Quick Sort: This algorithm sorts \( n \) given elements in an expected \( O(n \log n) \) time. The worst case run time could be \( \Omega(n^2) \). We also described a randomized sorting algorithm due to Frazer and McKellar. This is one of the best known sorting algorithms.

**SELECTION.** Given a sequence of \( n \) keys and an integer \( i \) with \( 1 \leq i \leq n \), the problem of selection is to identify the \( i \)th smallest element from out of the \( n \) keys. The BFPRT algorithm solves this problem in \( O(n) \) time. We also summarized a randomized selection algorithm that takes \( \tilde{O}(n) \) time.

**BUCKET and RADIX SORTING.** If \( X \) is a sequence of \( n \) keys where each key is an integer in the range \([1, m]\) then \( X \) can be sorted in \( O(m + n) \) time. If \( m = n^c \) for any constant \( c \), then radix sort can be used to sort \( X \) in \( O(n) \) time.

**MATRIX MULTIPLICATION.** Strassen’s algorithm multiplies two \( n \times n \) matrices in \( O(n \log_2 7) \) time.

**GREEDY ALGORITHMS.** This technique is used when we are interested in finding a subset of \( n \) given objects that satisfies a set of constraints and optimizes a given objective function. The general idea is to start with the empty set; select the next object \( O \) to be examined using a selection criterion; if the inclusion of \( O \) into the solution \( S \) will still keep it feasible we add \( O \) into \( S \), otherwise we discard \( O \); proceed in a similar fashion until all the objects have been examined; and finally output the solution \( S \).

We were able to solve the fractional knapsack problem in \( O(n \log n) \) time using the greedy approach. The idea is to process the objects in nonincreasing order of their profit densities.

We also showed that the minimum weight spanning tree problem can be solved in \( O((|V| + |E|) \log |V|) \) time on any weighted undirected graph \( G(V, E) \) employing the greedy technique. Prim’s algorithm has only one tree at any time. It looks at all the outgoing edges from the tree and includes the edge with the minimum weight. Kruskal’s algorithm starts with a forest of \( n \) trees and inserts one edge at a time into the forest (if the edge does not cause a cycle). The edges are sorted in nondecreasing order of the edge weights to begin with.

Dijkstra’s algorithm for the single source shortest path problem runs in \( O((|V| + |E|) \log |V|) \) time. This algorithm assumes that the input graph does not have any edges with negative weights.