to determine whether there exist two blocks in the two sequences that have identical sums and identical lengths.)

   For example, for the input sequences $10, 21, 11, 12, 19, 15$ and $12, 9, 2, 31, 21, 8$, the answer is "yes" because $11 + 12 + 19 = 9 + 2 + 31$.

   (a) First design and analyze an algorithm that solves the problem in $O(n^3)$ time by "brute force".
   Answer:
   Idea: For fixed k, compute sums $A_i = a_{i+1} + a_{i+2} + \cdots + a_{i+k}$ for $0 \le i \le n - k$. Compute B_i similarly. Observe that for $i > 0$, $A_i = A_{i-1} + a_{i+k} - a_i$, such that we can efficiently compute A_i given A_{i-1}. Once we have $A_0, \ldots, A_{n-k}$ and $B_0, \ldots, B_{n-k}$ we can compare all $(n-k+1)^2 \in \Theta((n-k)^2)$ pairs of sums of length-k blocks.

```
1.  A_0 = 0
2.  B_0 = 0
3.  for k = 1 to n do
4.      A_0 = A_0 + a_k
5.      B_0 = B_0 + b_k
6.          for i = 1 to n − k do
7.              A_i = A_{i−1} + a_{i+k} − a_i
8.              B_i = B_{i−1} + b_{i+k} − b_i
9.          for 0 = 1 to n − k do
10.             for j = 0 to n − k do
11.                 if A_i = B_j then return "yes"
12. Return "no"
```

Cost analysis: Lines 1,2, 4,5 and 12 require $\Theta(1)$ time. Lines 6-8 takes $\Theta(n-k)$ time. Lines 9-11 take $\Theta((n-k)^2)$ time, thus lines 4-11 costs $\Theta((n-k)^2)$. Lines 3-10 entail $\sum_{k=1}^{n}\Theta((n-k)^2)$. Letting $\ell = n-k$, we have that this costs

$$\sum_{\ell=1}^{n}\Theta(\ell^2) = \Theta\left(\sum_{\ell=1}^{n}\ell^2\right) = \Theta(n^3),$$

and thus this approach costs $\Theta(n^3)$ total.

(b) Design and analyze a better algorithm that solves the problem in $O(n^2 \log n)$ time. [Hint: use sorting.]

[Note: As stated in the guidelines `https://www.student.cs.uwaterloo.ca/~cs341/policies.html`, in algorithm design questions, you should present pseudocode, describe the ideas behind your algorithm or justifications (if correctness is not obvious), and provide an analysis of the time complexity of your algorithm.]

Answer:

Idea: Compute the sums for fixed k as in 4(a). Instead of comparing each pair of sums as in 4(a), sort the sums $A_0, \ldots, A_{n-k}$ and $B_0, \ldots, B_{n-k}$ separately, and then compare by incrementally scanning through both lists of sorted sums.

```
1.  A_0 = 0
2.  B_0 = 0
3.  for k = 1 to n do
4.      A_0 = A_0 + a_k
5.      B_0 = B_0 + b_k
6.      for i = 1 to n − k do
7.          A_i = A_{i−1} + a_{i+k} − a_i
8.          B_i = B_{i−1} + b_{i+k} − b_i
9.      ⟨A'_0, A'_1, ..., A'_{n−k}⟩ = mergesort(A_0, ..., A_{n−k})
10.     ⟨B'_0, B'_1, ..., B'_{n−k}⟩ = mergesort(B_0, ..., B_{n−k})
11.     i, j = 0, 0
12.     while i ≤ n − k and j ≤ n − k do
13.         if A'_i = B'_j then Return "yes"
14.         else if A'_i > B'_j then j = j + 1
15.         else i = i + 1
16. Return "no"
```

Cost analysis: We give a worst-case analysis. In the worst-case, we do not find two blocks with identical sums, and the algorithm does not return early on line 13. In this case, as exactly one i or j is incremented in every iteration of the while loop on line 12, the while loop iterates between $n-k+1$ and $2(n-k+1)-1$ times, or $\Theta(n-k)$ times. Sorting on lines 9 and 10 cost $\Theta((n-k)\log(n-k))$ each. Lines 6-8 cost $\Theta(n-k)$. Lines 4,5, and 11 cost $\Theta(1)$. The the cost of one iteration of the for loop on line 3 costs $\Theta((n-k)\log(n-k))$. The total cost thus becomes $\sum_{k=1}^{n}\Theta((n-k)\log(n-k))$, which is upper-bounded by $O(n\cdot n\log n) = O(n^2\log n)$.

