Chapter XI
Strategies for Static Tables

Dean Kelley
Minnesota State University, Mankato, USA

ABSTRACT

This chapter presents three alternatives for structuring static tables—those tables in which the collection of keys remains unchanged and in which the FIND operation is optimized. Each alternative provides performance guarantees for the FIND operation which can help those who design and/or implement systems achieve performance guarantees of their own. The chapter provides clear and concise algorithms for construction and/or usage and simple guidelines for choosing among the strategies. It is intended that this presentation will help inform system design decisions. It is further intended that this chapter will assist implementation activities for systems which make use of static tables.

INTRODUCTION

System designers and implementers are frequently required to guarantee system performance. Often, performance depends on efficient interaction with tables and dictionaries of data. When table operations require communication via resources whose availability may not be guaranteed (e.g., networked communication with a database), designers/implementers may not be able to make firm guarantees of system or system component performance. Consequently, it may be necessary to incorporate tables into a system rather than rely on external support for them.

Generally, tables fall into two categories depending on the type of operations they are required to support. *Static* tables are ones which, once built, do not change. The set of keys that they contain remains unchanged throughout their lifetime (although the data associated with the keys may change). A *dynamic* table may change its set of keys by means of insertion and deletion during the time that the table is active. Both types of table support searching to determine if a specific key is present as well as to retrieve data associated with a key.

This chapter focuses on techniques for implementing static tables. The first section presents necessary technical background. The second section presents a general technique for situations in which keys can be totally ordered and searching can be accomplished by key comparisons. The third section extends that technique to a situation in which the probabilities for search operations for each key are known. The fourth section presents

a technique in which, at most, a single key comparison is required and that comparison is only for equality, consequently the keys do not need to be totally ordered.

The topics of this chapter provide system designers and implementers with alternatives for static tables which can yield firm guarantees of performance.

BACKGROUND

This chapter is concerned with *static* tables, tables in which data are associated with keys and the set of keys remains unchanged throughout the life of the table. Access to a specific item of data is accomplished by searching for its key by means of a FIND operation.

Table techniques are evaluated by their consumption of time and space. Time is measured by the number of key comparisons which take place during a FIND operation. Space is measured by the size of the structure(s) necessary to contain the keys and additional supporting data necessary to make the technique work. We do not include the "user" data associated with the keys in the space cost.

Key comparisons may be for equality ("k_1 *equals* k_2") or order ("k_1 *is before* k_2"). The search techniques of the second and third sections require comparisons for order. Consequently, the comparison of any two keys in the set of keys must yield information about their relative order. When a relation equivalent to "$\leq$" holds for all elements of a set, that set is said to be *totally ordered*. Thus, the techniques of the second and third sections require that the keys be from a totally ordered set.

The study of efficient techniques for static tables owes much to the paper by Yao (1981). In this paper it was shown that if the number of keys in the table is small relative to the number of possible keys, then the number of comparisons necessary to determine if a particular key is present (and where it is) is $\Omega(\log n)$. That is, it requires at least a constant times $\log n$ comparisons. Consequently, binary search in a sorted array (the topic of the second section) is the best approach under these circumstances.

Alternatives to array-based tables lead to tree-like structures built with links. In the third section, we present a specific type of binary tree structure, the *optimal binary search tree*, which yields a guarantee of the *average* number of key comparisons over all possible searches. Binary search trees were independently discovered by a number of people. Gilbert and Moore (1959) provided the basis for constructing optimal binary search trees when the probabilities of each key being searched for are known.

Yao (1981) also observed that by allowing one additional item to be stored, the $\Omega(\log n)$ bound can be beat and constant-time FIND performance is possible. Typically, the additional item is the name of a hash function. Fredman, Komlòs, and Szemerédi (1984) showed that there is a structuring technique which uses $n+o(n)$ space and attains $O(1)$ time for the FIND operation. The tool of the final section is based on the improved technique of Czech, Havas, and Majewski (1992) and Majewski, Wormald, Havas, and Czech (1996).

SORTED TABLES

A *comparison-based* search technique is one in which information about the location of a key is obtained by comparing it with keys in the table. Comparison-based techniques require that the keys involved be from a totally ordered set so that any key can be compared with any other key. It is known (Yao, 1981) that $\Omega(\log n)$ comparisons are required for comparison-based static tables. Binary search (presented below) attains this cost and is, therefore, an optimal approach.

Binary Search in a Sorted Array

A sorted array $A[\ ]$ can support an efficient static table. Suppose that the key x is being searched for in $A[\ ]$. Because the array is in sorted order, com-