Searching and sorting

Garfield AP CS

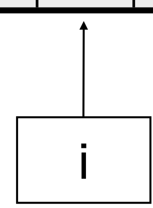
Why search and sort?

- Ever used Google?
- Ever sorted your music alphabetically?
- Most interesting programs search and/or sort
  - Baby Names
  - Shopping Cart
- The AP test will have a couple of questions on search and sort

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

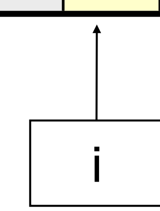


  - Notice that the array is sorted. Could we take advantage of this?

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?

Guessing game

- I’m thinking of a number between 0 and 100...
- What’s the best strategy?

Binary Search

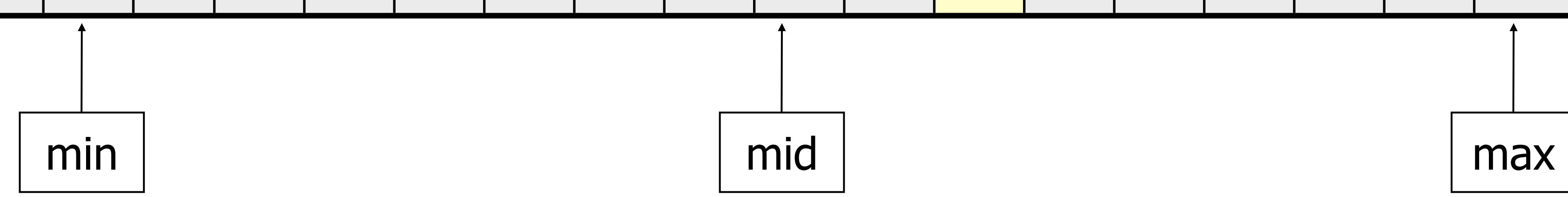
- **binary search**: Locates a target value in a sorted array/list by successively eliminating half of the array from consideration.
  - How many elements will it need to examine?
  - Example: Searching the array below for the value **42**:

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

Binary Search

- **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**:

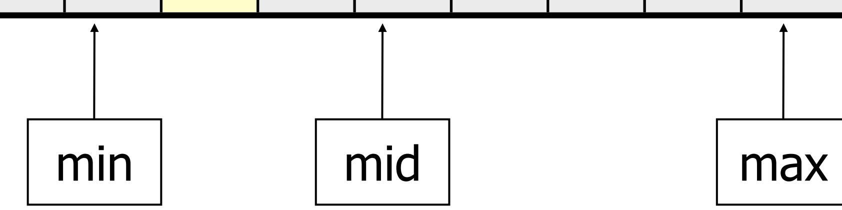
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	7	10	15	20	22	25	30	36	42	50	56	68	85	92	103



Binary Search

- **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



Binary Search

- **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

Binary Search

- **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

mid

Iterative Binary Search

```
// Returns the index of an occurrence of target in a,
// or a negative number if the target is not found.
// Precondition: elements of a are in sorted order
public static int binarySearch(int[] a, int target) {
    int min = 0;
    int max = a.length - 1;

    while (min <= max) {
        int mid = (min + max) / 2;
        if (a[mid] < target) {
            min = mid + 1;
        } else if (a[mid] > target) {
            max = mid - 1;
        } else {
            return mid;    // target found
        }
    }

    return -(min + 1);    // target not found
}
```

Divide and conquer!

- Class of algorithms
- Break down a problem into two or more sub-problems of the same (or related) type until these become simple enough to be solved directly

- Binary search just yields one sub problem so it’s not always included

- Nicely expressed recursively

Recursive Binary Search

- Write a recursive `binarySearch` method.
  - If the target value is not found, return its negative insertion point.

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

```
int index  = binarySearch(data, 42);   // 10
int index2 = binarySearch(data, 66);   // -14
```

Recursive code

```
// Returns the index of an occurrence of the given value in
// the given array, or a negative number if not found.
// Precondition: elements of a are in sorted order
public static int binarySearch(int[] a, int target) {
    return binarySearch(a, target, 0, a.length - 1);
}

// Recursive helper to implement search behavior.
private static int binarySearch(int[] a, int target,
                                int min, int max) {
    if (min > max) {
        return -1;        // target not found
    } else {
        int mid = (min + max) / 2;
        if (a[mid] < target) {         // too small; go right
            return binarySearch(a, target, mid + 1, max);
        } else if (a[mid] > target) {  // too large; go left
            return binarySearch(a, target, min, mid - 1);
        } else {
            return mid;   // target found; a[mid] == target
        }
    }
}
```

Efficiency

- **efficiency**: A measure of the use of computing resources by code.
  - can be relative to speed (time), memory (space), etc.
  - most commonly refers to run time

- Assume the following:
  - Any single Java statement takes the same amount of time to run.
  - A method call's runtime is measured by the total of the statements inside the method's body.
  - A loop's runtime, if the loop repeats N times, is N times the runtime of the statements in its body.

