Searching
(part 2)

October 17, 2007

ComS 207: Programming I (in Java)
Iowa State University, FALL 2007
Instructor: Alexander Stoytchev

Homework #7 Hints

- **For the first 2 problems:**
  - **Instead of entering he numbers you can use the random number generator.**
- **Matrix multiplication**
- **Other Questions?**

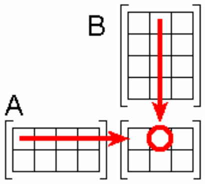
Problems with Input

```
Scanner scan = new Scanner(System.in);

String line  = scan.nextLine();
int num = Integer.parseInt(line);

String line2= scan.nextLine();
```

HW Hints: Matrix Multiplication



$$(AB)_{12} = \sum_{r=1}^{4} a_{1r}b_{r2} = a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} + a_{14}b_{42}$$

[http://en.wikipedia.org/wiki/Matrix_multiplication]

HW Hints: Use Google™

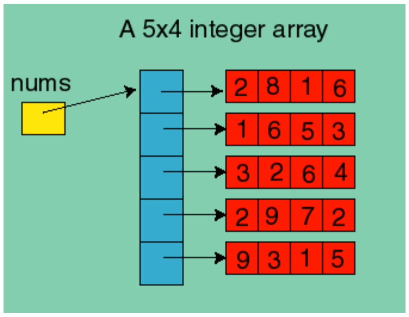
- **For example, if you don't know how to perform matrix multiplication just Google it!**

Quick review of last lecture

Arrays in Java

- Java represents 2D arrays as an array of arrays!
- In other words, a 2D integer array is really a 1D array of references to 1D integer arrays.
- The concept generalizes to N-dimensions

Anatomy of a 2D Array

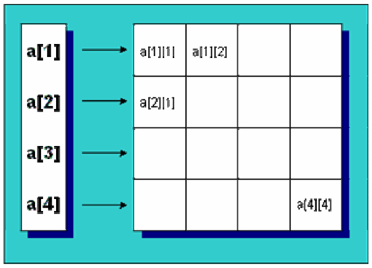


[http://www.willamette.edu/~gorr/classes/cs231/lectures/chapter9/arrays2d.htm]

Two-Dimensional Arrays

Expression	Type	Description
`table`	`int[][]`	2D array of integers, or array of integer arrays
`table[5]`	`int[]`	array of integers
`table[5][12]`	`int`	integer

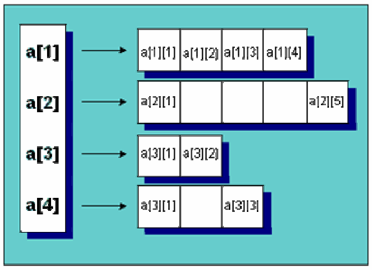
Example of a regular 2D array



Note: In Java the first index should be 0 not 1!

[http://livedocs.macromedia.com/coldfusion/5.0/Developing_ColdFusion_Applications/arrayStruct2.htm]

Example of a Ragged Array

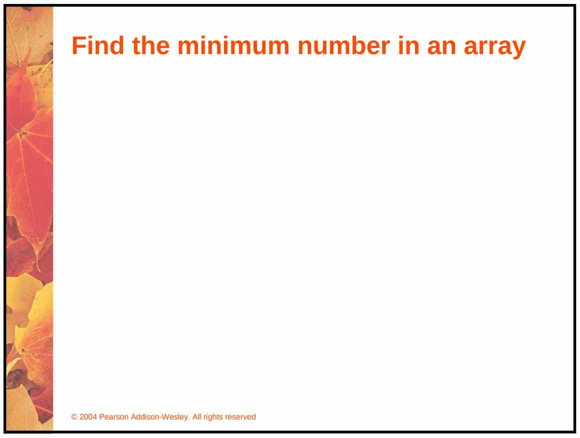


Note: In Java the first index should be 0 not 1!

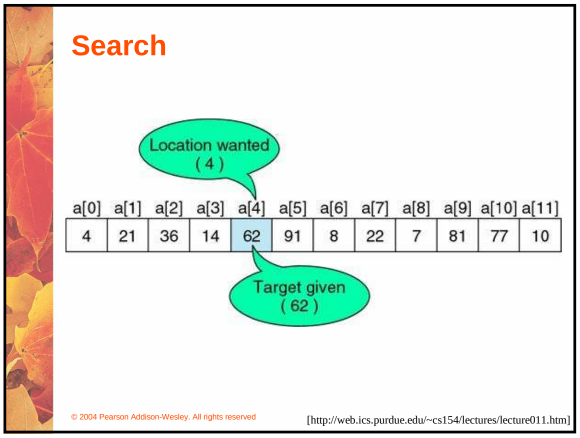
[http://livedocs.macromedia.com/coldfusion/5.0/Developing_ColdFusion_Applications/arrayStruct2.htm]

