Practice 4: The Power of Randomness
Type: Group Assignment
Due: Feb 3

`Randomness:` Sakthi kitchen's cook prepares 70kg rice every day to cater 1000 students. How does a cook check whether rice is cooked fully. Is he not applying some randomness; yes, he indeed checks couple of rice samples and declares that the entire rice is cooked. On very few occasions, his strategy fails. That is, he declares that the rice is cooked fully, however, it is partially cooked. Can we attempt a similar business in computing science.

1. Linear search religiously checks each cell of an array to say $x \in A$ or not. Can we attempt the following: randomly choose $\log n$ positions and check if x is present. Does this approach work always ? Efficacy analysis focuses on the quality of the solution and give a guarantee on the performance of the algorithm. For example, with what probability the algorithm is right. **Tasks:** (i) Generate 100 samples (integer array) of size 2^{16} each. Also, x to be searched is random. (ii) x to be searched is present in the array; choose a random location and assign the value present in it to x (iii) For the same input set, choose $\sqrt{n}$ locations randomly and perform the search (iv) compare the performance of linear search, randomized search type 1 and randomized search type 2 to assess how good these randomized algorithms are.

2. Randomized Sorting - Can you think of some strategy for randomized sorting. Note that any randomized algorithm need not output a sorted sequence. Efficacy analysis will tell us, on 100 input samples, how many of them are actually sorted by your strategy.

3. Why should we go for a randomized approach ? If we are successful, then the first approach takes $\theta(\log n)$ which is $o(n)$, way better than the linear search. The goal of randomized algorithm is to look for $o(n \log n)$ algorithm for sorting and $o(n)$ for searching with a good performance guarantee.

4. Can you think of some randomized strategy to verify Ramsey's magic. Bring in randomization only in the verification part (whether a graph on 7 vertices contains a K_3 or $\bar{K_3}$)

5. Given an integer array A and integer k, the objective is to find k largest elements and k smallest elements in A. `Constraints:` NO sorting. NO simple scan of A, k times. Can we do it in $O(n)$ time. Will randomness help us in some way?

6. `Randomized Sorting:` Let A be an array with 100 elements. Choose 5 locations of A randomly and swap the contents of it with contents in locations $A[51]$ to $A[55]$. Sort $A[51] \ldots A[55]$ in non-decreasing order. (i) Think of a strategy using which we can re-arrange A in such a way that $A[1], \ldots, A[50]$ contain values which are at most $A[51]$. Similarly, the values of $A[56], \ldots, A[100]$ are at least $A[55]$. Note, there is no order among the elements of $A[1..50]$ and $A[56..100]$. Can we perform a simple scan incurring $O(n)$ effort to accomplish this task.
   (ii) Now, one can apply Merge sort or Quick sort on $A[1..50]$ and $A[56..100]$ to get a sorted sequence. (iii) Instead, can we repeat the above task in $A[1..50]$ and $A[56..100]$ to get a sorted array A. `Constraints:` NO additional space to answer (i). By design, merge sort uses additional space, which is inevitable.
   Merge and Quick sort shall be discussed next week.