Sets and strings both represent collections of objects—the difference is whether order matters. Sets are collections of symbols whose order is assumed to carry no significance, while strings are defined by the sequence or arrangement of symbols.

The assumption of a fixed order makes it possible to solve string problems much more efficiently than set problems, through techniques such as dynamic programming and advanced data structures like suffix trees. The interest in and importance of string-processing algorithms have been increasing due to bioinformatics, Web searches, and other text-processing applications. Recent books on string algorithms include:

- *Gusfield* [Gus97] – To my taste, this remains is the best introduction to string algorithms. It contains a thorough discussion on suffix trees, with clear and innovative formulations of classical exact string-matching algorithms.

- *Crochemore, Hancart, and Lecroq* [CHL07] – A comprehensive treatment of string algorithms, written by a true leader in the field. Translated from the French, but clear and accessible.

- *Navarro and Raffinot* [NR07] – A concise but practical and implementation-oriented treatment of pattern-matching algorithms, with particularly thorough treatment of bit-parallel approaches.

- *Crochemore and Rytter* [CR03] – A survey of specialized topics in string algorithmics emphasizing theory.

Theoreticians working in string algorithmics sometimes refer to their field as *Stringology*. The annual *Combinatorial Pattern Matching* (CPM) conference is the primary venue devoted to both practical and theoretical aspects of string algorithmics and related areas.
18.1 Set Cover

**Input description:** A collection of subsets $S = \{S_1, \ldots, S_m\}$ of the universal set $U = \{1, \ldots, n\}$.

**Problem description:** What is the smallest subset $T$ of $S$ whose union equals the universal set—i.e., $\cup_{i=1}^{\lvert T \rvert} T_i = U$?

**Discussion:** Set cover arises when you try to efficiently acquire items that have been packaged in a fixed set of lots. You seek a collection of at least one of each distinct type of item, while buying as few lots as possible. Finding a set cover is easy, because you can always buy one of each possible lot. However, identifying a small set cover let you do the same job for less money. Set cover provided a natural formulation of the Lotto ticket optimization problem discussed in Section 1.6 (page 23). There we seek to buy the smallest number of tickets needed to cover all of a given set of combinations.

Boolean logic minimization is another interesting application of set cover. We are given a particular Boolean function of $k$ variables, which describes whether the desired output is 0 or 1 for each of the $2^k$ possible input vectors. We seek the simplest circuit that exactly implements this function. One approach is to find a disjunctive normal form (DNF) formula on the variables and their complements, such as $x_1 \bar{x}_2 + \bar{x}_1 \bar{x}_2$. We could build one “and” term for each input vector and then “or” them all together, but we might save considerably by factoring out common subsets of variables. Given a set of feasible “and” terms, each of which covers a