**A brief survey of sorting applications.**

* ***Commercial computing.*** Government organizations, financial institutions, and commercial enterprises organize much of this information by sorting it. Whether the information is accounts to be sorted by name or number, transactions to be sorted by time or place, mail to be sorted by postal code or address, files to be sorted by name or date, or whatever, processing such data is sure to involve a sorting algorithm somewhere along the way.
* ***Search for information.*** Keeping data in sorted order makes it possible to efficiently search through it using the classic *binary search* algorithm.
* ***Operations research****.* Suppose that we have N jobs to complete, where job j requires tj seconds of processing time. We need to complete all of the jobs, but want to maximize customer satisfaction by minimizing the average completion time of the jobs. The *shortest processing time first* rule, where we schedule jobs in increasing order of processing time, is known to accomplish this goal. As another example, consider the *load-balancing problem*, where we have M identical processors and N jobs to complete, and our goal is to schedule all of the jobs on the processors so that the time when the last job completes is as early as possible. This specific problem is NP-hard (see Chapter 6) so we do not expect to find a practical way to compute an optimal schedule. One method that is known to produce a good schedule is the *longest processing time first* rule, where we consider the jobs in decreasing order of processing time, assigning each job to the processor that becomes available first.
* ***Event-driven simulation.*** Many scientific applications involve simulation, where the point of the computation is to model some aspect of the real world in order to be able to better understand it. Doing such simulations efficiently can require appropriate algorithms and data structures. We consider a particle-collision simulation in [Section 6.1](http://algs4.cs.princeton.edu/61event/) that illustrates this point.
* ***Numerical computations****.* Scientific computing is often concerned with accuracy (how close are we to the true answer?). Accuracy is extremely important when we are performing millions of computations with estimated values such as the floating-point representation of real numbers that we commonly use on computers. Some numerical algorithms use priority queues and sorting to control accuracy in calculations.
* ***Combinatorial search.*** A classic paradigm in artificial intelligence is to define a set of *configurations* with well-defined moves from one configuration to the next and a priority associated with each move. Also defined is a *start* configuration and a *goal* configuration (which corresponds to having solved the problem. The *A\* algorithm* is a problem-solving process where we put the start configuration on the priority queue, then do the following until reaching the goal: remove the highest-priority configuration and add to the queue all configurations that can be reached from that with one move (excluding the one just removed).
* ***Prim's algorithm and Dijkstra's algorithm*** are classical algorithms that process graphs. Priority queues play a fundamental role in organizing graph searches, enabling efficient algorithms.
* ***Kruskal's algorithm*** is another classic algorithm for graphs whose edges have weights that depends upon processing the edges in order of their weight. Its running time is dominated by the cost of the sort.
* ***Huffman compression*** is a classic data compression algorithm that depends upon processing a set of items with integer weights by combining the two smallest to produce a new one whose weight is the sum of its two constituents. Implementing this operation is immediate, using a priority queue.
* ***String processing* algorithms** are often based on sorting. For example, we will discuss algorithms for finding the longest common prefix among a set of strings and the longest repeated substring in a given string that are based on first sorting suffixes the strings.