NC STATE UNIVERSITY

Program Complexity

Slides are modified from those provided by Marty Stepp
www.buildingjavaprograms.com

NC STATE UNIVERSITY

Abstract Data Types (ADTs)

- Abstract Data Type (ADT): specification of a collection or data and the operations that can be performed on the data
  - Describes *what* a collection does
  - Not *how* the collection does it

- Lists are linear data structures
  - Standard operations: add, remove, get, set, …
  - A client uses the abstraction of the list without needing to know the details about the internal workings
    - Most of the time

NC STATE UNIVERSITY

Tradeoffs with Data Structures

- So why are the multiple types of lists? Do we really need both an ArrayList and a LinkedList?
- There are tradeoffs:
  - How the data are stored internally
  - How the operations are implemented
- Implement a data structure to meet an interface
  - Client code uses the interface
  - Implementation can be customized for the characteristics of your data

4

NC STATE UNIVERSITY

Runtime Efficiency

- **Efficiency**: a measure of the use of computing resources
  - can be relative to speed (time), memory (space), etc.
- 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.

5

Efficiency Example

```
statement1;
statement2;
statement3;

for (int i = 1; i <= N; i++) {
    statement4;
}

for (int i = 1; i <= N; i++) {
    statement5;
    statement6;
    statement7;
}
```

3

N

3N

4N + 3

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;
}
```

N^2

4N

$N^2 + 4N$

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

NC STATE UNIVERSITY

Algorithmic Growth Rates

- We measure runtime in proportion to the input data size, N.
  - **Growth rate**: change in runtime as N changes.
- Say an algorithm runs $\mathbf{N^3 + 25N^2 + 8N + 17}$ statements.
  - Consider the runtime when N is *extremely large* .
  - We ignore constants like 25 because they are tiny next to N.
  - The highest-order term (N^3) dominates the overall runtime.
- We say that this algorithm runs "on the order of" N^3.
  - Or $\mathbf{O(N^3)}$ for short ("**Big-Oh** of N cubed")

NC STATE UNIVERSITY

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

Collection Efficiency

- What is the efficiency of various operations with different internal data structures?

Method	ArrayList	LinkedList
`add(value)`		
`add(index, value)`		
`indexOf`		
`get`		
`remove`		
`set`		
`size`		

Considerations with Complexity

- The Big-Oh notation only tells part of the story
  - The growth rate is asymptotically **less than or equal to** the given growth rate
- There are operations that may be more efficient
  - Adding/removing to/from the ***end*** of an array list is O(1)
  - Adding/removing to/from the ***front*** of a linked list is O(1)
  - Adding additional references to a linked list can increase efficiency
- We can create specialized data structures that take advantage of these efficiencies.