Home Work 0 (CS6150)

Aishwarya Asesh (u1063384)

September 15, 2016

Answer 1: Big Oh Notations
(a) $\mathcal{O}(n^2)$
(b) $\mathcal{O}(\log n)$
(c) $\mathcal{O}(1/n)$
(d) $\mathcal{O}(1)$
(e) The sum of given series can be written in general term as $\int_1^n \frac{1}{x} dx$ which equals to $\log n$. So we can say that the complexity for this series is $\mathcal{O}(\log n)$.

Answer 2: Removing Duplicates
Algorithm: The first step will be sorting of array A using mergesort. Let array C be the sorted array. Now scan array C, if nth element is not same as (n-1)th element of array C store the value in array B.
Correctness: 100% correctness is obtaind as array C now has all the duplicate values and array B contains all distinct elements.
Running Time: Quicksort takes $\mathcal{O}(n \log n)$ and scanning array C for duplicates takes $\log n$. Thus the overall complexity for the whole process is $\mathcal{O}(n \log n)$.

Answer 3: Square vs Multiply
Let A be an algorithm that can square any n digit number in time $\mathcal{O}(\log n)$. Then the n digit square can be implied as $a.a = a^2$. If we consider the multiplication of 2 different numbers a,b in such a way that $ab = \frac{(a+b)^2 - a^2 - b^2}{2}$, the overall time will be $\mathcal{O}(n \log n)$ as the process demands three calls to function A (complexity $\mathcal{O}(n \log n)$) and time for division $(\mathcal{O}(1) or \mathcal{O}(n))$.

Answer 4: Probability
(a) The outcome of the toss event can be written as (H,T). If the coin is tossed k times then total possibilities of outcomes are 2^k. The events that contain exactly one head are HTT....,THTTT.., and thus k times of them. Thus the probability is $\frac{k}{2^k}$.
(b) As there are k boxes and k different colors. The total sample space of coloring each box is k^k. For each box to have unique colors we have to ensure that after coloring a particular box the other box does not have the same color, that implies k,(k-1),(k-2)....1. Thus the probability of coloring each box uniquely is $\frac{k!}{k^k}$.

Answer 5: Sum of Array
Algorithm: Sorting of elements in A can be done using merge sort with time complexity $\mathcal{O}(n \log n)$. Let B contain the elements of newly sorted array. For each pair of j and k, compute $A[j] + A[k]$. Check if array B contains the previous generated sum. Output YES if the result is found in array B, else Output NO.
Correctness: As array B contains all the elements of array A in sorted order, for each true case of $A[i] = A[j] + A[k]$, the result must be present in array B. Thus we can expect 100% correctness of algorithm.
Running Time: Merge Sort takes $\mathcal{O}(n \log n)$ time complexity. Then for performing binary search for n^2, each takes $(n \log n)$. Thus the complexity equation is $\mathcal{O}(n^2 \log n) + \mathcal{O}(n \log n)$. Final time complexity is $\mathcal{O}(n^2 \log n)$