

## Topic 14

# Searching and Simple Sorts

"There's nothing in your head the sorting hat can't see. So try me on and I will tell you where you ought to be."

-The Sorting Hat, *Harry Potter and the Sorcerer's Stone*



## Sorting and Searching

- ▶ Fundamental problems in computer science and programming
- ▶ Sorting done to make searching easier
- ▶ Multiple different algorithms to solve the same problem
  - How do we know which algorithm is "better"?
- ▶ Look at searching first
- ▶ Examples use arrays of ints to illustrate algorithms

CS314

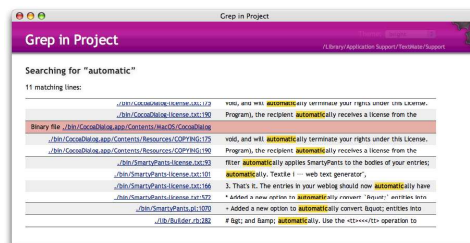
Searching and Simple Sorts

2

## Searching



recursive backtracking  
Google Search I'm Feeling Lucky



## Searching

- ▶ Given an array or list of data find the location of a particular value or report that value is not present
- ▶ linear search
  - intuitive approach?
  - start at first item
  - is it the one I am looking for?
  - if not go to next item
  - repeat until found or all items checked
- ▶ If items not sorted or unsortable this approach is necessary



CS314

Searching and Simple Sorts

4

## Linear Search

```
/* pre: data != null
   post: return the index of the first occurrence
   of target in data or -1 if target not present in
   data
*/
public int linearSearch(int[] data, int target) {
    for(int i = 0; i < data.length; i++)
        if( data[i] == target )
            return i;
    return -1;
}
```

## Linear Search, Generic

```
/* pre: data != null, no elements of data == null
   target != null
   post: return the index of the first occurrence
   of target in data or -1 if target not present in
   data
*/
public int linearSearch(Object[] data, Object target) {
    for (int i = 0; i < data.length; i++)
        if (target.equals(data[i]))
            return i;
    return -1;
}
```

T(N)? Big O? Best case, worst case, average case?

## Clicker 1

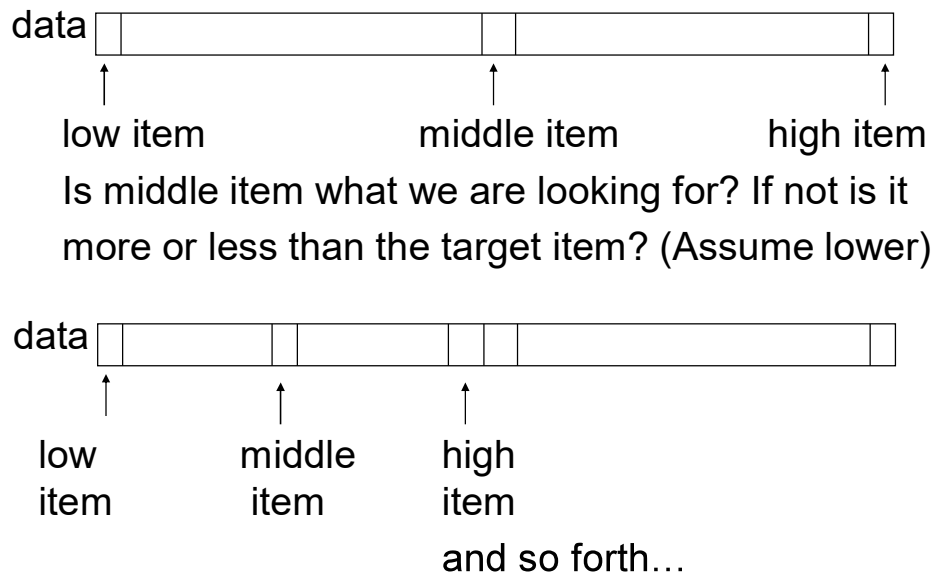
- What is the average case Big O of linear search in an array with N items, if an item is present once?

