COMP 250

Lecture 3

Array lists

Sept. 12, 2016
List

An ordered set of elements

\[ a_0, a_1, a_2, a_3, \ldots, a_{N-1} \]

\( N \) is the number of elements in the list, often called the “size” of the list.
What things do we do with a list?

get(i)       // Returns the i-th element (but doesn't remove it)
set(i,e)     // Replaces the i-th element with e
add(i,e)     // Inserts element e into the i-th position
remove(i)    // Removes the i-th element from list
remove(e)    // Removes element e from the list (if it is there)
clear()      // Empties the list.
isEmpty()    // Returns true if empty, false if not empty.
size()       // Returns number of elements in the list
Lists (next few lectures)

• array list (today)

• singly linked list

• doubly linked list
Arrays in Java

```java
int[ ] myInts = new int[15];


Shape[ ] shapes = new Shape[428];

shapes[298] = new Shape( △ );
```
int[ ] myInts = new int[15];

Shape[ ] shapes = new Shape[428];
shapes[298] = new Shape( ▲ );
Reference variables hold a number, sometimes called an “address” (in C and other languages), that indicates where an object is in the computer’s memory.

Often we illustrate a reference using an arrow.
Arrays have constant time access

A computer accesses an element in an array in “O(1)” time i.e. constant, independent of the length N of the array.

\[
\begin{align*}
    \text{\ldots} &= a[k] & // \text{ read} \\
    a[k] &= \text{\ldots} & // \text{ write}
\end{align*}
\]

address of \(a[k]\) = address of \(a[0]\) + \(k \times \text{size of array ‘slot’}\)

where ‘slot’ is the space required for the element stored in the array (either a primitive e.g. int or a reference i.e. address).
Arrays as Lists

Arrays are not lists.

But arrays can be used to make lists, sometimes called ‘array lists’.

Java has an ArrayList class.
size = 7
length = 8

size = 7
length = 9
What things do we do with a list?

get(i) // Returns the i-th element (but doesn't remove it)
set(i,e) // Replaces the i-th element with e
add(i,e) // Inserts element e into the i-th position
remove(i) // Removes the i-th element from list
remove(e) // Removes element e from the list (if it is there)
clear() // Empties the list.
isEmpty() // Returns true if empty, false if not empty.
size() // Returns number of elements in the list
Let’s assume that the array is \( a[\ ] \).

How to implement the various operations? (sketch only)

**get(i)**

```java
if (i >= 0) & (i < size) return a[i]
```

size = 6
length = 8
set(i, e)

if (i >= 0) & (i < size)
a[i] = e

set() replaces the object at index i.
add( i, e)

Make room by shifting, and then change reference.
add( i, e) {

    if  (i >=0) & (i <= size){

        for (j = size; j > i; j--)
            a[j] = a[j-1]  // shift (copy)

        a[i] = e       // replace value
        size = size + 1

    }

}
How to add an element to an array list when array is full?

```
add( i, e) {

    // Create an empty bigger array.

    // Copy all elements to bigger array.

    // Use add() code in earlier slide, with the bigger array.
}
```
How to add an element to an array list when array is full?

add( i, e) { // more general

    if (list size == array length){ // is array full?
        make new array (b) with 2 * length slots
        for ( int i=0; i < size; i++)
            b[i] = a[i] // copy elements to b

        a = b
    }

    // Use add() code in earlier slide.
}
Adding N elements to an array list

Suppose we initialize an array list to be of length 1 and, each time we try to add to a full array list, we double the length of the array.

Double length and copy element

Double length and copy elements

Double length and copy elements

add new element

add new element

add new elements

add new elements
Q: How many times $k$ do we need to double the length of the array so that it is of length $N$?

A: $2^k = N$, so $k = \log_2 N$

Q: How many copy operations are required to make an array list of length $N$?

A: $1 + 2 + 4 + 8 + \ldots + 2^{k-1} = 2^k - 1 = N - 1$
List Operations

- `get(i)`: Get the i-th element from the list.
- `set(i,e)`: Set the i-th element to e.
- `add(i,e)`: Add the element e at index i.
- `remove(i)`: Remove the i-th element from the list.
- `remove(e)`: Remove element e from the list (if it is there).
- `clear()`: Remove all elements from the list.
- `isEmpty()`: Return true if the list is empty, false otherwise.
- `size()`: Return the number of elements in the list.
remove(i)

if ( (i >= 0) and (i < size) ){
    tmp = a[i] // put aside and later return it

    for ( k = i; k < size-1; k++){
        a[ k ] = a[ k + 1 ] // shift (copy)
    }

    size = size – 1
    a[ size ] = null // clean
    return tmp
}

NOTE: The lecture ended before I had time to present this slide. Please see description in the lecture notes.