Other Stuff

- Arrays as parameters to methods



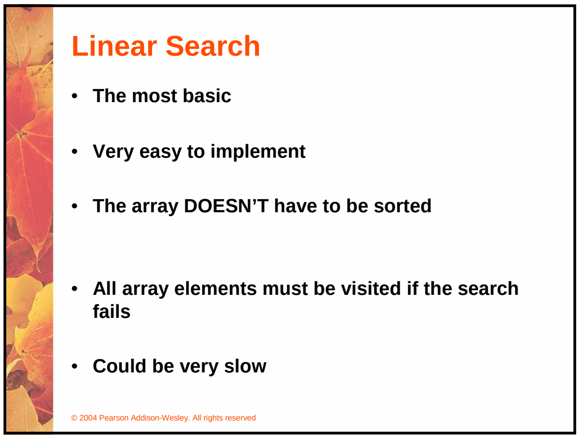
Find the minimum number in an array



Search

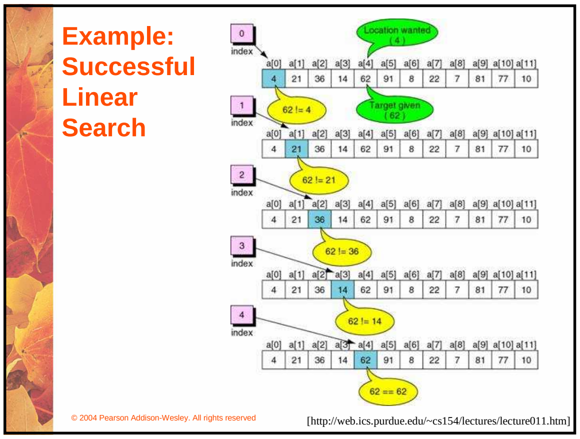
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]
4	21	36	14	62	91	8	22	7	81	77	10

[http://web.ics.purdue.edu/~cs154/lectures/lecture011.htm]



Linear Search

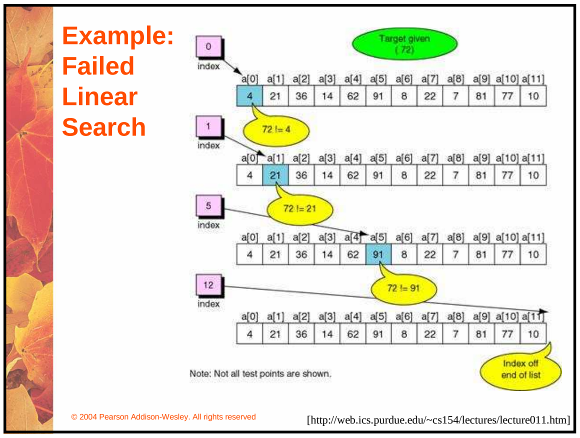
- The most basic
- Very easy to implement
- The array DOESN'T have to be sorted
- All array elements must be visited if the search fails
- Could be very slow



Example: Successful Linear Search

Location wanted
(4)
0
index
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
1
index
62 != 4
Target given
(62)
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
2
index
62 != 21
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
3
index
62 != 36
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
4
index
62 != 14
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
62 == 62

[http://web.ics.purdue.edu/~cs154/lectures/lecture011.htm]

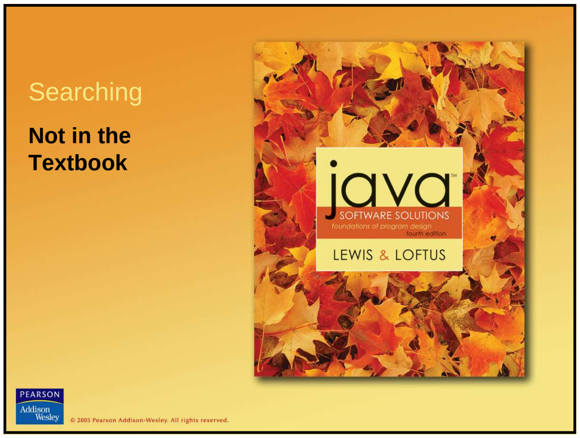


Example: Failed Linear Search

Target given
(72)
0
index
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
1
index
72 != 4
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
5
index
72 != 21
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
12
index
72 != 91
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 21 36 14 62 91 8 22 7 81 77 10
Index off
end of list

Note: Not all test points are shown.

