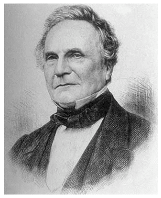
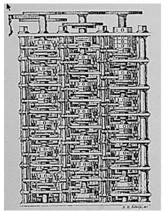
4.1 - 4.2 Analysis of Algorithms

Running Time

As soon as an Analytic Engine exists, it will necessarily guide the future course of the science. Whenever any result is sought by its aid, the question will arise - By what course of calculation can these results be arrived at by the machine in the shortest time? - Charles Babbage



Charles Babbage (1864)



Analytic Engine (schematic)

Overview

Analysis of algorithms. Framework for comparing algorithms and predicting performance.

Scientific method.
- Observe some feature of the universe.
- Hypothesize a model that is consistent with observation.
- Predict events using the hypothesis.
- Verify the predictions by making further observations.
- Validate the theory by repeating the previous steps until the hypothesis agrees with the observations.

Universe = computer itself.

Algorithmic Successes

N-body Simulation.
- Simulate gravitational interactions among N bodies.
- Brute force: N^2 steps.
- Barnes-Hut: N log N steps, enables new research.

Andrew Appel
PU '81

Discrete Fourier transform.
- Break down waveform of N samples into periodic components. Applications: DVD, JPEG, MRI, astrophysics, ….
- Brute force: N^2 steps.
- FFT algorithm: N log N steps, enables new technology.

Freidrich Gauss
1805

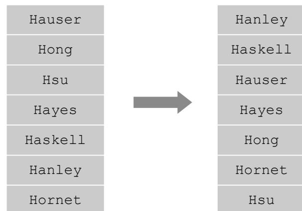
Sorting.
- Rearrange N items in ascending order.
- Fundamental information processing abstraction.

Jon von Neumann
IAS 1945

Case Study: Sorting

Sorting problem. Rearrange N items into ascending order.

Applications. Statistics, databases, data compression, computational biology, computer graphics, scientific computing, ...



Insertion Sort

Insertion Sort

Insertion sort.

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

Insertion Sort: Java Implementation

```java
public class Insertion {
   private static boolean less(double x, double y) {
      return (x < y);
   }

   private static void exch(double[] a, int i, int j) {
      double swap = a[i];
      a[i] = a[j];
      a[j] = swap;
   }                                              sorting helper functions

   public static void sort(double[] a) {
      for (int i = 0; i < a.length; i++) {
         for (int j = i; j > 0; j--) {
            if (less(a[j], a[j-1]))
               exch(a, j, j-1);
            else break;
         }
      }
   }                                              sorting algorithm
}
```

Insertion Sort: Observation

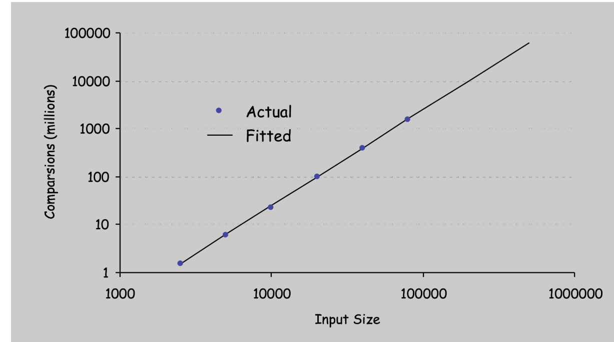
Observe and tabulate running time for various values of N.

- Data source: N random numbers between 0 and 1.
- Machine: Apple G5 1.8GHz with 1.5GB memory running OS X.
- Timing: Skagen wristwatch.

N	Comparisons	Time
5,000	6.2 million	0.13 seconds
10,000	25 million	0.43 seconds
20,000	99 million	1.5 seconds
40,000	400 million	5.6 seconds
80,000	1600 million	23 seconds

Insertion Sort: Experimental Hypothesis

Data analysis. Plot # comparisons vs. input size on log-log scale.



slope

Regression. Fit line through data points $\approx a N^b$.

Hypothesis. # comparisons grows quadratically with input size $\approx N^2/4$.

Insertion Sort: Prediction and Verification

Experimental hypothesis. # comparisons $\approx N^2/4$.

Prediction. 400 million comparisons for N = 40,000.

Observations.

N	Comparisons	Time
40,000	401.3 million	5.595 sec
40,000	399.7 million	5.573 sec
40,000	401.6 million	5.648 sec
40,000	400.0 million	5.632 sec

Agrees.

Prediction. 10 billion comparisons for N = 200,000.

Observation.

