1. (a) Consider $c := 12 + 11 + 10 = 33$ and $n_0 := 1$. Then $0 \le 12n^3 + 11n^2 + 10 \le cn^3$ for all $n \ge n_0$.

   (b) Consider $c := 1$ and $n_0 := 1$. Then $0 \le cn^3 \le 12n^3 + 11n^2 + 10$ for all $n \ge n_0$.

   (c) Follows from parts (a) and (b), i.e., let $c_1 := 33$, $c_2 := 1$ and $n_0 := 1$.

   (d) Let $c > 0$ be given. Set $n_0 > 0$ to be minimal such that $1000 < c \log n_0$ (i.e., $n_0 := 1 + \lfloor e^{1000/c} \rfloor$). Then $0 \le 1000n < cn \log n$ for all $n \ge n_0$.

   (e) Let $c > 0$ be given. Set $n_0 := 21 + c$. Then, for $n \ge n_0$, we have $n^n = n^{n-20}n^{20} \ge (21+c)^{1+c}n^{20}$. Since $(21+c)^{1+c} > c$, this shows that $0 \le cn^{20} < n^n$ for all $n \ge n_0$.

2. (a) $f(n) \in \omega(g(n))$. Either show directly that the definition of $\omega$, or show that $\lim_{n\to\infty} f(n)/g(n) = \infty$.

   (b) $f(n) \in \Theta(g(n))$. It suffices to show that both $f(n)$ and $g(n)$ are $\Theta(n^3)$.

3. (a) False. Counter example: Consider $f(n) := n$ and $g(n) := \begin{cases} 1 & n \text{ odd} \\ n^2 & n \text{ even} \end{cases}$. To prove the claim false it will be sufficient to show that $f(n) \notin O(g(n))$ and $f(n) \notin \Omega(g(n))$, since then the antecedent of the implication is satisfied while the consequent is not.

   If $f(n) \in O(g(n))$, then there exist constants $n_0 > 0$ and $c > 0$ such that $f(n) \le cg(n)$ for all $n \ge n_0$. But for any odd number $n_1 > c$ we have $f(n_1) = n_1 > c = cg(n_1)$, showing that $f(n) \notin O(g(n))$.

   Similarly, if $f(n) \in \Omega(g(n))$, then there exists constants $n_0 > 0$ and $c > 0$ such that $cg(n) \le f(n)$ for all $n \ge n_0$. But for any even number $n_1 > 1/c$ we have $cg(n_1) = cn_1^2 > n_1 = f(n_1)$, showing that $f(n) \notin \Omega(g(n))$.

   (b) True. We will show that $f(n)g(n)/(f(n)+g(n)) \le \min(f(n), g(n)) \le 2f(n)g(n)/(f(n)+g(n))$ for all $n \ge 1$. The desired result will then follow from the definition of $\Theta$ using $c_1 = 1$, $c_2 = 2$ and $n_0 = 1$.

   For brevity, let $f$ denote $f(n)$ and $g$ denote $g(n)$, $n \ge n_0$. By assumption, $f$ and $g$ are positive, so $fg/(f+g) = \min(f,g)\max(f,g)/(f+g)$, which is less than $\min(f,g)$ since $\max(f,g)/(f+g) < 1$. Similarly, $\min(f,g) = 2fg/(2\max(f,g)) \le 2fg/(f+g)$.

4. We have $S(n) = 2\,S(n) - S(n) = 1 + (\sum_{i=1}^{n-1}(1/2^i)) - n/2^n = 2 - (n+2)/2^n$.

5. (a) $2^n$, since each entry can be 0 or 1

   (b) $n+1$, when all entries of $v[1 \ldots n]$ are zero

   (c) $S_i$ is the set of boolean vectors of length $n$ for which the first $i-1$ entries are 0 and entry $i$ is 1. We have $|S_i| = 2^{n-i}$ since the first $i$ entries of a vector in $S_i$ are fixed while the last $n-i$ entries are chosen arbitrarily.



j43luo

This document is for the exclusive use of j43luo.

(d) The average case calls to print is

$$\frac{1}{2^n}\left((n+1)+\sum_{i=1}^{n} i2^{n-i}\right) = \frac{n+1}{2^n} + S(n) = 2 - \frac{1}{2^n}.$$

6. To arrive at a contradiction, assume there exists a value of $n$ for which the code fragment will not terminate. For $i \geq 0$, let $s_i$ be the value of $s$ after the $i$'th iteration of the loop. Then each $s_i$ is positive. Observe that during two consecutive iterations of the loop body, the first branch of the if statement will be executed at least once. Then $s_{i+2} \leq \max(\lfloor \lfloor s_i/4 \rfloor /4 \rfloor, \lfloor (2s_i)/4 \rfloor, 2\lfloor s_i/4 \rfloor) \leq \lfloor s_i/2 \rfloor \leq s_i - 1$. Thus $s_0, s_2, s_4, \ldots$ is an infinite, monotonically decreasing sequence of positive integers, a contradiction.

7. We count the number of times that line 6 is executed:

$$\begin{aligned}
\sum_{i=1}^{3n} \sum_{j=1388}^{2010} \sum_{k=4i}^{6i} 1 &= \sum_{i=1}^{3n} \sum_{j=1388}^{2010} (2i+1) \\
&= \sum_{i=1}^{3n} 623(2i+1) \\
&= 1246 \sum_{i=1}^{3n} i + 623 \sum_{i=1}^{3n} 1 = 623(3n)(3n+1) + 1869n \\
&\in \Theta(n^2)
\end{aligned}$$

Since line 6 contains only a single primitive operation and takes constant time, the total cost for line 6 is $\Theta(n^2)$. The primitive operation in line 1 is executed one time and the one in line 3 is executed $3n$ times. The total running time is thus $\Theta(n^2)$.

8. Consider the heap as a binary tree. The algorithm is based on the following property of a max-heap. For any node $v$, if the key stored in $v$ is strictly less than $c$, then all keys in the left and right subtrees of $v$ are also strictly less than $c$.

*Algorithm:* Check if the key at the root is less than $c$ and if so exit without reporting any integers. If the key at the root is at least $c$ then report it and make recursive calls on the left and right sub-heaps (sub-trees).

*Running time:* All nodes that are visited that contain a key that is strictly less than $c$ have a parent node that contains a key that is greater than or equal to $c$. Thus, the number of nodes visited that contain a key that is strictly less than $c$ is at most $2k$. The total number of nodes visited is therefore at most $3k$. This shows that the running time is $O(k)$.



j43luo

**This document is for the exclusive use of j43luo.**