Dictionary Implementations

Chapter 20

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Contents

- Array-Based Implementations
  - An Unsorted Array-Based Dictionary
  - A Sorted Array-Based Dictionary
- Vector-Based Implementations
- Linked Implementations
  - An Unsorted Linked Dictionary
  - A Sorted Linked Dictionary

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Objectives

- Implement the ADT dictionary by using either
  - An array
  - A vector
  - A chain of linked nodes

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Array-Based Implementations

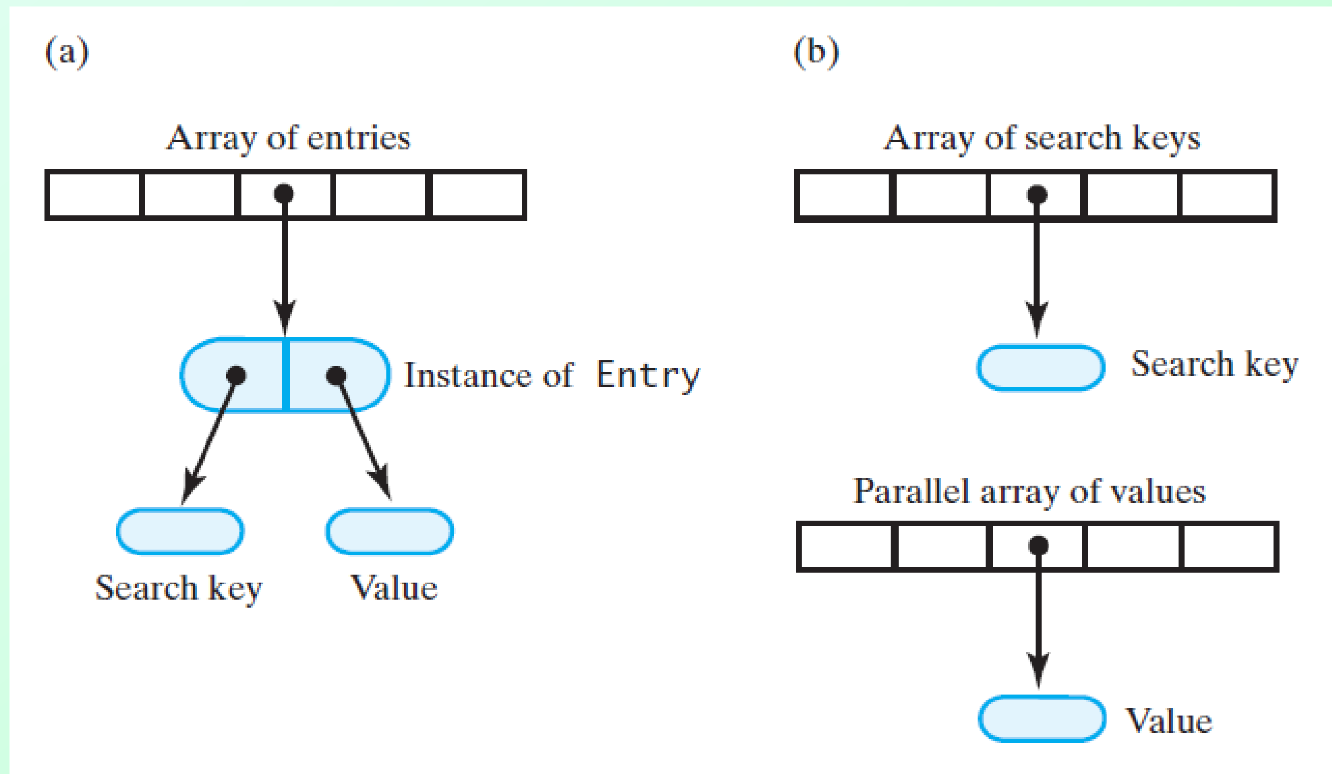
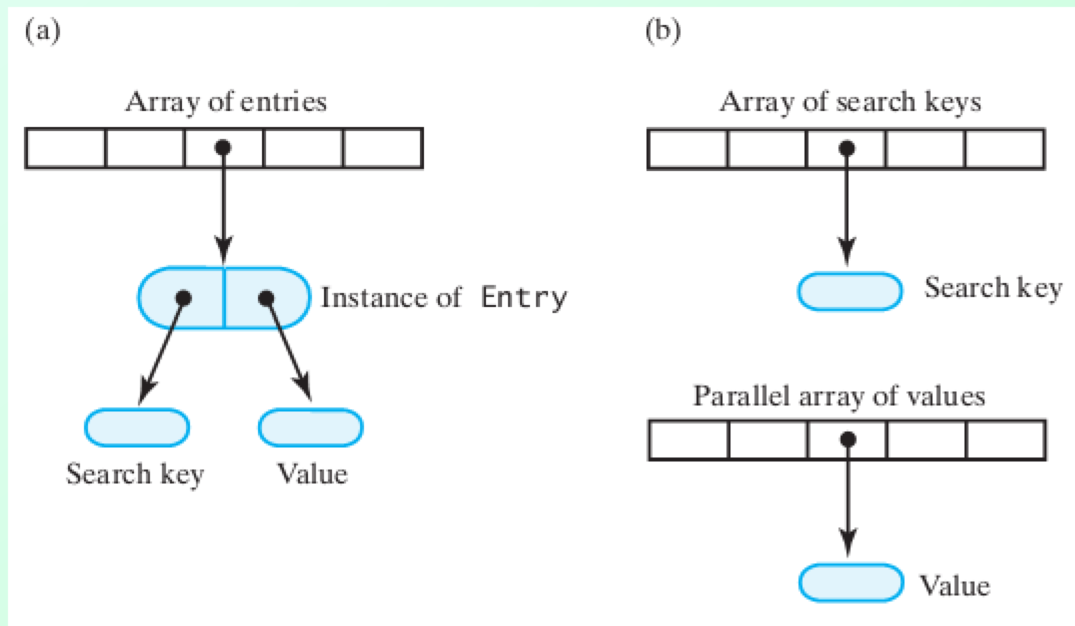