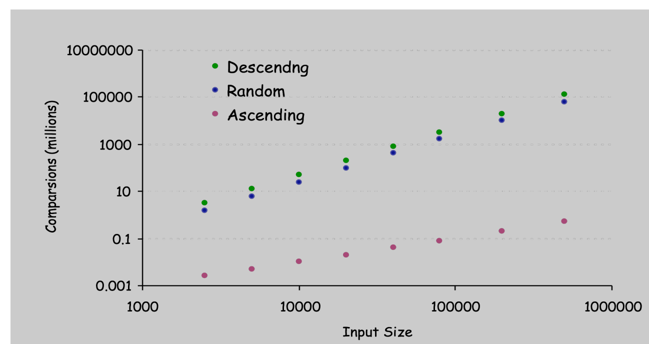
N	Comparisons	Time
200,000	9.997 billion	145 seconds

Agrees.

Insertion Sort: Validation

Number of comparisons depends on input family.

- Descending: $N^2/2$.
- Random: $N^2/4$.
- Ascending: N.



Insertion Sort: Theoretical Hypothesis

Experimental hypothesis.
- Measure running times, plot, and fit curve.
- Model useful for predicting, but not for explaining.

Theoretical hypothesis.
- Analyze algorithm to estimate # comparisons as a function of:
  - number of elements N to sort
  - average or worst case input
- Model useful for predicting and explaining.

Critical difference. Theoretical model is independent of a particular machine or compiler; applies to machines not yet built.

Insertion Sort: Analysis

Worst case. (descending)
- Iteration i requires i comparisons.
- Total = (0 + 1 + 2 + ... + N-1) ≈ N^2 / 2 compares.

E	F	G	H	I	J	D	C	B	A
						i			

Average case. (random)
- Iteration i requires i/2 comparisons on average.
- Total = (0 + 1 + 2 + ... + N-1) / 2 ≈ N^2 / 4 compares

A	C	D	F	H	J	E	B	I	G
						i			

Insertion Sort: Theoretical Hypothesis

Theoretical hypothesis.

Analysis	Comparisons	Stddev
Worst	N^2 / 2	-
Average	N^2 / 4	1/6 $N^{3/2}$
Best	N	-

Validation. Theory agrees with observations.

N	Actual	Predicted
200,000	9.9997 billion	10.000 billion

Insertion Sort: Lesson

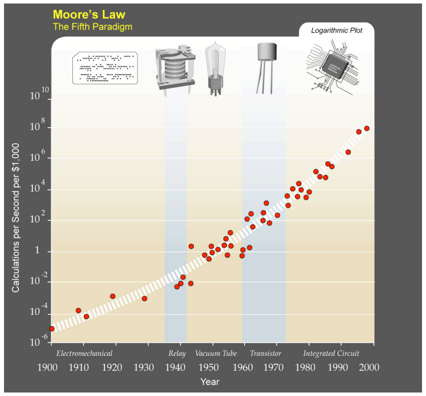
Lesson. Supercomputer can't rescue a bad algorithm.

Computer	Comparisons Per Second	Thousand	Million	Billion
laptop	10^7	instant	1 day	3 centuries
super	10^{12}	instant	1 second	2 weeks

Moore's Law

Moore's law. Transistor density on a chip doubles every 2 years.

Variants. Memory, disk space, bandwidth, computing power per $.



http://en.wikipedia.org/wiki/Moore's_law

Moore's Law and Algorithms

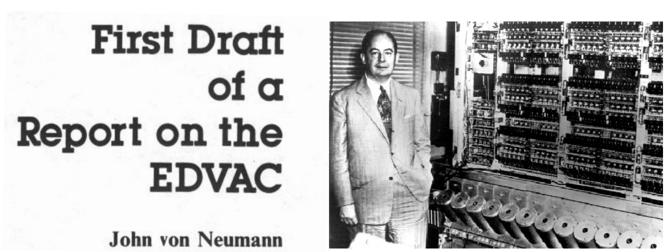
Quadratic algorithms do not scale with technology.

- New computer may be 10x as fast.
- But, has 10x as much memory so problem may be 10x bigger.
- With quadratic algorithm, takes 10x as long!

Lesson. Need linear algorithm to keep pace with Moore's law.

> Software inefficiency can always outpace Moore's Law.
> Moore's Law isn't a match for our bad coding. - Jaron Lanier

Mergesort



Mergesort

Mergesort.