Efficiency example

```
for (int i = 1; i <= N; i++) {
    for (int j = 1; j <= N; j++) {
        statement1;
    }
}

for (int i = 1; i <= N; i++) {
    statement2;
    statement3;
    statement4;
    statement5;
}
```

- How many statements will execute if N = 10? If N = 1000?

Efficiency example

```
for (int i = 1; i <= N; i++) {
    for (int j = 1; j <= N; j++) {      } N²
        statement1;
    }
}

for (int i = 1; i <= N; i++) {
    statement2;
    statement3;
    statement4;
    statement5;
}
```

- How many statements will execute if N = 10? If N = 1000?

Efficiency example

```
for (int i = 1; i <= N; i++) {
    for (int j = 1; j <= N; j++) {
        statement1;
    }
}
```

N^2

```
for (int i = 1; i <= N; i++) {
    statement2;
    statement3;
    statement4;
    statement5;
}
```

4N

- How many statements will execute if N = 10? If N = 1000?

Efficiency example

```
for (int i = 1; i <= N; i++) {
    for (int j = 1; j <= N; j++) {
        statement1;
    }
}

for (int i = 1; i <= N; i++) {
    statement2;
    statement3;
    statement4;
    statement5;
}
```

First loop: N^2

Second loop: $4N$

Total: $N^2 + 4N$

- How many statements will execute if N = 10? If N = 1000?

Complexity classes

- **complexity class**: A category of algorithm efficiency based on the algorithm's relationship to the input size N.

Class	Big-Oh	If you double N, ...	Example
constant	$O(1)$	unchanged	10ms
logarithmic	$O(\log_2 N)$	increases slightly	175ms
linear	$O(N)$	doubles	3.2 sec
log-linear	$O(N \log_2 N)$	slightly more than doubles	6 sec
quadratic	$O(N^2)$	quadruples	1 min 42 sec
cubic	$O(N^3)$	multiplies by 8	55 min
...	...	...	...
exponential	$O(2^N)$	multiplies drastically	$5 * 10^{61}$ years

Binary Search Efficiency

- **binary search** successively eliminates half of the elements.
  - Algorithm: Examine the middle element of the array.
    - If it is too big, eliminate the right half of the array and repeat.
    - If it is too small, eliminate the left half of the array and repeat.
    - Else it is the value we're searching for, so stop.
  - Which indexes does the algorithm examine to find value **22**?
  - What is the runtime complexity class of binary search?

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
value	-4	-1	0	2	3	5	6	8	11	14	22	29	31	37	56

Binary Search Complexity

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.

  N, N/2, N/4, N/8, ..., 4, 2, 1

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.

  N, N/2, N/4, N/8, ..., 4, 2, 1

  – How many divisions does it take?

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.
  N, N/2, N/4, N/8, ..., 4, 2, 1
  - How many divisions does it take?
- Think of it from the other direction:
  - How many times do I have to multiply by 2 to reach N?

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.

  N, N/2, N/4, N/8, ..., 4, 2, 1

  – How many divisions does it take?

- Think of it from the other direction:
  – How many times do I have to multiply by 2 to reach N?

  1, 2, 4, 8, ..., N/4, N/2, N

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.

  N, N/2, N/4, N/8, ..., 4, 2, 1

  – How many divisions does it take?

- Think of it from the other direction:
  – How many times do I have to multiply by 2 to reach N?

  1, 2, 4, 8, ..., N/4, N/2, N

  – Call this number of multiplications "x".

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.

  N, N/2, N/4, N/8, ..., 4, 2, 1

  - How many divisions does it take?

- Think of it from the other direction:
  - How many times do I have to multiply by 2 to reach N?

    1, 2, 4, 8, ..., N/4, N/2, N
  - Call this number of multiplications "x".

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.

  N, N/2, N/4, N/8, ..., 4, 2, 1

  - How many divisions does it take?

- Think of it from the other direction:
  - How many times do I have to multiply by 2 to reach N?

    1, 2, 4, 8, ..., N/4, N/2, N
  - Call this number of multiplications "x".

    $2^x = N$

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.

  N, N/2, N/4, N/8, ..., 4, 2, 1

  – How many divisions does it take?

- Think of it from the other direction:
  – How many times do I have to multiply by 2 to reach N?

    1, 2, 4, 8, ..., N/4, N/2, N

  – Call this number of multiplications "x".

  $2^x = N$

  $\mathbf{x = \log_2 N}$

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.

  N, N/2, N/4, N/8, ..., 4, 2, 1

  - How many divisions does it take?

- Think of it from the other direction:
  - How many times do I have to multiply by 2 to reach N?

    1, 2, 4, 8, ..., N/4, N/2, N
  - Call this number of multiplications "x".

