Ulster University

COM410 Programming in Practice

B3.1 Sequential Search

People are always looking for something . . .

- Searching is one of the most common tasks done by computers
- Two elementary search strategies: (1) **Sequential Search**; (2) **Binary Search**

Previously mentioned in ADT `Bag` . . .

- (As previously seen) we search a list for a desired item using a method such as `contains()`

```
72          @return A newly allocated array of all of the entries in the list.
73                  list is empty, the returned array is empty.
74       */
75
76      public boolean contains(T anEntry);
77      /* Check whether this list contains a given entry
78
79         @param anEntry The object that is the desired entry
80
81         @return true if the list contains anEntry, false otherwise
82       */
83
84      public int getLength();
85      /* Gets the length of this list
```

```
public boolean contains(T anEntry) {
    boolean found = false;
    int index = 0;
    while (!found && index < this.numberOfEntries) {
        if (this.bag[index++].equals(anEntry)) found = true;
    }
    return found;
}
```

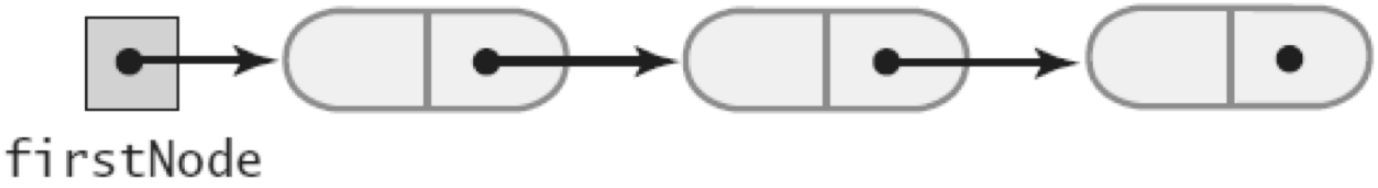
- The precise implementation of `contains()` depends on how the entries in the bag are stored (array or linked chain)
- Remember, `ArrayList` and `LinkedList` both provide a `contains()` method.

Efficiency of a Sequential Search of an Array

- **Best case**: desired item is first item in the array, so search will be **O(1)**
- **Worst case**: search the entire array (regardless of success), making *n* comparisons for array of *n* entries, so search will be **O(n)**
- **Average case**: typically search will look at one-half of the array entries, so will be O(n/2), which is just **O(n)**

Sequential Search of an Unsorted Chain

- Within a linked chain implementation of ADT List, method `contains()` would search the chain of linked nodes for the target



- Again, the sequential search will look at consecutive entries in the list beginning with the first node until it finds a node with data equivalent to the target or all nodes have been checked without success

Efficiency of a Sequential Search of a Chain

- **Best case**: desired item is in the first node of the chain, so search will be **O(1)**
- **Worst case**: search the entire chain (regardless of success), making *n* comparisons for chain of *n* entries, so search will be **O(n)**
- **Average case**: typically search will look at one-half of the nodes in the chain, so will be O(n/2), which is just **O(n)**

Sequentially Searching a Sorted Collection

- Traverse collection until it either reaches a node that contains **or is larger than** the desired item or examine all items in the collection without success
- If a larger item is found, then the element being searched for is not present

```
Algorithm inArray(array, entry)
// Returns true if array contains element, false otherwise

set index to 0
while index < array length and array[index] <= entry
     if array[index] equals entry
        return true
     end if
     add 1 to index
end while
return false
```

Scenario

- **Card in hand**
  - Revisit your **DataStructures** project and add a new class `Hand` which has an instance variable that stores an `Array` of `Card` objects sorted in ascending order of rank value. The class should contain a method `inHand()` that accepts a card as a parameter and uses a sequential search to return `true` if a card of equal rank is contained in the hand and `false` otherwise.
  - The `main()` method of the class should create a hand of 5 random cards sorted by ascending rank. The application should then generate another 10 cards at random, checking for each whether a card of equal rank value is contained in the hand.
  - Output the hand of cards followed by each random card in turn with a message stating whether or not a card of equal rank is contained in the hand.