- Divide array into two halves.
- Recursively sort each half.
- Merge two halves to make sorted whole.

input

M E R G E S O R T E X A M P L E

sort left

E E G M O R R S T E X A M P L E

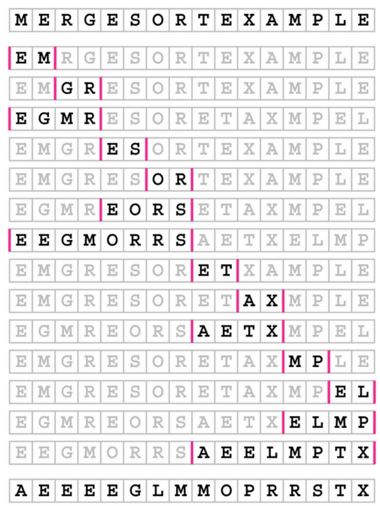
sort right

E E G M O R R S A E E L M P T X

merge

A E E E E G L M M O P R R S T X

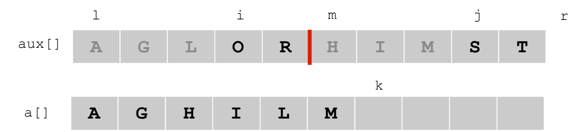
Mergesort: Example



Merging

Merging. Combine two pre-sorted lists into a sorted whole.

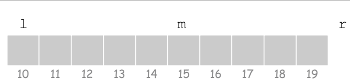
How to merge efficiently? Use an auxiliary array.




```java
private static void merge(double[] a, double[] aux, int l, int m, int r) {
   for (int k = l; k < r; k++) aux[k] = a[k];
   int i = l, j = m;
   for (int k = l; k < r; k++) {
      if      (i >= m)                a[k] = aux[j++];
      else if (j >= r)                a[k] = aux[i++];
      else if (less(aux[j], aux[i])) a[k] = aux[j++];
      else                            a[k] = aux[i++];
   }
}
```

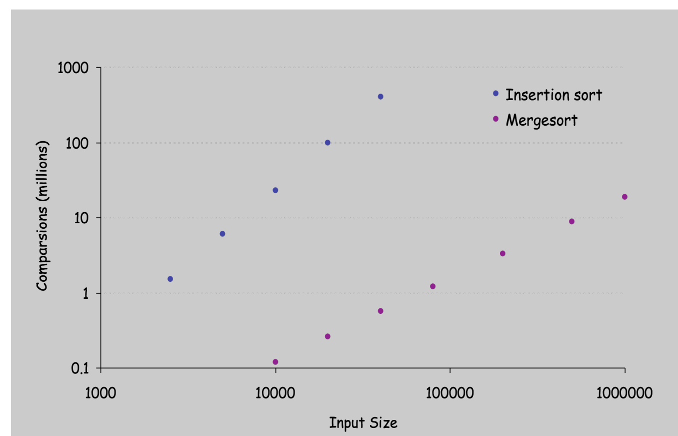

Mergesort: Java Implementation

```java
public class Merge {
   private static boolean less(double x, double y)
      // as before

   private static void merge(double[] a, double[] aux, int l, int m, int r) {
      // see previous slide

   private static void sort(double[] a, double[] aux, int l, int r) {
      if (r <= l + 1) return;
      int m = l + (r - l) / 2;
      sort(a, aux, l, m);
      sort(a, aux, m, r);
      merge(a, aux, l, m, r);
   }

   public static void sort(double[] a) {
      double[] aux = new double[a.length];
      sort(a, aux, 0, a.length);
   }
}
```



Mergesort: Preliminary Hypothesis

Experimental hypothesis. Number of comparisons ≈ 20N.



Mergesort: Prediction and Verification

Experimental hypothesis. Number of comparisons ≈ 20N.

Prediction. 80 million comparisons for N = 4 million.

Observations.

N	Comparisons	Time
4 million	82.7 million	3.13 sec
4 million	82.7 million	3.25 sec
4 million	82.7 million	3.22 sec

Agrees.

Prediction. 400 million comparisons for N = 20 million.

Observations.

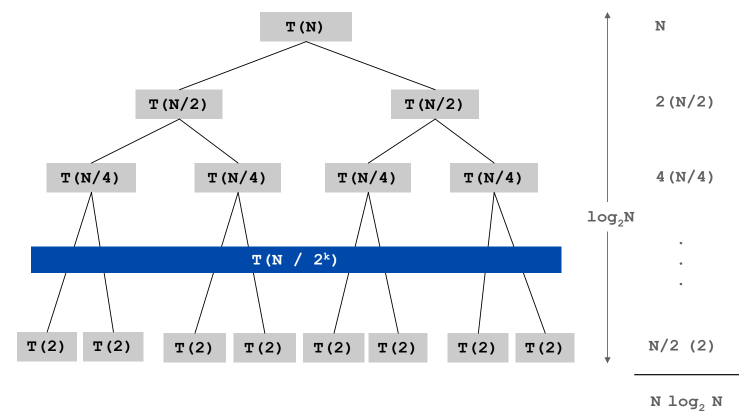
N	Comparisons	Time
20 million	460 million	17.5 sec
50 million	1216 million	45.9 sec

Not quite.

