Searching (part 2)
October 17, 2007

Homework #7 Hints

• For the first 2 problems:
  • Instead of entering the 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

\[
(AB)_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}
\]

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

A 5x4 integer array

Example of a regular 2D array

Example of a Ragged Array

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

Other Stuff

- Arrays as parameters to methods
Find the minimum number in an array

Search

**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

Example:
Failed Linear Search

Searching

Not in the Textbook
Java Example: Finding the index of a number in a sorted array of integers using linear search

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.

Example: Successful Binary Search
Example: BinarySearch.java

Analysis of Searching Methods

- For an array of size $n$
  - Sequential Search (Average-Case) $\frac{n}{2}$
  - Sequential Search (Worst-Case) $n$
  - Binary Search (Average-Case) $\frac{\log(n)}{2}$
  - Binary Search (Worst-Case) $\log(n)$