Figure 20-1 Two possible ways to use arrays to represent the entries in a dictionary: (a) an array of objects that encapsulate each search key and corresponding value; (b) parallel arrays of search keys and values

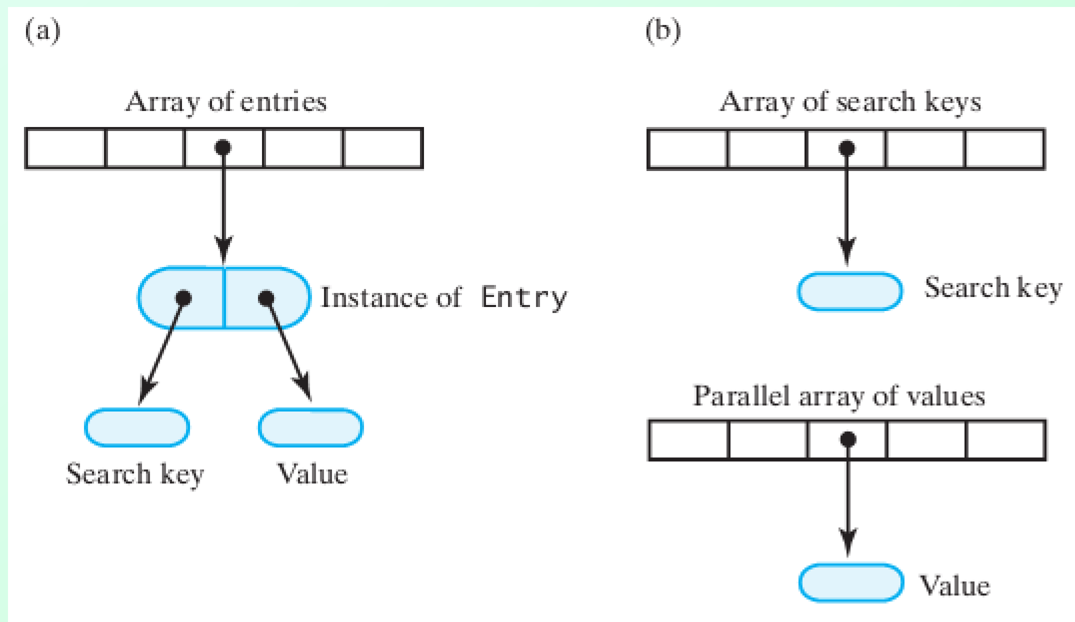
Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 1 How do the memory requirements for the two representations compare?



Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 1 How do the memory requirements for the two representations compare?



The memory requirements for the search keys and the values are the same for each representation, so let's ignore them. The memory requirement for the representation shown in (a) uses three references for each entry in the dictionary: one in the array and two in the Entry object. The parallel arrays in (b) require only two references for each dictionary entry. Thus, for n entries in the dictionary, the representation in (a) requires 3n references, but the representation in (b) requires only 2n references. However, if each array has a length of m, where m is greater than n, (a) has m − n unused locations and (b) has twice that number.

