Data Structure Concepts

(2008.07.15)

Frank Ducrest
Data Structure

• Organization of data in a useful way

• Example: an array is the most commonly used data structure and is considered the simplest data structure

• Different data structures organize data in different ways

• The way that a data structure organizes data determines how data stored in the data structure can be accessed

• Examples of common data structures:

  array, list, stack, queue, tree, dictionary (mapping)
Arrays

• Random access data structure; elements of data can be accessed sequentially or in any order desired via an index

• Advantages: simple in concept and notation, may be used to store unordered or ordered data

• Disadvantages: inefficient when memory space needs to be enlarged or reduced, inefficient when the largest possible memory needed is much larger than the expected amount of memory usage, inefficient when used to contain ordered but dynamic data, requires contiguous memory
List

- Sequential access data structure
- Data may be ordered or unordered, inside an array or as independent elements
- Advantages: simple in concept and navigation (traversal), memory given to each element need not be contiguous, only need to allocate memory for each actual element, well suited to insertion of unordered volatile data
- Disadvantages: sequential (traversal) navigation only so location / change / removal of data elements is inefficient, insertion of ordered volatile data inefficient
Stack

- A first in, last out (FILO) data structure; elements are pushed onto the top or popped off of the top
- Can be stored in an array or as independent elements
- Example of uses: accounting, inventory control, evaluation of expressions in a running program
Queue

UK for “line”

A first in, first out (FIFO) data structure; elements enter from the rear and are removed from the front; egalitarian in nature; a restricted list

Can be stored in an array or as independent elements

Example of uses: simulation (such as a bank queue), control of access to common resources (such as a network printer)
**Binary-Tree**

- Data structure where each element is a node that has two children
- Very efficient to build and search if the data being inserted is fairly random
- Requires balancing of subtrees if data volatile
Dictionary

• Also called hash table, map or index-sequential

• Mapping of elements in an ordered data structure to elements in an unordered data structure; i.e. a key is matched to a value

• Values in the elements in ordered data structure act as a key value; the ordered data structure is searched, when the desired element is found, the location of the element in the unordered data structure (which is part of the element located in the ordered data structure) is revealed

• Efficient in certain circumstances; for example: when data elements are very large, but index elements are small; i.e. data cannot fit into memory, but index can
## Simple Dictionary (Map) Illustration

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Employee Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>Barney</td>
</tr>
<tr>
<td>19</td>
<td>Rubble</td>
</tr>
<tr>
<td>35</td>
<td>303 Cobblestone Way</td>
</tr>
<tr>
<td>37</td>
<td>Bedrock</td>
</tr>
<tr>
<td>54</td>
<td>...</td>
</tr>
<tr>
<td>63</td>
<td>Fred</td>
</tr>
<tr>
<td>70</td>
<td>Flintstone</td>
</tr>
<tr>
<td>72</td>
<td>301 Cobblestone Way</td>
</tr>
<tr>
<td>...</td>
<td>Bedrock</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>