1. **Preliminaries.** We say \( f(n) = O(g(n)) \) if \( f(n) \leq cg(n) \) for all \( n \geq n_0 \) for some constants \( c \) and \( n_0 \). We say \( f(n) = \Omega(g(n)) \) if and only if \( g(n) = O(f(n)) \). Also, \( f(n) = \Theta(g(n)) \) if \( f(n) = O(g(n)) \) and \( f(n) = \Omega(g(n)) \).

A partial list of functions in increasing order is: \( O(1) \), \( (\log n)^c \), \( \log n \), \( (\log n)^{1+\mu} \), \( n^c \), \( n \), \( n^{1+\mu} \), \( 2^n \), \( 2^{n^{1+\mu}} \) where \( 0 < c < 1 \) and \( \mu > 0 \) are constants.

Stirling’s approximation: \( n! \approx \left(n/e\right)^n \sqrt{2\pi n} \).

\[ \sum_{i=1}^{n} i = n(n+1)/2. \quad \sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6. \quad \sum_{i=1}^{n} i^3 = n^2(n+1)^2/4. \]

2. **Master theorem.** Consider the recurrence relation: \( T(n) = aT(n/b) + f(n) \), where \( a \geq 1 \) and \( b > 1 \) are constants. **Case 1:** If \( f(n) = O(n^{log_b a-\epsilon}) \) for some constant \( \epsilon > 0 \), then \( T(n) = \Theta(n^{log_b a}) \). **Case 2:** If \( n^{log_b a} = \Theta(f(n)) \), then \( T(n) = \Theta(f(n) \log n) \). **Case 3:** If \( f(n) = \Omega(n^{log_b a+\epsilon}) \) for some constant \( \epsilon > 0 \) and \( af(n/b) \leq cf(n) \) for some constant \( c < 1 \), then \( T(n) = \Theta(f(n)) \).

3. **Randomized algorithms.** A Monte Carlo algorithm runs for a prespecified amount of time and its output is correct with high probability. By high probability we mean a probability of \( \geq (1 - n^{-\alpha}) \), for any constant \( \alpha \). A Las Vegas algorithm always outputs the correct answer and its run time is a random variable. We say the run time of a Las Vegas algorithm is \( O(f(n)) \) if the run time is \( \leq cf(n) \) for all \( n \geq n_0 \) with probability \( \geq (1 - n^{-\alpha}) \), for some constants \( c \) and \( n_0 \).

**Chernoff bounds:** If \( X \) has a binomial distribution \( B(n, p) \), then \( Pr[X \geq (1 + \epsilon)np] \leq \exp(-\epsilon^2 np/3) \) and \( Pr[X \leq (1 - \epsilon)np] \leq \exp(-\epsilon^2 np/2) \), for any \( 0 < \epsilon < 1 \).

4. **Data Structures.** A dictionary supports three operations: INSERT, DELETE, and SEARCH. A (min) priority queue supports: INSERT, FindMin and DeleteMin. If one uses a 2-3 tree, each of these operations takes \( O(\log n) \) time, \( n \) being the number of elements in the data structure.

In the Union-Find paradigm we start with \( n \) sets: \( \{1\}, \{2\}, \ldots, \{n\} \). The goal is to perform a sequence of union and find operations. An arbitrary sequence of \( m \) union-find operations can be performed in time \( O(m \alpha(m)) \) where \( \alpha \) is the inverse Ackermann’s function.

5. **Sorting.** Given a sequence of \( n \) numbers (or keys), the problem of sorting is to rearrange this sequence in either nondecreasing order or nonincreasing order. The mergesort algorithm has a worst case run time of \( O(n \log n) \). The run time of quicksort is \( O(n^2) \) in the worst case and \( O(n \log n) \) on the average.

We proved a sampling lemma: If \( s \) keys are picked randomly from a sequence \( X \) of \( n \) keys and if the \( s \) keys are sorted and used to partition \( X \) into \( s + 1 \) parts, then the size of each part is \( \tilde{O} \left( \frac{n^2}{s} \log n \right) \). Frazer-McKellar’s randomized algorithm does sorting using \( n \log n + \tilde{O}(n \log \log n) \) comparisons.

A sorting problem is called general sorting if the only information known about the keys is that the keys are from a linear order. Any general sorting algorithm makes use of comparison as the basic operation. We have shown that any general sorting algorithm needs \( \log(n!) = \Omega(n \log n) \) comparisons in the worst case.

We can sort \( n \) keys in \( O(n) \) time if the keys are integers in the range \([1, n^c]\) (for any constant \( c \)).