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Array-Based Implementations

- We will implement an array of objects as shown in Figure 20-1a
- View Listing 20-1
  - Class **`ArrayDictionary`**
  - Inner class **`Entry`**

Note: Code listing files must be in same folder as PowerPoint files for links to work

Copyright ©2012 by Pearson Education, Inc. All rights reserved

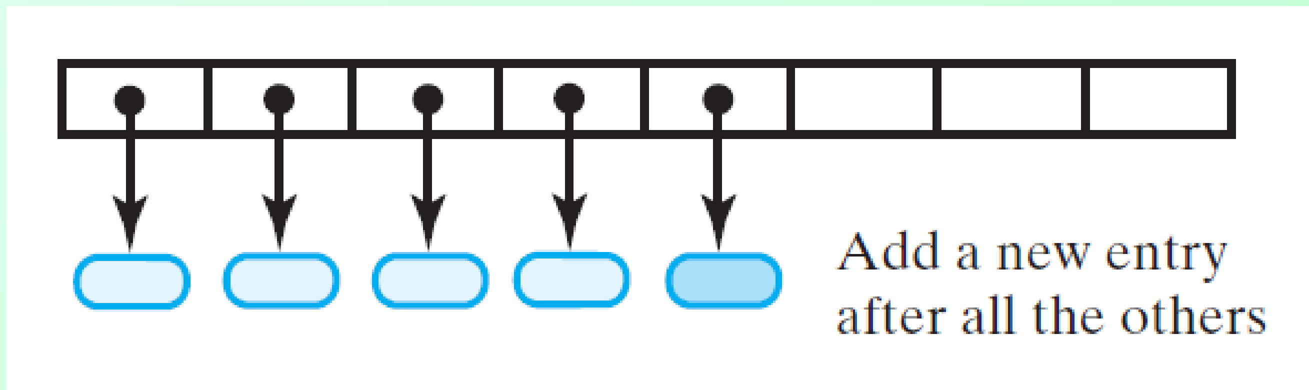


Figure 20-2 Adding a new entry to an unsorted array-based dictionary

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Figure 20-3 Removing an entry from an unsorted array-based dictionary

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Array-Based Implementations

- Worst case efficiencies
  - Addition O(n)
  - Removal O(n)
  - Retrieval O(n)
  - Traversal O(n)
- At same time realize overhead required for occasionally enlarging array

Copyright ©2012 by Pearson Education, Inc. All rights reserved

A Sorted Array-Based Dictionary

- Search keys must belong to a class that implements the interface `Comparable`
- Some of implementation for unsorted dictionary can still be used
- View outline of class, Listing 20-2

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 2 Describe how the previous algorithm for a sorted array-based dictionary differs from the one given in Segment 20.4 for an unsorted dictionary.

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 2 Describe how the previous algorithm for a sorted array-based dictionary differs from the one given in Segment 20.4 for an unsorted dictionary.

The initial search determines the insertion point when the dictionary is sorted, whereas the insertion point for an unsorted dictionary is always right after the last entry in the array. Insertion into a sorted dictionary generally requires shifting other entries in the array. No shifting is necessary for an unsorted dictionary.

Copyright ©2012 by Pearson Education, Inc. All rights reserved

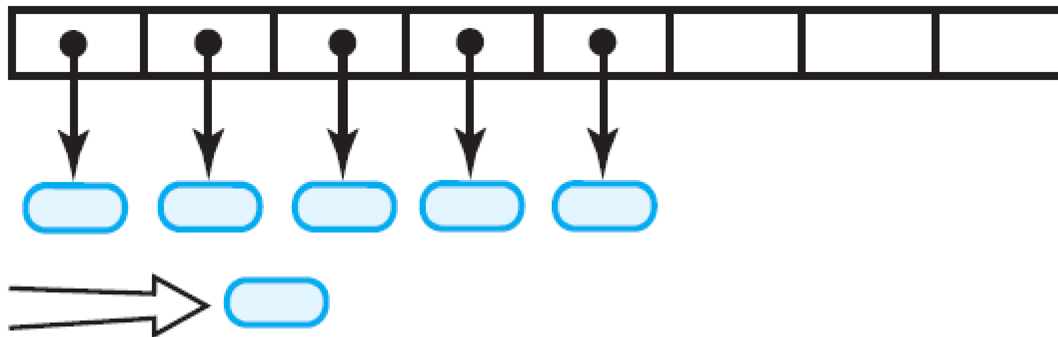


Figure 20-4 Adding an entry to a sorted array-based dictionary: (a) search;