- A.  $O(1)$
- B.  $O(\log N)$
- C.  $O(N)$
- D.  $O(N \log N)$
- E.  $O(N^2)$

## Searching in a Sorted Array or List

- If items are sorted then we can *divide and conquer*
- dividing your work in half with each step
  - generally a good thing
- The Binary Search with array in ascending order
  - Start at middle of list
  - is that the item?
  - If not is it less than or greater than the item?
  - less than, move to second half of list
  - greater than, move to first half of list
  - repeat until found or sub list size = 0

## Binary Search



CS314

Searching and Simple Sorts

9

## Binary Search in Action

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53

```
public static int bsearch(int[] data, int target) {
    int indexOfTarget = -1;
    int low = 0;
    int high = data.length - 1;
    while(indexOfTarget == -1 && low <= high) {
        int mid = low + ((high - low) / 2);
        if( data[mid] == target )
            indexOfTarget = mid;
        else if( data[mid] < target)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return indexOfTarget;
}
// mid = (low + high) / 2; // may overflow!!!
// or mid = (low + high) >>> 1; using bitwise op
```

CS314

Searching and Simple Sorts

10

Trace When Key == 3  
Trace When Key == 30

Variables of Interest?

CS314

Searching and Simple Sorts

11

## Clicker 2

What is the worst case Big O of binary search in an array with N items, if an item is present?

- A.  $O(1)$
- B.  $O(\log N)$
- C.  $O(N)$
- D.  $O(N \log N)$
- E.  $O(N^2)$

CS314

Searching and Simple Sorts

12

# Generic Binary Search

```
public static <T extends Comparable<? super T>> int
    bsearch(T[] data, T target) {

    int result = -1;
    int low = 0;
    int high = data.length - 1;
    while( result == -1 && low <= high ) {
        int mid = low + ((high - low) / 2);
        int compareResult = target.compareTo(data[mid]);
        if(compareResult == 0)
            result = mid;
        else if(compareResult > 0)
            low = mid + 1;
        else
            high = mid - 1; // compareResult < 0
    }
    return result;
}
```

CS314

Searching and Simple Sorts

13

# Recursive Binary Search

```
public static int bsearch(int[] data, int target) {
    return bsearch(data, target, 0, data.length - 1);
}

public static int bsearch(int[] data, int target,
    int low, int high) {

    if(low <= high){
        int mid = low + ((high - low) / 2);
        if( data[mid] == target )
            return mid;
        else if(data[mid] > target)
            return bsearch(data, target, low, mid - 1);
        else
            return bsearch(data, target, mid + 1, high);
    }
    return -1;
}

// Clicker 3 Is this a recursive backtracking algorithm?
A. NO
B. YES
```

CS314

Searching and Simple Sorts

14

# Other Searching Algorithms

- ▶ Interpolation Search
  - more like what people really do
- ▶ Indexed Searching
- ▶ Binary Search Trees
- ▶ Hash Table Searching
- ▶ best-first
- ▶ A\*

CS314

Searching and Simple Sorts

15

# Sorting



U.S. All-time List - Marathon

As of 4/24/08

## Women

1	1	2:19:36	Deena Kastor nee Drossin
2		2:21:16	Drossin (2)
3	2	2:21:21	Joan Benoit Samuelson
4		2:21:25	Kastor (3)
5		2:22:43	Benoit (2)
6		2:24:52	Benoit (3)
7		2:26:11	Benoit (4)
8	3	2:26:26	Julie Brown
9	4	2:26:40	Kim Jones

