ITT DUBLIN
Institute of Technology Tallaght
Institiúid Teicneolaíochta Tamhlacht

The Dictionary Abstract Data Type

Goodrich et al. chapter 8

ITT DUBLIN
Institute of Technology Tallaght
Institiúid Teicneolaíochta Tamhlacht

The Dictionary ADT

- Dictionary data structures enable us to store, search for and retrieve elements from a collection based on some key value associated with them.
- We have already studied some data structures which have this ability.
  - Last semester we studied the hash table ADT (recall we stored a student in a hash table based on using the X number as a key)
  - This semester we have studied the associative container `map` in the STL
  - Each of these is an example of a dictionary
- An item to be stored in a dictionary has the form (k, e) where k is the key and e is the element
- Dictionaries may be defined to have unique keys
  - as in the STL `map`
- Or they may be defined to store multiple items with the same key (duplicates)
  - The STL provides a template class called `multimap` which supports duplicate keys

ITT DUBLIN
Institute of Technology Tallaght
Institiúid Teicneolaíochta Tamhlacht

Ordered and Unordered Dictionaries

- The essential feature of a dictionary is that it enables us to look things up based on a key
- Dictionaries may be ordered or unordered
  - In either case we use a key as an identifier that is assigned to each element by a user or application.
- In the ordered dictionary, items are stored in an order which is based on their key values.
  - a 'total order' relation must be defined on the keys …
  - … means we understand 'greater than', 'less than' as well as 'equal to'
  - The dictionary must be able to support operations like
    `closestBefore(k)`
    `closestAfter(k)`
- In the unordered dictionary, no such ordering exists.
  - we can only test for equality of keys.
  - The dictionary can be implemented using an unordered vector, list or general sequence to store the key-value pairs

Basic operations supported by a Dictionary ADT

- *Note, these are not the names of the methods provided by the `map` and `multimap` STL template classes, but methods with similar functionality exist*
  - size()
  - isEmpty()
  - elements() returns an iterator to the elements stored in the map
  - keys() returns an iterator to the keys stored in the map
  - find(k) return position of item with key k, or null
  - findAll(k) returns an iterator of positions for all items with key k
  - insertItem(k, e)
  - removeElement(k) an error condition if no such k found in map
  - removeAllElements(k) removes some no. of elements (may be 0)

- Notice that none of these methods assume any notion of an 'order' in the way the items are stored.
  - They only assume that it is possible to check that the key of an item matches a given key

Dictionaries: performance analysis

- The dictionary implemented with an unordered sequence can insert elements in constant time
  - Because no sorting based on the key is needed
- But removal or retrieval requires scanning through the entire sequence
  - Thus worst case running time for retrieval is O(n)
- The unordered dictionary is a good implementation in cases where there are many insertions into the dictionary, but relatively few extractions
- In an ordered dictionary, elements are stored in an order based on their key value
- The key value pairs may be stored in a vector referred to as a look-up table
- Insertions may take O(n) time
  - If we need shift all items along to accommodate it
- Retrievals are very fast, because we can use a binary search
  - Which is O(logn)
- The ordered dictionary is a better option if there are relatively fewer insertions than there are retrievals from the dictionary

ITT DUBLIN
Institute of Technology Tallaght
Institiúid Teicneolaíochta Tamhlacht

Unordered Dictionaries – log files

- Unordered dictionaries are also referred to as log files or audit trails
- Consider situations where we wish to archive structured data
  - For example, financial database systems storing all their transactions
  - OS programs such as web servers storing all requests they process over the internet
- Such archive or logging requirements are often implemented with a dictionary realised by using an *un-ordered* vector, list or general sequence to store key-element pairs
- And such an implementation is often called a log file or audit trail
- Archiving database and operating system transactions have precisely the requirements provided by an unordered dictionary
  - Insertions must be simple and fast
  - Retrievals are based on a key value
  - But are only needed when something goes wrong, so performance is not such an issue

ITT DUBLIN
Institute of Technology Tallaght
Institiúid Teicneolaíochta Tamhlacht

Ordered Dictionaries: improving insertion and retrieval times

- The ordered dictionary can never equal the log-file for insertions
  - That would be constant time
- Retrievals are very fast, because we can use a binary search
  - Which is O(logn)
- But insertions may take O(n) time if we use an ordered vector to store the data
- Ideally, we need a different data structure than a vector to use within the dictionary implementation
  - which will improve insertion time …
  - … without compromising the retrieval time
- Insertion time can be improved if the data is stored internally in a binary search tree instead of a vector
  - If the BST is well balanced Insertions and retrievals may both be done in O(logn) time, a huge improvement
    - But it may not remain well-balanced, It depends on the order in which items are inserted
  - Insertions and retrievals from a BST in the worst case are O(n)

- Our next topic will explore other tree structures to use for a dictionary – for example a balanced binary search tree