Copyright ©2012 by Pearson Education, Inc. All rights reserved

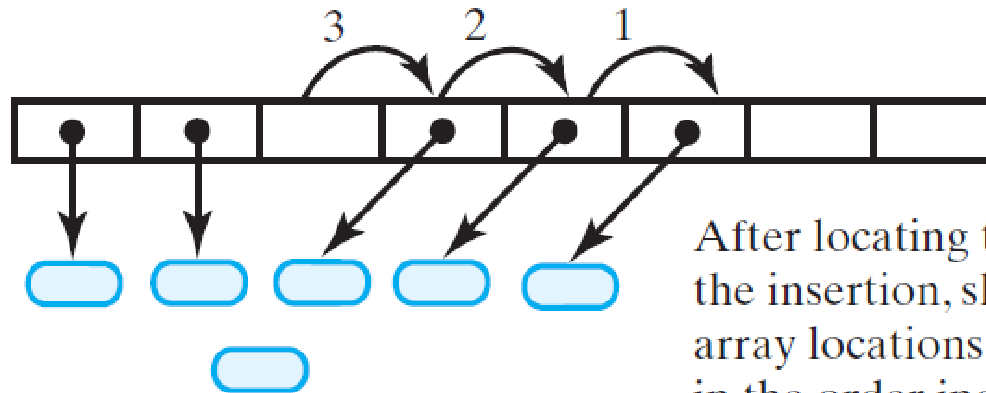


Figure 20-4 Adding an entry to a sorted array-based dictionary: (b) make room;

Copyright ©2012 by Pearson Education, Inc. All rights reserved

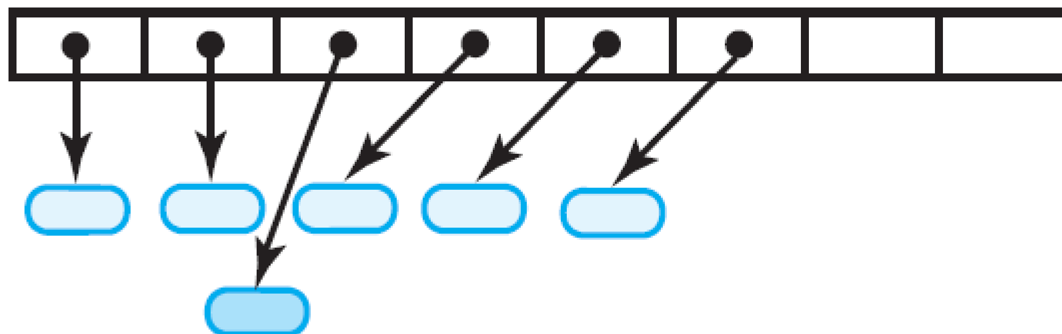


Figure 20-4 Adding an entry to a sorted array-based dictionary: (c) insert

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 3 A binary search would be faster, in general, than the modified sequential search just given—particularly when the dictionary is large. Implement the private method locateIndex for a sorted dictionary using a binary search.

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 3 A binary search would be faster, in general, than the modified sequential search just given—particularly when the dictionary is large. Implement the private method locateIndex for a sorted dictionary using a binary search.

```
private int  locateIndex(K key)
{       return  binarySearch(0, numberOfEntries - 1, key);
}
private int  binarySearch(int  first,  int  last, K key)
{       int  result;
        if (first > last)
                result = first;
        else
        {       int  mid = first + (last - first) / 2;
                K midKey = dictionary[mid].getKey();
                if (key.equals(midKey))
                        result = mid;
                else if (key.compareTo(midKey) < 0)
                        result = binarySearch(first, mid - 1, key);
                else
                        result = binarySearch(mid + 1, last, key);
        }
        return  result;
}
```