Song Name	Time	Track #	Artist	Album
Letters from the Wasteland	4:29	1 of 10	The Wallflowers	Breach
When You're On Top	3:54	1 of 13	The Wallflowers	Red Letter Days
Hand Me Down	3:35	2 of 10	The Wallflowers	Breach
How Good It Can Get	4:11	2 of 13	The Wallflowers	Red Letter Days
Sleepwalker	3:31	3 of 10	The Wallflowers	Breach
Closer To You	3:17	3 of 13	The Wallflowers	Red Letter Days
I've Been Delivered	5:01	4 of 10	The Wallflowers	Breach
Everybody Out Of The Water	3:42	4 of 13	The Wallflowers	Red Letter Days
Witness	3:34	5 of 10	The Wallflowers	Breach
Three Ways	4:19	5 of 13	The Wallflowers	Red Letter Days
Some Flowers Bloom Dead	4:43	6 of 10	The Wallflowers	Breach
Too Late to Quit	3:54	6 of 13	The Wallflowers	Red Letter Days
Mourning Train	4:04	7 of 10	The Wallflowers	Breach
If You Never Got Sick	3:44	7 of 13	The Wallflowers	Red Letter Days
Up from Under	3:38	8 of 10	The Wallflowers	Breach
Health and Happiness	4:03	8 of 13	The Wallflowers	Red Letter Days
Murder 101	2:31	9 of 10	The Wallflowers	Breach
See You When I Get There	3:09	9 of 13	The Wallflowers	Red Letter Days
Bridge	7:42	10 of 10	The Wallflowers	Breach
Feels Like Summer Again	3:48	10 of 13	The Wallflowers	Red Letter Days
Everything I Need	3:37	11 of 13	The Wallflowers	Red Letter Days
Here in Pleasantville	3:40	12 of 13	The Wallflowers	Red Letter Days
Empire in My Mind (Bonus Track)	3:31	13 of 13	The Wallflowers	Red Letter Days

## Sorting

- A fundamental application for computation
- Done to make finding data (searching) faster
- Many different algorithms for sorting
- One of the difficulties with sorting is working with a fixed size storage container (array)
  - if resize, that is expensive (slow)
- The simple sorts are slow
  - bubble sort
  - selection sort
  - insertion sort

## Selection sort

### ‣ Algorithm

- Search through the data and find the smallest element
- swap the smallest element with the first element
- repeat starting at second element and find the second smallest element

```
public static void selectionSort(int[] data) {  
  
    for (int i = 0; i < data.length - 1; i++) {  
        int min = i;  
        for (int j = i + 1; j < data.length; j++)  
            if (data[j] < data[min])  
                min = j;  
        int temp = data[i];  
        data[i] = data[min];  
        data[min] = temp;  
    }  
}
```

## Selection Sort in Practice

44 68 191 119 119 37 83 82 191 45 158 130 76 153 39 25

What is the  $T(N)$ , *actual* number of statements executed, of the selection sort code, given an array of  $N$  elements? What is the Big  $O$ ?

## Generic Selection Sort

```
public static <T extends Comparable<? super T>>  
    void selectionSort(T[] data) {  
  
    for(int i = 0; i < data.length - 1; i++) {  
        int min = i;  
        for(int j = i + 1; j < data.length; j++)  
            if( data[min].compareTo(data[j]) > 0 )  
                min = j;  
        T temp = data[i];  
        data[i] = data[min];  
        data[min] = temp;  
    }  
}
```

## Insertion Sort

- ▶ Another of the  $O(N^2)$  sorts
- ▶ The first item is sorted
- ▶ Compare the second item to the first
  - if smaller swap
- ▶ Third item, compare to item next to it
  - need to swap
  - after swap compare again
- ▶ And so forth...

## Insertion Sort Code

```
public void insertionSort(int[] data) {  
    for(int i = 1; i < data.length; i++) {  
        int temp = data[i];  
        int j = i;  
        while(j > 0 && temp < data[j - 1]){  
            // swap elements  
            data[j] = data[j - 1];  
            data[j - 1] = temp;  
            j--;  
        }  
    }  
}
```

- ▶ Best case, worst case, average case Big O?

## Clicker 4 - Comparing Algorithms

- ▶ Which algorithm do you think has a smaller  $T(N)$  given random data, selection sort or insertion sort?

- A. Insertion Sort
- B. Selection Sort
- C. About the same