CS100J 30 October 2007

Algorithms on arrays Reading: 8.3–8.5

The searching, sorting, and other algorithms will be on the course website, along with a JUnit testing class for them.

Please punctuate this:

Dear John, I want a man who knows what love is all about you are generous kind thoughtful people who are not like you admit to being useless and inferior you have ruined me for other men I yearn for you I have no feelings whatsoever when we're apart I can be forever happy will you let me be yours

Gloria

This is a neat example of the ambiguity that English can cause, if not used properly! We try to use English properly and precisely, but ambiguity tends to creep in because of difference in cultures in which people grow up and simply because of differences of opinion. Read on!

Dear John:
I want a man who knows what love is all about. You are generous, kind, thoughtful. People who are not like you admit to being useless and inferior. You have ruined me for other men. I yearn for you. I have no feelings whatsoever when we're apart. I can be forever happy -- will you let me be yours?
Gloria

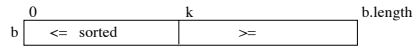
Dear John:
I want a man who knows what love is. All about you are generous, kind, thoughtful people, who are not like you. Admit to being useless and inferior. You have ruined me. For other men, I yearn. For you, I have no feelings whatsoever. When we're apart, I can be forever happy. Will you let me be?
Yours,
Gloria .

Today and Thursday

- Look at horizontal notation for writing assertions about arrays.
- **Develop** several methods that process arrays. The idea is to help you learn how to develop algorithms.
  - Write a function to tell whether two arrays are equal.
  - Write a function to copy an array.
  - Write a function to tell whether two DNA sequences are complements of each other.
  - Look at other algorithms that manipulate arrays
- Look at storing a table of values in a Java array.
  - including adding a value to the table,
  - deleting the last value of the table,
  - deleting some other value from the table.

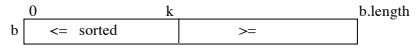
The material on tables is in Sec. 8.4 of course text.

Horizontal notation for arrays, strings, Vectors



Example of an assertion about an array b. It asserts that:

1. b[0..k–1] is sorted (i.e. its values are in ascending order)
2. Everything in b[0..k–1] is ≤ everything in b[k..b.length–1]

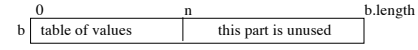


1. b[0..k] is sorted (i.e. its values are in ascending order)
2. Everything in b[0..k] is ≤ everything in b[k+1..b.length–1]

Maintain a table of values in an array

As a program is executed, it may have to maintain a table of values, say temperatures, within an array. The table will start out empty; then values will be added to it. We must say *where* in the array the values are stored.

```
int[] b= new int[5000];  // The n values in the table are in b[0..n–1]
int n= 0;                // 0 ≤ n <= 5000
```

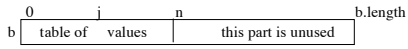



```
// Add t to the table:
b[n]= t;
n= n+1;
```

```
// Delete last element of table
// (assuming it exists).
n= n –1;
```

Deleting an element of the table doesn't require to change the array at all! It's just that a value that was in the table is now in the unused part.

Maintain a table of values in an array



// Delete value b[j] from the table.

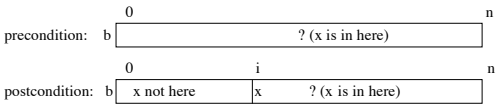
If the order of values in the table doesn't matter:

```
n= n–1;
b[j]= b[n];
```

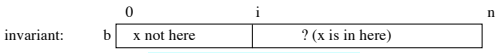
If the order of values in table does matter:

```
n= n–1;
// Move b[j+1..n] to b[j..n–1]
// inv: b[j+1..k–1] have been moved
for (int k= j+1; k <= n; k= k+1 ) {
   // Process b[k] (move it down 1 pos.)
   b[k–1]= b[k];
}
// post: b[j+1..n] have been moved
```

Find first position of x in array b. x is guaranteed to be in the array.
We use the horizontal notation to write assertions about arrays.



The invariant is found easily from the postcondition



1. Makes invariant true

```
i= 0;
while (b[i] != x) {
        i= i + 1;
}
```

2. When this is true, so is postcondition

3. Make progress toward termination

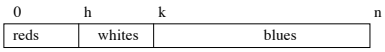
4. Invariant is true after repetend.

• Dutch national flag. Vague spec.: b[0..n-1] contains only red, white, blue balls. Sort it using only swaps.

Better spec.: Precondition: n >= 0

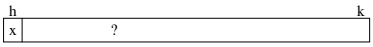
Permute b[0..n–1] to truthify:

postcondition: b[0..h–1] are red balls
b[h..k–1] are white balls
b[k..n–1] are blue balls

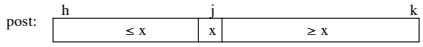


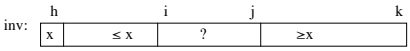
invariant :

Partition algorithm:



Permute the values of the array and store a value in j to truthify:





• Linear search. Vague spec.: find first occurrence of v in b[h..k-1].
Better spec.: Store an integer in i to truthify:
postcondition: (0) v is not in b[h..i-1]
(1) Either i= k or v = b[k]
invariant: v is not in b[h..i-1]

• Finding the min. Vague spec.: Find the min of b[h..k]
Better spec.: Precondition: h <= k (because an empty set of values has no min)
Store in i to truthify:
postcondition: b[m] is the min of b[h..k] (and it is first occurrence of the min)
invariant: b[m] is the min of b[h..t-1] (and it is first occur. of the min)

• Binary search: Vague spec: Look for v in sorted array segment b[h..k].
Better spec:
Precondition: b[h..k] is sorted (in ascending order).
Store in i to truthify:
postcondition: b[h..i] <= v and v < b[i+1..k]
invariant: b[h..i] <= v and v < b[j..k]