Copyright ©2012 by Pearson Education, Inc. All rights reserved

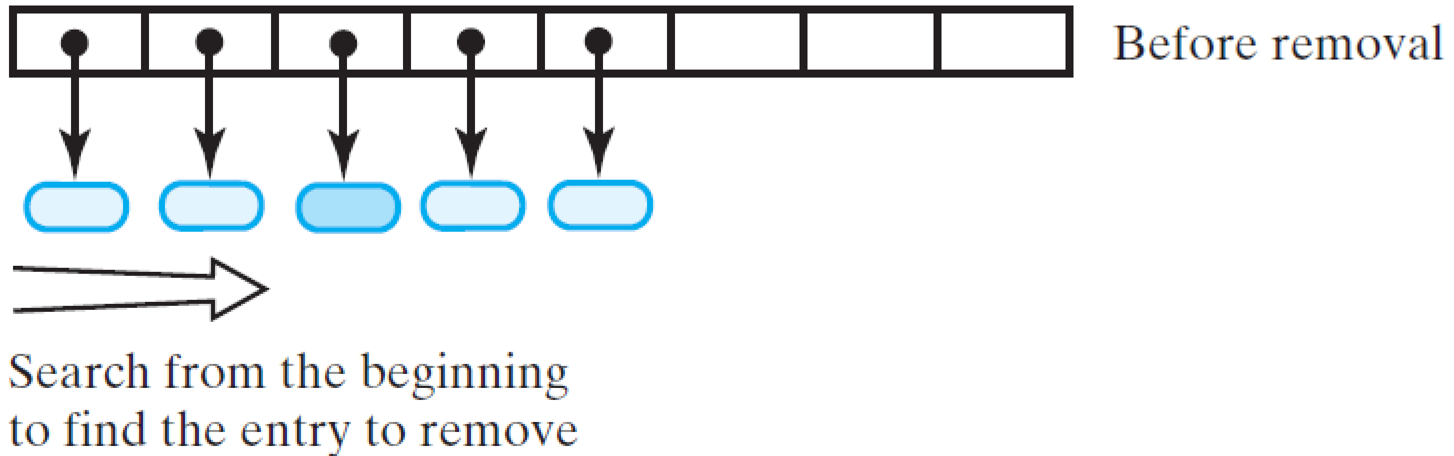


FIGURE 20-5 Removing an entry from a sorted array-based dictionary: (a) search;

Copyright ©2012 by Pearson Education, Inc. All rights reserved

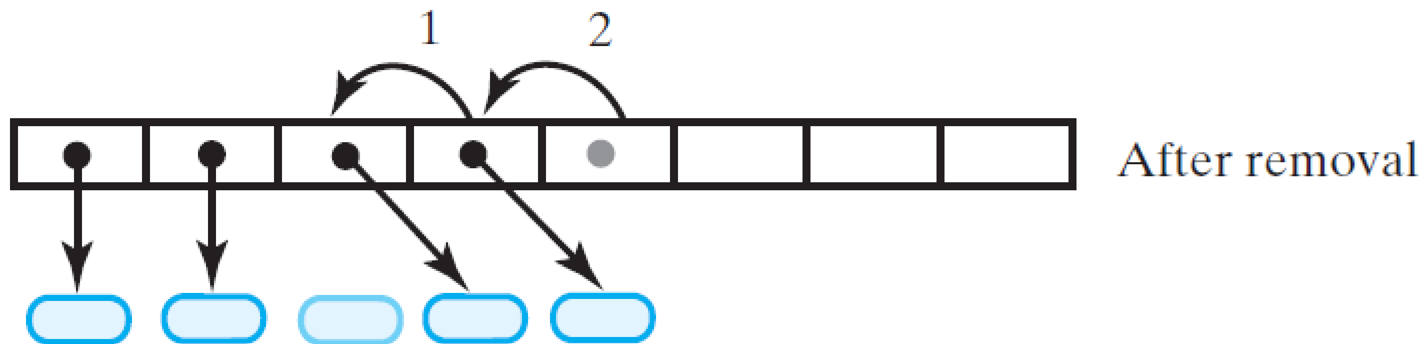


FIGURE 20-5 Removing an entry from a sorted array-based dictionary: (b) shift entries

Copyright ©2012 by Pearson Education, Inc. All rights reserved

A Sorted Array-Based Dictionary

- Worst-case efficiencies when `locateIndex` uses a binary search,
  - Addition O(n)
  - Removal O(n)
  - Retrieval O(log n)
  - Traversal O(n)

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 4 When the sorted array-based implementation of a dictionary uses a binary search, its retrieval operation is O(log n). Since add and remove use a similar search, why are they not O(log n) as well?

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 4 When the sorted array-based implementation of a dictionary uses a binary search, its retrieval operation is O(log n). Since add and remove use a similar search, why are they not O(log n) as well?

Typically, add must shift array entries to make room for a new entry, and remove must shift array entries to avoid a vacancy within the array. These shifts of data are O(n) operations in the worst case. The best case occurs when the addition or removal is at the end of the array. These operations are O(1).