5. Consider the following problem: given a set of n points $\{(x_1,y_1),\ldots,(x_n,y_n)\}$ in 2D, we want to find a square with the smallest area that encloses all the given points. The square's sides must be parallel to the x- and y-axis.

   For example, for the input set $\{(1,4),(3,6),(2,7),(4,0),(5,5),(10,8),(9,2)\}$, one possible answer is the square with corners $(1,0),(10,0),(10,9),(1,9)$ and area 81.

   Design and analyze an algorithm that solves the problem using at most $3n-3$ comparisons.

   Answer:

   1. Find minimum and maximum of $x_1,\ldots,x_n$ by way of the optimal algorithm presented in class. Call them $x_{\min}$ and $x_{\max}$.
   2. Find $y_{\min}$ and $y_{\max}$ similarly.
   3. $\delta_x = x_{\max} - x_{\min}$
   4. $\delta_y = y_{\max} - y_{\min}$
   4. if $\delta_x > \delta_y$
   5. Return $\langle(x_{\min},y_{\min}),(x_{\min}+\delta_x,y_{\min}),(x_{\min}+\delta_x,y_{\min}+\delta_x),(x_{\min},y_{\min}+\delta_x)\rangle$
   6. else
   7. Return $\langle(x_{\min},y_{\min}),(x_{\min}+\delta_y,y_{\min}),(x_{\min}+\delta_y,y_{\min}+\delta_y),(x_{\min},y_{\min}+\delta_y)\rangle$

   Analysis: Lines 1 and 2 require exactly $3n/2-2$ comparisons each. Line 4 requires an additional comparison, giving a total of $3n-3$ comparisons.

6. Give tight asymptotic (Θ) bounds for the solution to the following recurrences by using the recursion-tree method (you may assume that n is a power of 10 in (a), or a power of 3/2 in (b)). Show your work.

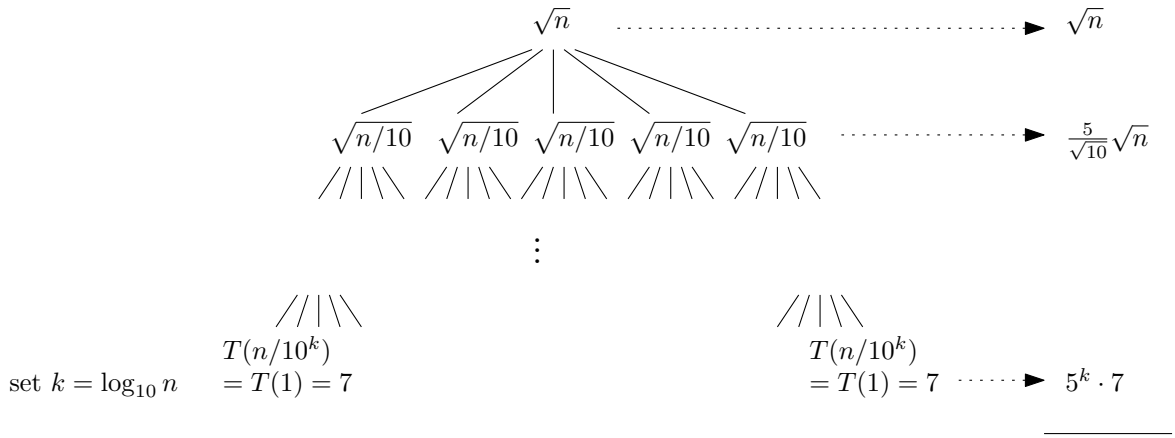
   (a)

$$T(n) = \begin{cases} 5\,T(n/10) + \sqrt{n} & \text{if } n > 1 \\ 7 & \text{if } n \le 1 \end{cases}$$

   Answer:

   Total:

