CSE 143
Lecture 6 (b)

Binary Search

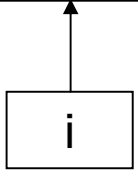
reading: 13.1

slides created by Marty Stepp
http://www.cs.washington.edu/143/

Sequential search

- **sequential search**: Locates a target value in an array/list by examining each element from start to finish.
  - How many elements will it need to examine?
  - Example: Searching the array below for the value **42**:

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	68	85	92	103



  - Notice that the array is sorted. Could we take advantage of this?

Binary search (13.1)

- **binary search**: Locates a target value in a *sorted* array/list by successively eliminating half of the array from consideration.
  - How many elements will it need to examine?
  - Example: Searching the array below for the value **42**:

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	68	85	92	103

min mid max

The `Arrays` class

- Class `Arrays` in `java.util` has many useful array methods:

Method name	Description
`binarySearch(`**array**`, `**value**`)`	returns the index of the given value in a *sorted* array (or < 0 if not found)
`binarySearch(`**array**`, `**minIndex**`, `**maxIndex**`, `**value**`)`	returns index of given value in a *sorted* array between indexes *min* / *max* - 1 (< 0 if not found)
`copyOf(`**array**`, `**length**`)`	returns a new resized copy of an array
`equals(`**array1**`, `**array2**`)`	returns `true` if the two arrays contain same elements in the same order
`fill(`**array**`, `**value**`)`	sets every element to the given value
`sort(`**array**`)`	arranges the elements into sorted order
`toString(`**array**`)`	returns a string representing the array, such as `"[10, 30, -25, 17]"`

- Syntax: `Arrays.`**methodName**`(`**parameters**`)`

Arrays.binarySearch

```
// searches an entire sorted array for a given value
// returns its index if found;  a negative number if not found
// Precondition: array is sorted
Arrays.binarySearch(array, value)

// searches given portion of a sorted array for a given value
// examines minIndex (inclusive) through maxIndex (exclusive)
// returns its index if found;  a negative number if not found
// Precondition: array is sorted
Arrays.binarySearch(array, minIndex, maxIndex, value)
```

- The `binarySearch` method in the `Arrays` class searches an array very efficiently if the array is sorted.
  - You can search the entire array, or just a range of indexes (useful for "unfilled" arrays such as the one in `ArrayIntList`)
  - If the array is not sorted, you may need to sort it first

Using `binarySearch`

```
// index    0  1  2  3   4   5   6   7   8   9  10  11  12  13  14  15
int[] a = {-4, 2, 7, 9, 15, 19, 25, 28, 30, 36, 42, 50, 56, 68, 85, 92};

int index  = Arrays.binarySearch(a, 0, 16, 42);   // index1 is 10
int index2 = Arrays.binarySearch(a, 0, 16, 21);   // index2 is -7
```

- `binarySearch` returns the index where the value is found
- if the value is *not* found, `binarySearch` returns:

  `-(insertionPoint + 1)`

  - where `insertionPoint` is the index where the element *would* have been, if it had been in the array in sorted order.
  - To insert the value into the array, negate `insertionPoint` + 1

```
int indexToInsert21 = -(index2 + 1);  // 6
```