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Vector-Based Implementations

- Similar in spirit to an array based implementation
- Can use one or two vectors
- With vector do not need methods
  - **`ensureCapacity,`**
  - **`makeRoom,`**
  - **`removeArrayEntry`**

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Vector-Based Implementations

- Consider Listing 20-3, **`SortedVectorDictionary`**
- Note implementation of class **`KeyIterator`**. Listing 20-4

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Linked Implementations

- Chain of linked nodes
- Chain can provide as much storage as necessary
- Encapsulate the two parts of an entry into an object

Copyright ©2012 by Pearson Education, Inc. All rights reserved

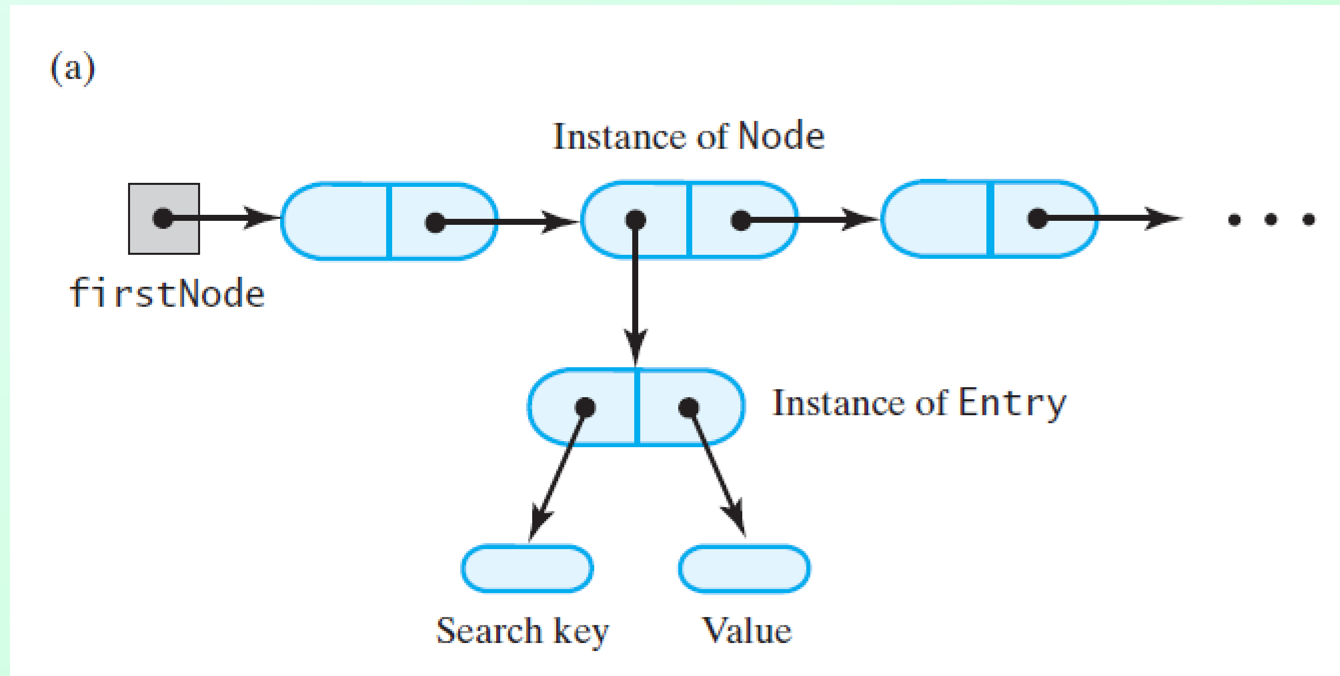


Figure 20-6 Three possible ways to use linked nodes to represent the entries in a dictionary: (a) a chain of nodes that each reference an entry object;

