Strings

initialize via assignment vs. constructor

length()

indexOf()

charAt()

endsWith()

equals()

toUpperCase(), toLowerCase()

vauleOf() (int, float, double)

substring()

Plain arays and the "array discipline"

initialized values pcked tight to front

must have a count variable indicating number of elements in the array

count -1 is the index where the last initialized elem is located

count represents index where next element should be placed

limitations of arrays ( overflow etc)

basic array operations:

add, remove, addToEnd, removeFromEnd

Searching & sorting

sequential/linear search runs in O(N)

binary search runs in O(log2N)

bubbleSort runs in O(N squared)

selectionSort runs in O(N squared)

insertInOrder runs in O(N squared)

Algorithmic Complexities in increasing (worse) order:

Big Oh: O(N) notation describes work required in terms of input size N

O(1) a.k.a. constant time: A constant time operation is one that can be performed in a constant

number of computer instruction that is the same regardless of the size of the data set. Assume you have

an array of ints and you want to retrieve the value of the i'th element in that array. This can be

accomplished by the syntax arr[i] and it will take the same amount of work/time to access that value

whether the array has five elements of 2 billion elements. The compiler computes the address of that

i'th element by using the address in the variable arr and adding i \* the size of the element type in

the array. This bit of arithmetic does not take any less or more time t o calculate just because the

numbers may be bigger or smaller. Thus is a constant time operation.

O(log2 n) a.k.a. logarithmic time. Assume you have a sorted array of int and you wish to search for a

specific number. Because the array is sorted you can employ the binary search technique. Go to the

middle of the array and compare the number you land on against the number you are looking for. If the

number you are looking for is less than the number in the middle you can discard the entire upper

half of the array and repeat your search process on the lower half. If you land on a number that is

less than the one you are looking for then you can discard the lower half of the array and repeat

your search on the upper half. By doing so you are eliminating half of the array every time you compare

your target number against the one you land on. Since an array on length N can only be halved log2N times,

you will either find or not find your number in at most log2N comparisons.

O(n) a.k.a. linear time. Assume you have a UN-sorted array of ints and you wish to search for a specific number.

Because the array is unsorted you would employ a front to back scan of the array and compare every number

in the array to the one you are looking for. Thus you are required to compare all <b>n</b> numbers in the

array to the one you are looking for. If the length of the array doubles then the number of required

comparisons double. The amount of work is directly proportional ( linear) the size of the input.

O(n squared) a.k.a. squared time or quadratic time. </strong> Assume you

have an UN-sorted array of int and you wish to sort that array using al algorithm like bubblesort. The

bubble sort algorithm requires 1 + 2 +3 +4 +5 + ... + n-1 comparisons and swaps to sort the array. The

sum of the series from 1 to n == (n(n+1)) / 2 which has n squared (n\*\*2) as its highest term. We use

only the highest term to characterize our complexity and thus describe bubble sort as an n squared complexity

operation. This same logic applies to all polynomial expressions such as n to the third or fourth powers etc.

which describe generally as <b>polynomial time

A nested for loop on N is typically N squared time. A triple nested for loop on N would be N cubed time, and so on.

O(2 to the n) a.k.a. exponential time. Assume you have an array of ints and you wish to print out all the

subsets of that array. Discrete math tells us that any set of n elements has 2 to the n elements in its

power set (set of all possible subsets of a given set). Thus in order to list those subsets we must do at

least 2 to the n print statements.

O(n!) a.k.a. factorial Assume you have a String and you wish to print out all the possible permeations

(arrangements) of the letters. Discrete math tells us that any set of n elements has n! permutations. Thus

in order to list those permutations we must do at least n! print statments.

There are algorithms with worse than exponential time complexity

2D Arrays

definitions and initialization

row major indexing

fundamental operations

escaping a maze

Pseudo Random Numbers: The Random class

how to declare a Random variable

seeding the Random object

nextInt() with and w/out a modulus

nextBoolean()

generating random numbers within a specified range

Wrapper Classes

Integer and .parseInt() method

Double and .parseDouble() method

same for Float, Boolean etc.

ArrayList:

Maintains count for you every time you .add() / .remove()

etc. Count gotten by calling: .size()

Automatically allocates space for 10 elements by default.

Automatically doubles its capacity when full.

Enforces array discipline by keeping all values packed to front of array

-vs- Plain array

User maintains count.

User must specify initial capacity. Once specified can never change.

User must watch for array filling up and take action to avoid overflow.

Does not enforce array discipline and thus permits storage of elements noncontiguous to the front.

HashSet:

Uses hashing algorithm to assign a unique array index to every unique element

Hashing operation take O(1) time regardless of number of elements currently in the hashSet.

operations: .add .remove .contains

TreeSet:

Uses a binary search tree to store values and keeps them in sorted order.

Insert, search and remove are all O(log2N) operations. Another advantage over array or ArrayList

is that memoery is allocated incrementally as needed and there is now resizing operation.

Very efficient when memory is "shredded".

HashMap:

Stores a key an an associated value with the key. Keys must be unique, values need not be.

operations: .put .remove .containsKey .containsValue

Classes & OOP

The following concepts are the core of our coverage of classes and object oriented programming

Members: can be data or methods, can be public or private (we don't care about other types in CS 007)

Constructors: Provide means of initialization at time of construction. Usually overloaded.

If you don't write a constructor, Java gives you a default one that sets all values to default.

If you write any constructor then you get no default constructor.

A constructor that takes values to initialize all the class data is called a full constructor.

Where practical all other constructors should call on the full constructor (keyword this)

Acessors & mutators a.k.a. setters and getters

Generally data members should be private.

To provide access, public getters and/or setters must be provided.

If a class has no public setters for any members then that class is immutable

Inheritence: uses keyword extend to define a subclass of a parent type. The new (child) class

shares the IS-A relationship with the parent.

Polymophism: Assuming a given type A and any type B that has been derived from A. Any context where

an A type is expected, a B type may appear instead. Any sub type C derived from B is also an A type.

Interfaces: Any class may impliement an interface. Doing so requires the implimenting class to write

whatever method(s) is/are required by the interface.

Algorithmic Complexities in increasing (worse) order:

big Oh notation describes work requires proportionate to the input size N

O(1) a.k.a. constant time: A constant time operation is one that can be performed in a constant number of computer instruction that is the same regardless of the size of the data set.

O(log2 N) a.k.a. logarithmic time. (binary search)

O(N) a.k.a. linear time. (sequential search)

O(N squared) a.k.a. squared time. (bubble sort,selection sort, insertion sort)

O(2 to the N) a.k.a. exponential time (all subsets)

O(N!) a.k.a. factorial times. ( all permutations)

Recursion and Backtracking:

Anything written with a loop can be written using recursion

The inverse is not always true

A recursive method calls itself

3 components of a recursive method: Base case, modification of input or serach space, & recursive call

recursion should not be used in cases where a loop can be written, due to the excessive memory

overhead on the call stack.

Call stack records the history or methods calls starting with main.

Only the top most method is executing at any time.

We used recursion and backtracking in the Swamp and boggle assignments.

You should be able to answer questions about the programming assignments.