1. a) Note that \((\log n)^{\log \log n} = (2^{\log \log n})^{\log \log n} = 2^{(\log \log n)^2}\). Also, \((\log \log n)^{\sqrt{\log n}} = 2^{\log \frac{1}{2} \sqrt{\log n}}\). From the list of functions given in class, \((\log \log n)^2 = o(\sqrt{\log n})\). Thus, \((\log n)^{\log \log n} = O((\log \log n)^{\sqrt{\log n}})\). The given statement is true.

b) \((2.01)^n = 2^{\log 2.01 \cdot n} = \Omega(2^{1.007n})\). From the list of functions given in class, \(n^{10} = o(2^{0.007n})\). Thus, \(n^{10.2n} = O((2.01)^n)\). The given statement is true.

2. Run time of Test is \(\sum_{i=1}^{3n} \sum_{j=1}^{5n} 7j^2 + 5j = 7 \sum_{i=1}^{3n} \sum_{j=1}^{5n} j^2 + 5 \sum_{i=1}^{3n} \sum_{j=1}^{5n} j = 7 \sum_{i=1}^{3n} \sum_{j=1}^{5n} \frac{5n(5n+1)(10n+1)}{6} + 5 \sum_{i=1}^{3n} \sum_{j=1}^{5n} (5n+1) = \Theta(n^4)\).

3. Consider the following algorithm:

   for \(i := 1\) to \(\alpha \log n\) do
   
   Pick a random \(j \in [1, n]\). If \(a[j] = 1\) then output “Type II” and quit;

   Output: "Type I";

   Analysis: Note that if the array is of type I, the above algorithm will never give an incorrect answer. Thus assume that the array is of type II. We’ll calculate the probability of an incorrect answer as follows.

   The output will be incorrect if all of the random elements picked are zeros. The probability of this happening is \(\leq \left(\frac{1}{2}\right)^{\alpha \log n} = n^{-\alpha}\).

   Thus the output of this algorithm is correct with high probability.

4. Keep a 2-3 tree and a variable Increment. To begin with the value of Increment is zero. Here are the algorithms for the four operations:

   **INSERT**(*x*): insert an element with a key value of \((x - \text{Increment})\) into the 2-3 tree.

   **SEARCH**(*x*): search for the element whose key is \((x - \text{Increment})\) in the 2-3 tree;

   **DELETE**(*x*): delete the element with the key \((x - \text{Increment})\) from the 2-3 tree;

   **ADD_ALL**(*y*): increment the value of Increment by *y*.

   Clearly, each of the above algorithms takes \(O(\log n)\) time.
5. Recurrence relation for the run time of \(A\) is: \(T(n) = 27T(n/3) + n^3\). Here \(a = 27, b = 3, f(n) = n^3\). \(n^{\log_b a} = n^3\). Case 2 of Master theorem applies. Thus, \(T(n) = \Theta(n^3 \log n)\).

Recurrence relation for the run time of \(B\) is: \(T(n) = 225T(n/15) + n^{2.5}\). Here \(a = 225, b = 15, f(n) = n^{2.5}\). \(n^{\log_b a} = n^2\). Case 3 of Master theorem applies implying that \(T(n) = \Theta(n^{2.5})\).

Therefore, algorithm \(A\) is preferable.

6. Note that if \(A\) and \(B\) are two sorted sets then we can compute \(A \cap B\) in \(O(|A| + |B|)\) time by merging the two sorted sequences. Given the sets \(A_1, A_2, \ldots, A_m\), start by intersecting \(A_1\) with \(A_2\) to get \(B_1\); \(A_3\) with \(A_4\) to get \(B_2\); \(\cdots\); \(A_{m-1}\) with \(A_m\) to get \(B_{m/2}\).

This will take a total time of \(O \left( \sum_{i=1}^{m/2} |A_{2i-1}| + |A_{2i}| \right)\). This in turn is no more than \(cn\) for some constant \(c\). Realize that \(|B_1| \leq \min\{|A_1|, |A_2|\}; |B_2| \leq \min\{|A_3|, |A_4|\}; \) and so on. Next, intersect \(B_1\) with \(B_2\) to get \(C_1\); \(B_3\) with \(B_4\) to get \(C_2\); and so on. These intersections will take time no more than \(c \left( \sum_{i=1}^{m/2} |B_i| \right)\). However, \(\sum_{i=1}^{m/2} |B_i| \leq \frac{n}{2}\). Thus the second stage of intersections will take time no more than \(cn^2\).

Continue the above stages of intersections until we are left with only one set. The time taken in any stage is no more than one half of the time spent in the previous stage. Therefore the run time of the entire algorithm is \(\leq c \left[ n + \frac{n}{2} + \frac{n}{4} + \cdots \right] \leq 2cn = O(n)\).