Copyright ©2012 by Pearson Education, Inc. All rights reserved

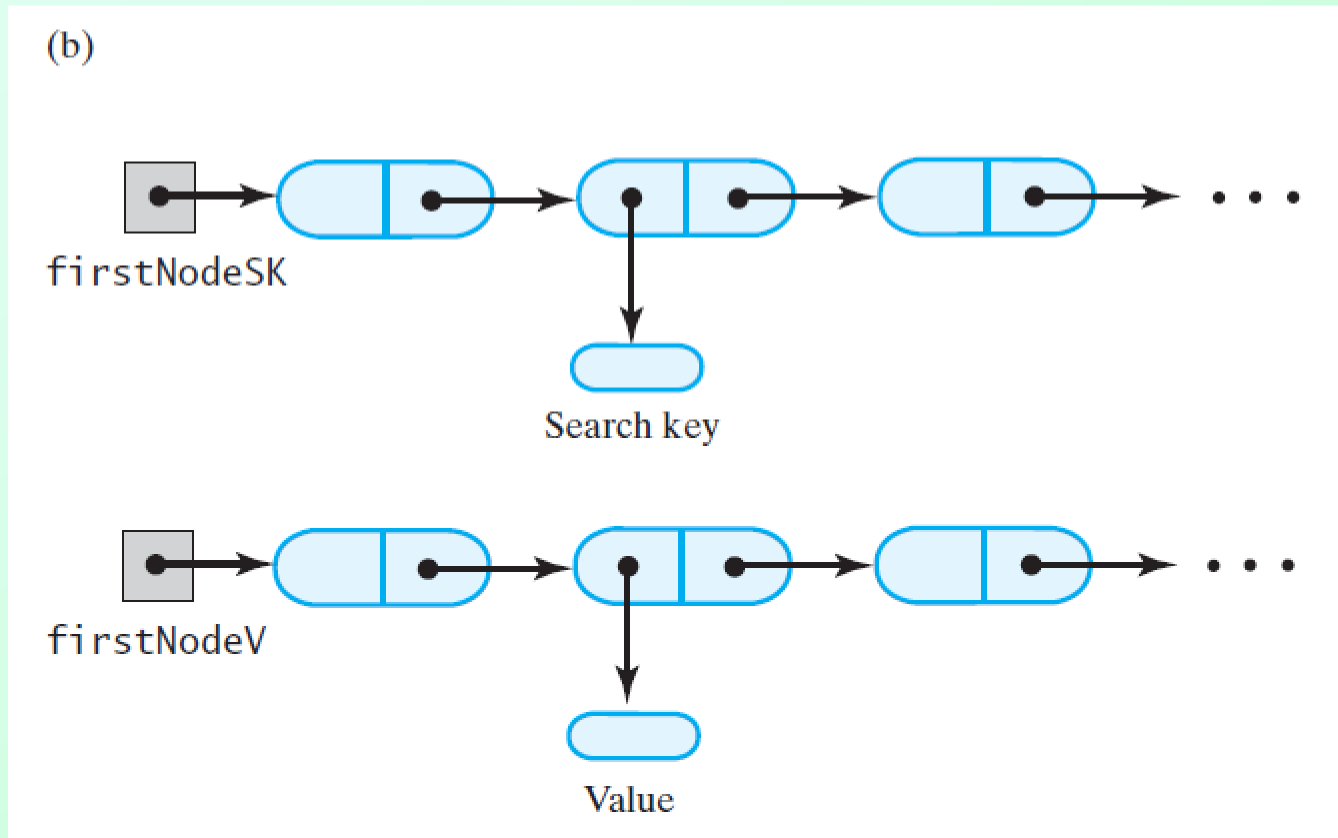


Figure 20-6 Three possible ways to use linked nodes to represent the entries in a dictionary:
(b) parallel chains of search keys and values;

Copyright ©2012 by Pearson Education, Inc. All rights reserved

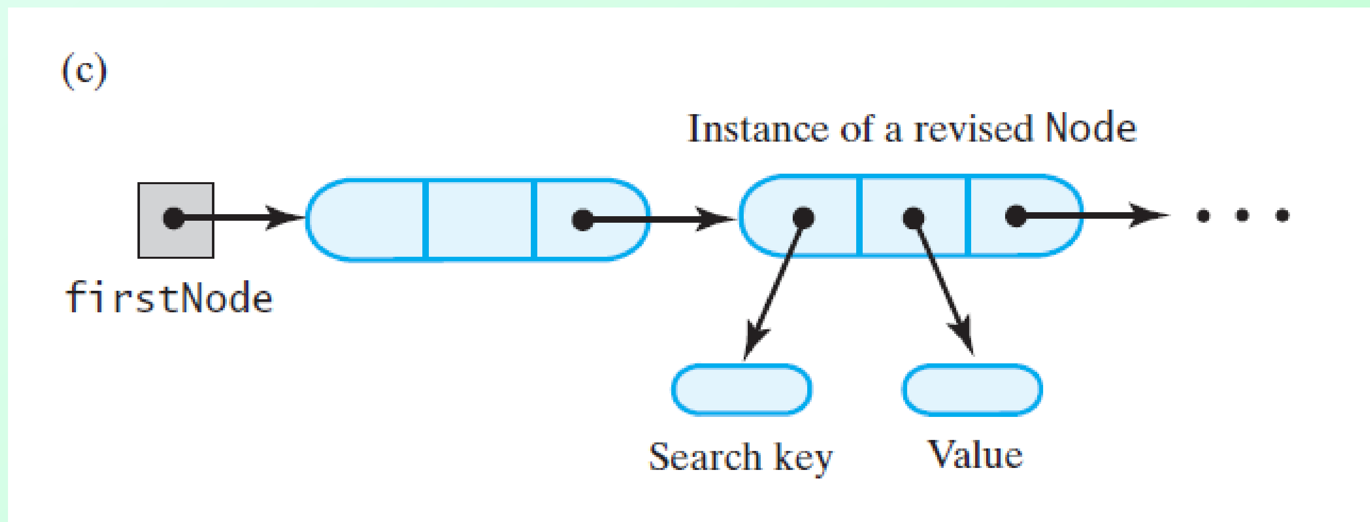


Figure 20-6 Three possible ways to use linked nodes to represent the entries in a dictionary: (c) a chain of nodes that each reference a search key and a value

