Performance Measurement

Performance Analysis

Paper and pencil.

Don't need a working computer program or even a computer.

Some Uses Of Performance Analysis

- determine practicality of algorithm
- predict run time on large instance
- compare 2 algorithms that have different asymptotic complexity
  - e.g., O(n) and $O(n^2)$

Limitations of Analysis

Doesn't account for constant factors.

but constant factor may dominate

$1000n$ vs n^2

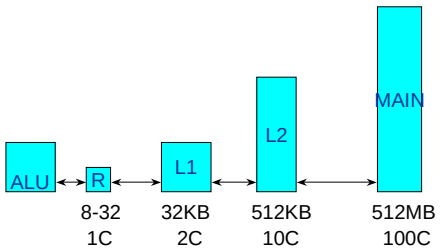
and we are interested only in

$n < 1000$

Limitations of Analysis

Modern computers have a hierarchical memory organization with different access time for memory at different levels of the hierarchy.

Memory Hierarchy



Limitations of Analysis

Our analysis doesn't account for this difference in memory access times.

Programs that do more work may take less time than those that do less work.

Performance Measurement

Measure actual time on an actual computer.

What do we need?

Performance Measurement Needs

- programming language
- working program
- computer
- compiler and options to use
  javac -o

Performance Measurement Needs

- data to use for measurement
  worst-case data
  best-case data
  average-case data
- timing mechanism --- clock

Timing In Java

```
long startTime = System.currentTimeMillis();
// gives time in milliseconds since 1/1/1970 GMT

// code to be timed comes here

long elapsedTime = System.currentTimeMillis()
                                - startTime;
```

Shortcoming

Clock accuracy
  assume 100 milliseconds

Repeat work many times to bring total time to be >= 1 second

Accurate Timing

```
long startTime = System.currentTimeMillis();
long counter;
do {
        counter++;
        doSomething();
    } while (System.currentTimeMillis() -
                startTime < 1000)
long elapsedTime = System.currentTimeMillis()
                            - startTime;
float timeForMethod =
                ((float) elapsedTime)/counter;
```

Accuracy

Now accuracy is 10%.

first reading may be just about to change to startTime + 100

second reading may have just changed to finishTime

so finishTime - startTime is off by 100ms

Accuracy

first reading may have just changed to startTime

second reading may be about to change to finishTime + 100

so finishTime - startTime is off by 100ms

Accuracy

Examining remaining cases, we get

trueElapsedTime =
finishTime - startTime +- 100ms

To ensure 10% accuracy, require

elapsedTime = finishTime – startTime
>= 1sec

What Went Wrong?

```
long startTime = System.currentTimeMillis();
long counter;
do {
        counter++;
        InsertionSort.insertionSort(a);
    } while (System.currentTimeMillis() -
                startTime < 1000)
long elapsedTime = System.currentTimeMillis()
                            - startTime;
float timeForMethod =
                ((float) elapsedTime)/counter;
```

The Fix

```
long startTime = System.currentTimeMillis();
long counter;
do {
        counter++;
        // put code to initialize a[] here
        InsertionSort.insertionSort(a);
    } while (System.currentTimeMillis() -
                startTime < 1000)
```

Time Shared System

UNIX

time MyProgram

Bad Way To Time

```
do {
        counter++;
        startTime = System.currentTimeMillis();
        doSomething();
         elapsedTime +=
              System.currentTimeMillis()
                 - startTime;
    } while (elapsedTime < 1000)
```