CSCI 136: Data Structures and Advanced Programming

Lecture 27

Hash tables

Instructor: Kelly Shaw Williams

Your to-dos

1. Read **before Wed**: Bailey, Ch 16-16.2.
2. Lab 9 (partner lab), **due Tuesday 11/29 by 10pm**.
3. Quiz, due Saturday evening.

Topics

- Hash tables
- Hash functions

Announcements

Computer Science Colloquium
Friday, November 18 @ 2:35pm in Wege (TCL 123)
Daniel Malinsky (Columbia)

*Identifying Causal Determinants of Clinical Outcomes from Electronic Health Records Using Graphical Structure Learning: Challenges and Opportunities in Causal Discovery*

Many goals within causal inference, including estimating average treatment effects and understanding path-specific mechanisms, depend on knowing the qualitative causal structure underlying a domain. In this work we apply methods for graphical causal discovery (specifically the FCI algorithm) to observational data in the form of electronic health records (EHR) from Johns Hopkins Hospital. Our goal is to understand the causal determinants of postoperative length of stay for patients undergoing cardiac surgery procedures, in order to inform possible interventions that support faster patient recovery. We discuss the challenges in applying causal discovery methods to electronic health records and opportunities for future work.
Hash tables

Dan’s favorite data structure

Note about lab 9:

You should use the structure5 Hashtable implementation.
But if you want the extra challenge, implement your own!

Recall: arrays

An array is a data structure consisting of a sequential collection of elements, each identified by an index.

<table>
<thead>
<tr>
<th>A</th>
<th>13</th>
<th>2</th>
<th>451</th>
<th>42</th>
<th>9</th>
<th>6</th>
<th>-4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Performance guarantees:
1. read an element: O(1)
2. write an element: O(1)

Can we capture some of this for a more general structure?

Generalization: associative array

An associative array is a data structure consisting of a sequential collection of elements, each identified by a key. An associative array is a map.

<table>
<thead>
<tr>
<th>A</th>
<th>13</th>
<th>2</th>
<th>451</th>
<th>42</th>
<th>9</th>
<th>6</th>
<th>-4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Joe</td>
<td>Adam</td>
<td>Sue</td>
<td>Ed</td>
<td>Sam</td>
<td>Fay</td>
<td>Dan</td>
<td>Ted</td>
</tr>
</tbody>
</table>

Performance guarantees:
1. read an element: O(1)?
2. write an element: O(1)?

How can we make this happen?
What about MapTree?

It is already a map, which is good, but...

<table>
<thead>
<tr>
<th>A</th>
<th>13</th>
<th>2</th>
<th>451</th>
<th>42</th>
<th>9</th>
<th>6</th>
<th>-4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>Adam</td>
<td>Sue</td>
<td>Ed</td>
<td>Sam</td>
<td>Fay</td>
<td>Dan</td>
<td>Ted</td>
<td></td>
</tr>
</tbody>
</table>

Performance guarantees:
1. **read** an element: $O(\log n)$ (assuming balance)
2. **write** an element: $O(\log n)$ (assuming balance)

Not fast enough!

Could we actually just use an array?

<table>
<thead>
<tr>
<th>A</th>
<th>13</th>
<th>2</th>
<th>451</th>
<th>42</th>
<th>9</th>
<th>6</th>
<th>-4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>Adam</td>
<td>Sue</td>
<td>Ed</td>
<td>Sam</td>
<td>Fay</td>
<td>Dan</td>
<td>Ted</td>
<td></td>
</tr>
</tbody>
</table>

What do you think? What's the **obstacle**?

Need: function to map key to index

Suppose we have a **function**:

$$h(k) \to z$$

where $k$ is a key of **arbitrary type** and $z \in \mathbb{Z}_0^+$, then we could construct another function:

```java
int index(K key) {
    return h(key) % A.length;
}
```

Hash function

A **hash function** is any function that can be used to map data of **arbitrary size** onto data of a **fixed size**.

<table>
<thead>
<tr>
<th>A</th>
<th>13</th>
<th>2</th>
<th>451</th>
<th>42</th>
<th>9</th>
<th>6</th>
<th>-4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>Adam</td>
<td>Sue</td>
<td>Ed</td>
<td>Sam</td>
<td>Fay</td>
<td>Dan</td>
<td>Ted</td>
<td></td>
</tr>
</tbody>
</table>

String of length 2.
String of length 3.
String of length 4.