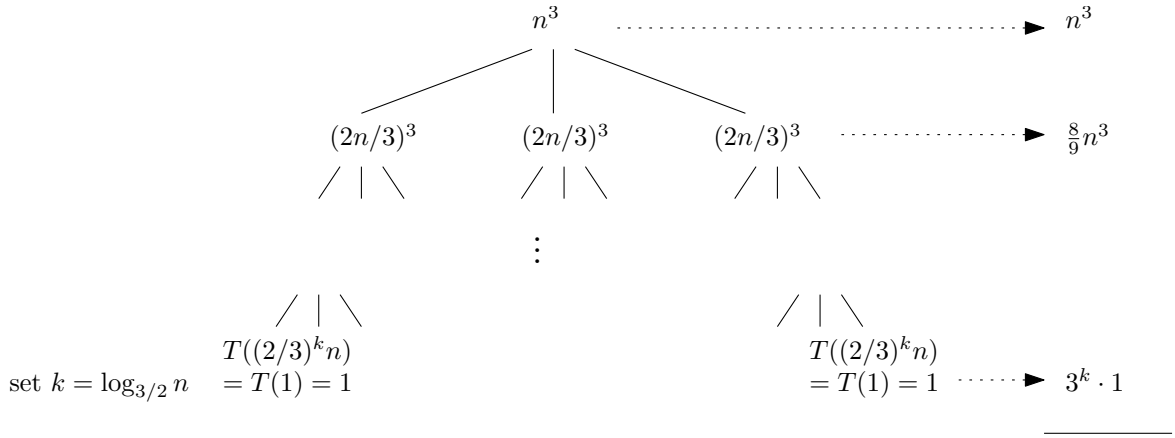
$$\begin{aligned}
\sqrt{n}\sum_{i=0}^{k-1}\left(\frac{5}{\sqrt{10}}\right)^i + 5^k\cdot 7 &= \Theta\left(\sqrt{n}\left(\frac{5}{\sqrt{10}}\right)^k + 5^k\right) \\
&= \Theta\left(\sqrt{n}\left(\frac{5}{\sqrt{10}}\right)^k + 5^k\right) \\
&= \Theta\left(\sqrt{n}\left(\frac{5}{\sqrt{10}}\right)^{\log_{10} n} + 5^{\log_{10} n}\right) \\
&= \Theta(\sqrt{n}\cdot n^{\log_{10}(5/\sqrt{10})} + n^{\log_{10} 5}) \\
&= \Theta(n^{1/2+\log_{10} 5-1/2} + n^{\log_{10} 5}) \\
&= \Theta(n^{\log_{10} 5}).
\end{aligned}$$



(b)

$$T(n) = \begin{cases} 3\,T(2n/3) + n^3 & \text{if } n > 1 \\ 1 & \text{if } n \le 1 \end{cases}$$

Answer:



Total:

$$\begin{aligned} n^3 \sum_{i=0}^{k-1} (8/9)^i \;+\; 3^k \cdot 1 &= \Theta(n^3 + 3^k) \\ &= \Theta(n^3 + 3^{\log_{3/2} n}) \\ &= \Theta(n^3 + n^{\log_{3/2} 3}) \\ &= \Theta(n^3 + n^{2.70951\cdots}) \\ &= \Theta(n^3). \end{aligned}$$

7. (a) Solve part (a) of the previous question by the master method.
Answer: $a = 5$, $b = 10$, $f(n) = \sqrt{n}$, $d = \log_{10} 5 = 0.69897\cdots$, $\varepsilon = 0.001$.
Since $f(n) = n^{0.5} = O(n^{d-\varepsilon})$, we are in Case 1 and $T(n) = \Theta(n^d) = \Theta(n^{\log_{10} 5})$.

(b) Solve part (b) of the previous question by the master method.

Answer: $a = 3$, $b = 3/2$, $f(n) = n^3$, $d = \log_{3/2} 3 = 2.70951\cdots$, $\varepsilon = 0.001$.

Since $f(n)/n^{d+\varepsilon} = n^{0.289\cdots}$ is an increasing function, we are in Case 3 and $T(n) = \Theta(f(n)) = \Theta(n^3)$.