Symbol Tables

Bùi Tiến Lên

2021

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Contents

Introduction

Concept 1

A **symbol table** is a data structure of **key-value** pair abstraction that supports two basic operations:

- **Insert** a value (item) with specified key.
- Given a key, **search** for the corresponding value.

- For example, DNS lookup.
  - Insert domain name with specified IP address.
  - Given domain name, find corresponding IP address.

Search

	Target known	Target unknown
Location known	 *Lookup*	 *Browse*
Location unknown	 *Locate*	 *Explore*

Symbol table applications

Application	Purpose Of Search	Key	Value
dictionary	find definition	word	definition
book index	find relevant pages	term	list of page numbers
file share	find song to download	name of song	computer ID
financial account	process transactions	account number	transaction details
web search	find relevant web pages	keyword	list of page names
compiler	find properties of variables	variable name	type and value
routing table	route Internet packets	destination	best route
DNS	find IP address	domain name	IP address
reverse DNS	find domain name	IP address	domain name
genomics	find markers	DNA string	known positions
file system	find file on disk	filename	location on disk

Abstract Data Type

Symbol-Table Abstract Data Type

```
template <class Key, class Value>
class SymbolTable {
private:
  // Implementation-dependent code
public:
  int count() = 0;
  Value search(Key) = 0;
  void insert(Key, Value) = 0;
  void remove(Key) = 0;
  Key select(int) = 0;
};
```

Conventions

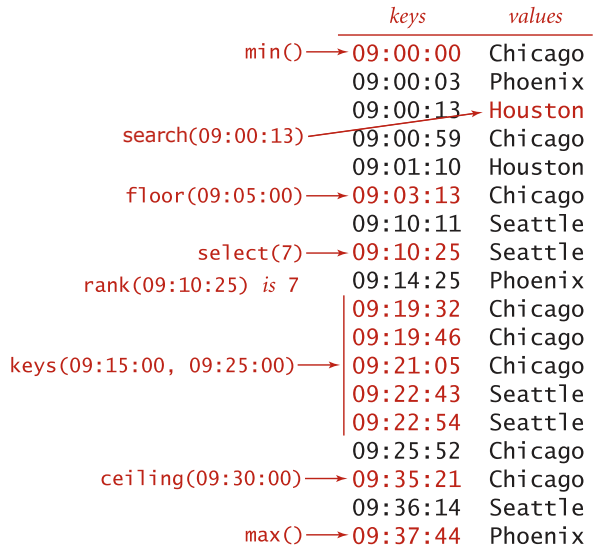
- Value type:
  - Any generic type.
  - Values are not null. (`nullValue`)
- Key type:
  - Keys are any generic type.
  - Keys are Comparable.
  - Keys are unique and not null. (`nullKey`)

Ordered Symbol-Table Abstract Data Type

- For ordered symbol-table, we need the following additional methods

Methods	Meanings
`Key min()`	smallest key
`Key max()`	largest key
`Key floor(Key key)`	largest key less than or equal to key
`Key ceiling(Key key)`	smallest key greater than or equal to key
`Key select(int k)`	key of rank k
`int rank(Key key)`	number of keys less than key
`vector<Key> range(int l, int r)`	keys in sorted set of keys [l..r]
`vector<Key> keys(Key lo, Key hi)`	keys in [lo..hi], in sorted order

Examples of ordered symbol table API



Elementary Implementations

- Sequential Search
- Binary Search
- Interpolation Search
- Selection

Array-based Symbol Table

```
template <class Key, class Value>
class ArraySymbolTable: public SymbolTable<Key, Value> {
private:
  Value *values;
  Key *keys;
  int N;
public:
  ArraySymbolTable() {
    ...
  }
  ...
};
```

Sequential Search

- The `search` function can scan through the array of `keys` to look for an item with the specified `key`, returning `nullValue` when encountering an item with a larger key

```
Value seqsearch(int l, int r, Key key) {
  for(int i=l; i<=r; i++)
    if (keys[i] == key) return values[i];
  return nullValue;
}
Value search(Key key) {
  return seqsearch(0, N-1, key);
}
```

- **Challenge**: Can we make any improvement?

Analysis

Theorem 1

Sequential search in a symbol table with N ordered items uses about $N/2$ comparisons for ***search hits*** *andh* ***search misses*** *(on the average)*

Binary Search

Idea

Given that the array `keys` is sorted, the search checks the middle element of the active region.

- If the middle element is the target element, the search terminates.
- Otherwise, the search recursively continues to the left or right half of the region, depending on the value of the middle element.

Implementation

```
Value binsearch(int l, int r, Key key) {
    int m;
    do {
        m = (l + r) / 2;
        if (keys[m] == key)
            return values[m];
        else if (keys[m] > key)
            r = m - 1;
        else
            l = m + 1;
    } while (l <= r);
    return nullValue;
}
```

- **Challenge**: Reimplement the function using recursion.

Analysis

Theorem 2

Binary search never uses more than $\log_2(N+1)$ *comparisons for a search (hit or miss)*

Interpolation Search

- We can replace the formula

$$m \leftarrow l + \frac{1}{2}(r - l) \tag{1}$$

with

$$m \leftarrow l + \frac{key - keys[l]}{keys[r] - keys[l]}(r - l) \tag{2}$$

Selection

Problem. Finding the k-th smallest of a set of keys without required full sort.

```
template <class Item>
void select(Item a[], int l, int r, int k) {
  if (r <= l) return;
  int i = partition(a, l, r);
  if (i > k) select(a, l, i - 1, k);
  if (i < k) select(a, i + 1, r, k);
}
```

- **Challenge**: reimplement the function without using recursion

Analysis

Theorem 3

Quicksort-based selection is linear time on the average

Cost summary for basic symbol-table implementations

implementation	worst case			average case			ordered iteration	key
	search	insert	remove	search hit	insert	remove		
unordered list	N	1	N	$N/2$	1	$N/2$	no	equal
ordered list	N	N	N	$N/2$	$N/2$	$N/2$	yes	compare
ordered array	$\log_2 N$	N	N	$\log_2 N$	$N/2$	$N/2$	yes	compare
goal?								

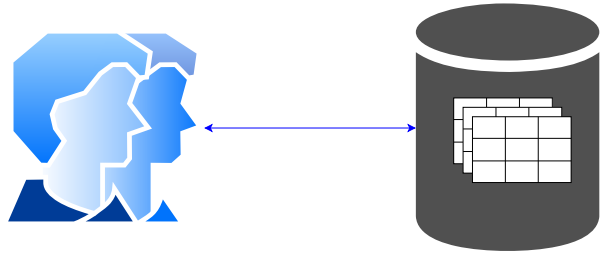
Workshop

Quiz

1. What is a symbol table?

Projects

1. (Big project) Design and implement a tiny relational database project.



References

Cormen, T. H. (2009).
Introduction to algorithms.
MIT press.

Sedgewick, R. (2002).
Algorithms in Java, Parts 1-4, volume 1.
Addison-Wesley Professional.

Walls and Mirrors (2014).
Data Abstraction And Problem Solving with C++.
Pearson.