Arrays

Arrays

- One variable that can hold multiple values.
- Limitations:
  - length must be set when declared & can't change
  - can only hold a single type
- that is…
  - can't lengthen an array mid-program
  - can't store an `int` and `String` in the same array

Declaring an Array

- Arrays are objects in Java
  - ie. require the declare reference, `new` object syntax
- To create a pointer to an array, append `[]` to the type
  - eg. for type `int`:

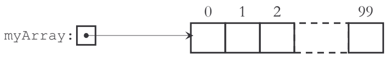
    ```
    int[] myArray;
    ```
- Like other reference declarations, this just creates reference, not object.

Array Objects

- To allocate the memory for the array:
  - eg. create array of 100 `int`s:

    ```
    new int[100]
    ```
  - All at once:

    ```
    int[] myArray = new int[100];
    ```
- Creates this:



Array Elements

- The square brackets are used to subscript.
  - eg.

    ```
    int[] myArray = new int[100];
    myArray[0] = 2;
    myArray[4] = myArray[0] + 1;
    System.out.println(myArray[4]);
      // prints 3
    myArray[99] = 2;   // ok
    myArray[100] = 2;  // error
    ```

Searching

Searching

- Common problem: find an item in an array
  - find exact match/find item that contains
  - return position/return element

Linear Search

- In general, we have to go through every element in the array.
- Pseudocode:

```
for i from 0 to length-1:
    if array[i] == target:
        return i
return -1  // not in array, -1 will indicate that.
```

- Java implementation in text

Properties

- Will work on any array
  - searches every element: will find the target if it's there.
- Slow
  - searches every element
  - might not be necessary in some arrays

Binary Search

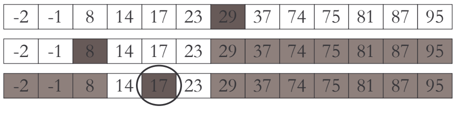
- Suppose a sorted array.
  - then, we can avoid looking at every item.
  - eg.

-2	-1	8	14	17	23	29	37	74	75	81	87	95

  - We are looking for 17 in this array.
- Half the array can quickly be eliminated:
  - Look in middle: 29
  - Can ignore second half of array.

Details

- Keep track of the "candidate" part of the array.
  - Look at the middle of the candidate part.
  - Found it: done!
  - Not found: throw away one half.
- eg.



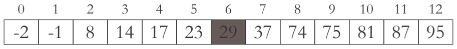
-2	-1	8	14	17	23	29	37	74	75	81	87	95
-2	-1	8	14	17	23	29	37	74	75	81	87	95
-2	-1	8	14	17	23	29	37	74	75	81	87	95

Pseudocode

```
first = 0            // start of candidate array
last = length-1  // end of candidate array
while first ≤ last:
    mid = (first + last)/2
    if        array[mid] = target:  return mid
    else if array[mid] < target:  first = mid + 1
    else if array[mid] > target:  last = mid – 1
return -1  // not in array
```

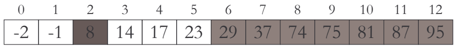
Example, again

first = 0 last = 12 mid = 6



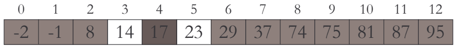
0	1	2	3	4	5	6	7	8	9	10	11	12
-2	-1	8	14	17	23	29	37	74	75	81	87	95

first = 0 last = 5 mid = 2



0	1	2	3	4	5	6	7	8	9	10	11	12
-2	-1	8	14	17	23	29	37	74	75	81	87	95

first = 3 last = 5 mid = 4



0	1	2	3	4	5	6	7	8	9	10	11	12
-2	-1	8	14	17	23	29	37	74	75	81	87	95

return 4

Speed

- binary search
  - example took 3 steps
  - worst case: 4 ($\approx \log_2 n$)
- linear search
  - worst case: 13 ($= n$)
- binary search is **much** faster for large arrays.
  - … if it can be used: only works on sorted arrays