Copyright ©2012 by Pearson Education, Inc. All rights reserved

An Unsorted Linked Dictionary

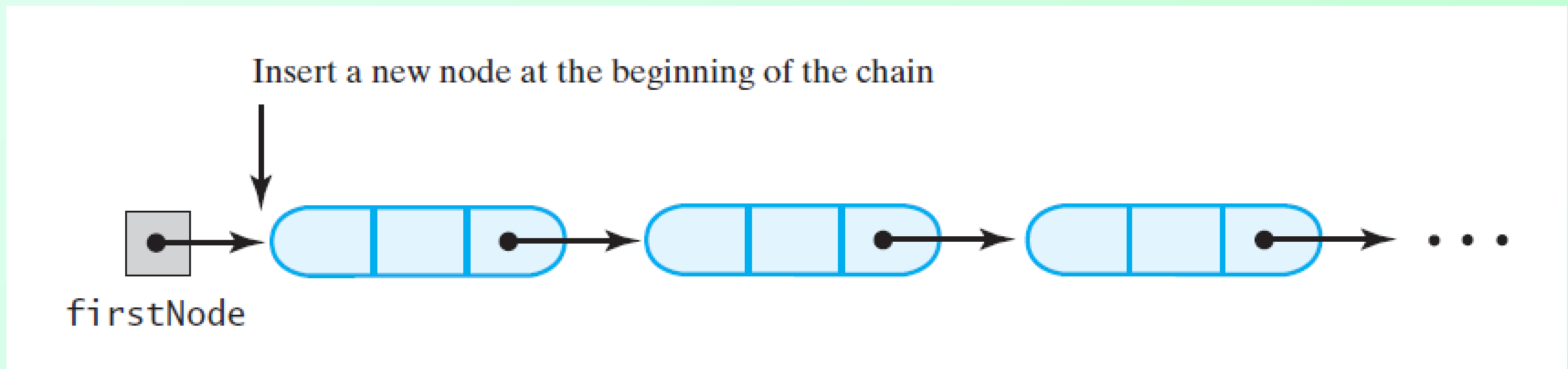


Figure 20-7 Adding to an unsorted linked dictionary

Copyright ©2012 by Pearson Education, Inc. All rights reserved

An Unsorted Linked Dictionary

- For this implementation, worst-case efficiencies of operations
  - Addition O(*n*)
  - Removal O(*n*)
  - Retrieval O(*n*)
  - Traversal O(*n*)

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 5 To remove an entry from an unsorted array-based dictionary, we replaced the removed entry with the last entry in the array (see Segment 20.6). Should we use the same strategy to remove an entry from an unsorted linked dictionary? Explain.

Copyright ©2012 by Pearson Education, Inc. All rights reserved

Question 5 To remove an entry from an unsorted array-based dictionary, we replaced the removed entry with the last entry in the array (see Segment 20.6). Should we use the same strategy to remove an entry from an unsorted linked dictionary? Explain.

No. Replacing the entry to be removed with the last entry in a chain would require a traversal of the chain. We would need references to both the last node and the next-to-last node so that we could delete the last node. Although we can ignore the last entry in an array, we should shorten the chain by setting the link portion of the next-to-last node to null. Note that having a tail reference does not eliminate the need for a traversal, since we need but do not have a reference to the next-to-last node . The strategy for an unsorted array-based dictionary avoids shifting any of the other entries. No shifting is needed in a linked implementation. After locating the node to delete, you simply adjust either the head reference or the reference in the preceding node.

Copyright ©2012 by Pearson Education, Inc. All rights reserved

A Sorted Linked Dictionary

- Adding a new entry requires sequential search of chain
  - Do not have to look at the entire chain, only until pass node where it should have been
- Listing 20-5, class **`SortedLinkedDictionary`**
- Note private inner class for iterator, Listing 20-6

Copyright ©2012 by Pearson Education, Inc. All rights reserved

A Sorted Linked Dictionary

- Worst case efficiencies of dictionary operations for sorted linked implementation
  - Addition O(n)
  - Removal O(n)
  - Retrieval O(n)
  - Traversal O(n)

Copyright ©2012 by Pearson Education, Inc. All rights reserved

End

Chapter 20

Copyright ©2012 by Pearson Education, Inc. All rights reserved