**CS1110 31 March 2008**

**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.

Haikus (5-7-5) seen on Japanese computer monitors

Yesterday it worked.
Today it is not working.
Windows is like that.

Serious error.
All shortcuts have disappeared.
Screen. Mind. Both are blank.

A crash reduces
Your expensive computer
To a simple stone.

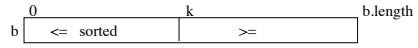
The Web site you seek
Cannot be located, but
Countless more exist.

Three things are certain:
Death, taxes, and lost data.
Guess which has occurred?

Chaos reigns within.
Reflect, repent, and reboot.
Order shall return.

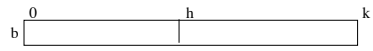
1

Horizontal notation for arrays, strings, Vectors



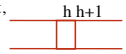

Example of an assertion about an array b. It asserts that:

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]




Given the index h of the First element of a segment and the index k of the element that Follows the segment, the number of values in the segment is k – h.

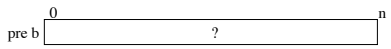
b[h .. k – 1] has k – h elements in it.

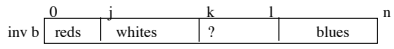



(h+1) – h = 1

2

How to make invariant look like initial condition




1. Make red, white, blue section empty: use formulas for no. of values in these sections, set j, k , l so that they have 0 elements.

2. Compare precondition with invariant. E.g. in precondition, 0 marks first unknown. In invariant, k marks first unknown. Therefore, k and 0 must be the same.

3

How to learn these algorithms
(Need to know dutch national flag and binary search for quiz).

1. Practice writing pre- and post-conditions. If we say "Dutch National Flag", you should be able to write them down.

2. Practice developing the invariant from the pre- and post-conditions.

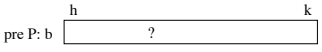
3. Practice developing the loop (with initialization), using the four loopy questions.

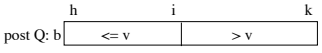
4

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

Called binary search because each iteration of the loop cuts the array segment still to be processed in half

Below, the array is in non-descending order:



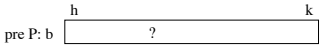

Eg. b[h..k] is (2, 2, 2, 2, 5, 7, 7, 8)

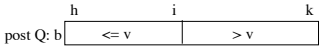
| v | i |
|---|---|
| 1 | h–1 |
| 2 | h+3 |
| 3 | h+3 |
| 9 | h+7 |

v in: i is index of rightmost occurrence
v not in: it belongs between b[i], b[i+1]

5

Binary search: b[h..k] is sorted (ascending order)




Called binary search because each iteration of the loop cuts the array segment still to be processed in half

Write down an invariant

6

**How many iterations does binary search make?**

Suppose k-h is a power of 2.

How many iterations does binary search perform?

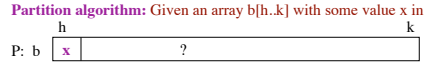
| i | t | (t–i) / 2 | so next value of t is |
|---|---|---|---|
| 0 | 16 | 8 | 8 |
| 0 | 8 | 4 | 4 |
| 0 | 4 | 2 | 2 |
| 0 | 2 | 1 | 1 |
| 0 | 1 | | |
| 0 | 2^n | 2^(n-1) | 2^(n-1) |

| p | s^p |
|---|---|
| 0 | 2^0 = 1 |
| 1 | 2^1 = 2 |
| 2 | 2^2 = 4 |
| 3 | 2^3 = 8 |
| 4 | 2^4 = 16 |
| 5 | 2^5 = 32 |
| 6 | 2^6 = 64 |
| … | … |

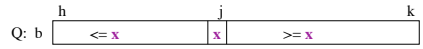
Requires ~ n iterations for an array segment of size 2^n

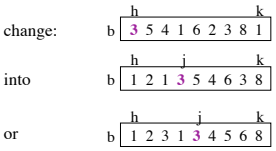
7

**Partition algorithm:** Given an array b[h..k] with some value x in b[h]:




Swap elements of b[h..k] and store in j to truthify P:



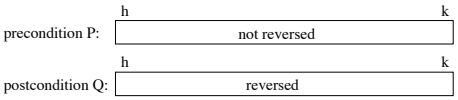

Algorithm is an important piece of the most famous sorting algorithm, quicksort

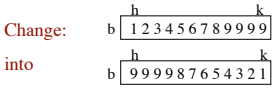
x is called the pivot value.

x is not a program variable; x just denotes the value initially in b[h].

8

**Reversal:** Reverse the elements of array segment b[h..k].




9

**Check whether two arrays are equal**

```
/** = "b and c are equal" (both null or both contain
       arrays whose elements are the same) */
public static boolean equals(int[] b, int[] c) {




}
```

10