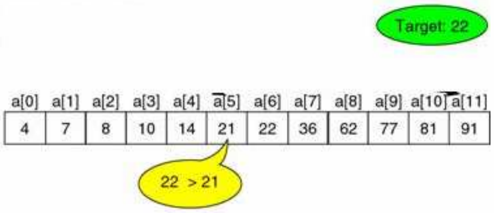
[http://web.ics.purdue.edu/~cs154/lectures/lecture011.htm]



Searching

Not in the Textbook

Java Example: Finding the index of a number in a sorted array of integers using linear search


Target: 22
a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9] a[10] a[11]
4 7 8 10 14 21 22 36 62 77 81 91
22 > 21

Example: LinearSearch_InSortedArray.java

Analysis

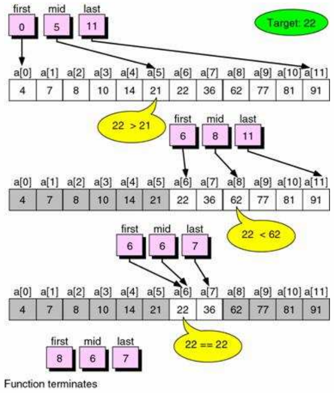
- **If the list is unsorted we have to search all numbers before we declare that the target is not present in the array.**
- **Because the list is sorted we can stop as soon as we reach a number that is greater than our target**
- **Can we do even better?**

Binary Search

- **At each step it splits the remaining array elements into two groups**
- **Therefore, it is faster than the linear search**
- **Works only on an already SORTED array**
- **Thus, there is a performance penalty for sorting the array**

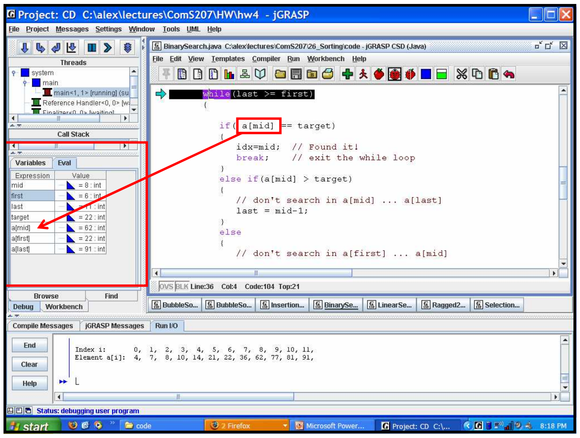
[http://web.ics.purdue.edu/~cs154/lectures/lecture011.htm]

Example: Successful Binary Search



[http://web.ics.purdue.edu/~cs154/lectures/lecture011.htm]

Example: BinarySearch.java


Project: CD C:\alex\lectures\ComS207\HW\hw4 - jGRASP
File Project Messages Settings Window Tools UML Help
Threads
system
main
main<1, 1> [running] (su
Reference Handler<0, 0> [w
Finalizer<0, 0> [waiting]
Call Stack
Variables
Eval
BinarySearch.java C:\alex\lectures\ComS207\26_Sorting\code - jGRASP CSD (Java)
File Edit View Templates Compiler Run Workbench Help
OVS BLK Line:36 Col:4 Code:104 Top:21
Browse
Find
Debug
Workbench
BubbleSo...
BubbleSo...
Insertion...
BinarySe...
LinearSe...
Ragged2...
Selection...
Compile Messages
jGRASP Messages
Run I/O
End
Clear
Help
Status: debugging user program
start
code
2 Firefox
Microsoft Power...
Project: CD C:\...
8:18 PM

Expression	Value
mid	= 8 : int
first	= 6 : int
last	= 11 : int
target	= 22 : int
a[mid]	= 62 : int
a[first]	= 22 : int
a[last]	= 91 : int

```
        while(last >= first)
        {

            if( a[mid] == target)
            {
                idx=mid;  // Found it!
                break;    // exit the while loop
            }
            else if(a[mid] > target)
            {
                // don't search in a[mid] ... a[last]
                last = mid-1;
            }
            else
            {
                // don't search in a[first] ... a[mid]
```

```
Index i:       0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11,
Element a[i]:  4,  7,  8, 10, 14, 21, 22, 36, 62, 77, 81, 91,
```

Analysis of Searching Methods

- For an array of size n

- Sequential Search (Average-Case) n/2
- Sequential Search (Worst-Case) n

- Binary Search (Average-Case) log(n)/2
- Binary Search (Worst-Case) log(n)

THE END