Why not “Benedict Cumberbatch”? 
Hash table

A hash table is a data structure that implements the map abstract data type. A hash table uses a hash function to compute an index into an array of buckets, from which the desired value can be found.

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>-4</th>
</tr>
</thead>
</table>
```

“Dan”, -4
index(“Dan”) → 6
A[index(“Dan”)] = -4

Hash function

Hash functions must also provide the following guarantees:

- **Determinism**: a given input value must always generate the same hash value.
- **Uniformity**: maps the expected inputs as evenly as possible over its output range.
- **Equivalence**: any two values that are considered equivalent should produce the same hash value.

Question

Is a function that generates a random number a good hash function?

**No.** Random numbers do tend to be uniform, but are not deterministic.

Activity

See if you can come up with a simple hash function for strings.

- **Determinism**: a given input value must always generate the same hash value.
- **Uniformity**: maps the expected inputs as evenly as possible over its output range.
- **Equivalence**: any two values that are considered equivalent should produce the same hash value.
Hash codes

Hashing is so important that every object in Java has a built-in hash function.

<table>
<thead>
<tr>
<th>Public (int hashCode)</th>
<th>public (int hashCode)</th>
<th>public (int hashCode)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a hash code value for the object. This method is supported for the benefit of hash tables such as those provided by HashSet.</td>
<td>Returns a hash code value for the object. This method is supported for the benefit of hash tables such as those provided by HashSet.</td>
<td>Returns a hash code value for the object. This method is supported for the benefit of hash tables such as those provided by HashSet.</td>
</tr>
<tr>
<td>The general contract of hashCode is:</td>
<td>The general contract of hashCode is:</td>
<td>The general contract of hashCode is:</td>
</tr>
<tr>
<td>- Whenever it is invoked on the same object more than once during an execution of a Java application, the hashcode method must consistently return the same integer, provided no information used in hashcode computations on the object is modified. The integer need not remain consistent if the object is modified. If two objects are equal according to the equals(Object) method, then calling the hashcode method on each of the two objects must produce the same integer result.</td>
<td>- Whenever it is invoked on the same object more than once during an execution of a Java application, the hashcode method must consistently return the same integer, provided no information used in hashcode computations on the object is modified. The integer need not remain consistent if the object is modified. If two objects are equal according to the equals(Object) method, then calling the hashcode method on each of the two objects must produce the same integer result.</td>
<td>- Whenever it is invoked on the same object more than once during an execution of a Java application, the hashcode method must consistently return the same integer, provided no information used in hashcode computations on the object is modified. The integer need not remain consistent if the object is modified. If two objects are equal according to the equals(Object) method, then calling the hashcode method on each of the two objects must produce the same integer result.</td>
</tr>
<tr>
<td>If two objects are unequal according to the equals (Object) method, then calling the hashcode method on each of the two objects must produce a different integer result.</td>
<td>If two objects are unequal according to the equals (Object) method, then calling the hashcode method on each of the two objects must produce a different integer result.</td>
<td>If two objects are unequal according to the equals (Object) method, then calling the hashcode method on each of the two objects must produce a different integer result.</td>
</tr>
<tr>
<td>As much as is reasonably practical, the hashcode method defined by class Object does return distinct integer values for distinct objects. (This is typically implemented by converting the internal address of the object into an integer, but this implementation technique is not required by the Java programming language.)</td>
<td>As much as is reasonably practical, the hashcode method defined by class Object does return distinct integer values for distinct objects. (This is typically implemented by converting the internal address of the object into an integer, but this implementation technique is not required by the Java programming language.)</td>
<td>As much as is reasonably practical, the hashcode method defined by class Object does return distinct integer values for distinct objects. (This is typically implemented by converting the internal address of the object into an integer, but this implementation technique is not required by the Java programming language.)</td>
</tr>
<tr>
<td>Returns:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Is our simple hash function actually good?

Recap & Next Class

**Today:**
- Hash tables
- Hash functions

**Next class:**
- Collisions
- Graphs