$2^x = N$

$\mathbf{x = \log_2 N}$

Binary Search Complexity

- For an array of size N, it eliminates ½ until 1 element remains.

  N, N/2, N/4, N/8, ..., 4, 2, 1

  – How many divisions does it take?

- Think of it from the other direction:
  – How many times do I have to multiply by 2 to reach N?

  1, 2, 4, 8, ..., N/4, N/2, N

  – Call this number of multiplications "x".

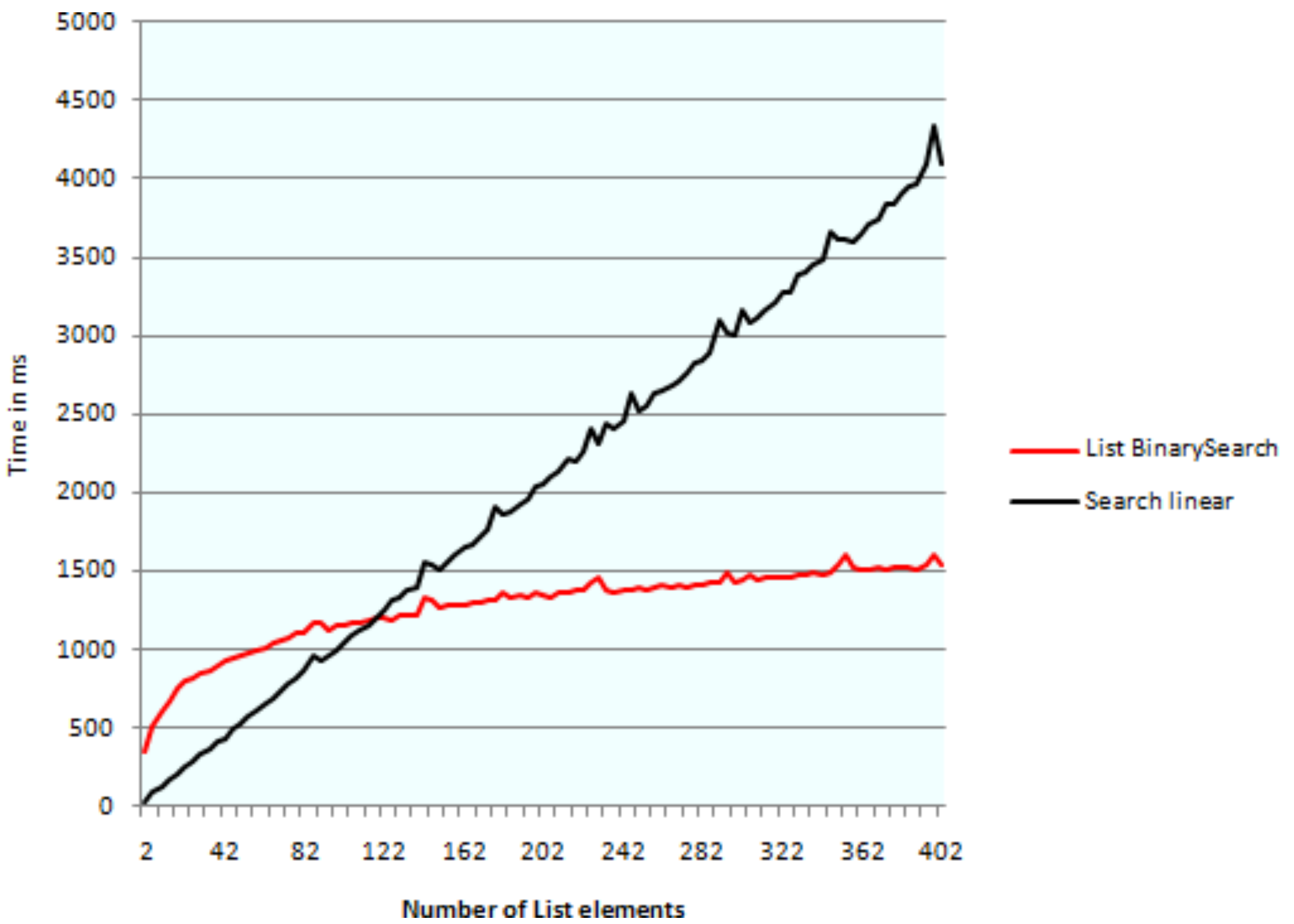
  $2^x = N$

  $x = \log_2 N$

- Binary search is in the **logarithmic** complexity class.

Time graphs

- For small data sets, linear search is faster!



Best/worst cases

- **Linear search**
  - best case: item at front and only one comparison needed
  - worst case: item not there and n comparisons needed
  - average case: item somewhere in middle and ~n/2 needed

- **Binary search**
  - best case: item in middle and only one comparison needed
  - worst case: item not there and log(n) comparisons needed
  - average case: item somewhere in middle and ~log(n)/2