Mergesort: Analysis

Analysis. To mergesort array of size N, mergesort two subarrays of size N/2, and merge them together using ≤ N comparisons.

we assume N is a power of 2



Mergesort: Theoretical Hypothesis

Theoretical hypothesis.

Analysis	Comparisons
Worst	$N \log_2 N$
Average	$N \log_2 N$
Best	$1/2\ N \log_2 N$

Validation. Theory now agrees with observations.

N	Actual	Predicted
10,000	120 thousand	133 thousand
20 million	460 million	485 million
50 million	1,216 million	1,279 million

Mergesort: Lesson

Lesson. Great algorithms can be more powerful than supercomputers.

Computer	Comparisons Per Second	Insertion	Mergesort
laptop	10^7	3 centuries	3 hours
super	10^{12}	2 weeks	instant

N = 1 billion

Binary Search

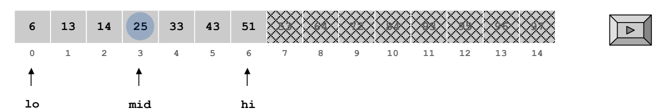
Searching a Sorted Array

Searching a sorted array. Given a sorted array, determine the index associated with a given key.

Ex. Dictionary, phone book, book index, credit card numbers.

Binary search.

- Examine the middle key.
- If it matches, return its index.
- Otherwise, search either the left or right half.



Binary Search: Java Implementation

Invariant. Algorithm maintains `a[lo]` ≤ `key` ≤ `a[hi]`.

```
public static int binarySearch(double[] a, double key) {
   int lo = 0;
   int hi = N-1;
   while (lo <= hi) {
      int m = lo + (hi - lo) / 2;
      if (key == a[m])) return m;
      if (key  < a[m])) hi = m - 1;
      else              lo = m + 1;
   }
   return -1;
}
```

Java library implementation. See `Arrays.binarySearch()`.

Binary Search: Analysis

Analysis. To binary search in an array of N elements, need to do 1 comparison and binary search in an array of N/2 elements.

N → N/2 → N/4 → N/8 → ... → 1

Q. How many times can you divide a number by 2 until you reach 1?

A. $\log_2 N$.

Scientific Method

Scientific method applies to estimate running time.

- Experimental analysis: not difficult to perform experiments.
- Theoretical analysis: may require advanced mathematics.
- Small subset of mathematical functions suffice to describe running time of many fundamental algorithms.

$\log_2 N$

```
while (N > 1) {
   N = N / 2;
   ...
}
```

N

```
for (int i = 0; i < N; i++)
   ...
```

N^2

```
for (int i = 0; i < N; i++)
   for (int j = 0; j < N; j++)
      ...
```

$N \log_2 N$

```
public static void g(int N) {
   if (N == 0) return;
   g(N/2);
   g(N/2);
   for (int i = 0; i < N; i++)
      ...
}
```

2^N

```
public static void f(int N) {
   if (N == 0) return;
   f(N-1);
   f(N-1);
   ...
}
```


Order of Growth Classifications

Order of growth.

- Estimate running time as a function of input size N.
- Ignore lower order terms.
  - when N is large, terms are negligible
  - when N is small, we don't care
- Ex: $6N^3 + 17N^2 + 56 \sim 6N^3$.

Donald Knuth
Turing award '74

Function	Description	When N doubles, running time
1	Constant algorithm is independent of input size.	does not change
log N	Logarithmic algorithm gets slightly slower as N grows.	increases by a constant
N	Linear algorithm is optimal for processing N inputs.	doubles
N log N	Linearithmic algorithm scales to huge N.	slightly more than doubles
N^2	Quadratic algorithm is impractical for large N.	quadruples
2^N	Exponential algorithm is not usually practical.	squares!

Summary

How can I evaluate the performance of my algorithm?

- Computational experiments.
- Theoretical analysis.

What if it's not fast enough?

- Understand why.
- Buy a faster computer.
- Find a better algorithm in a textbook.
- Discover a new algorithm.

Attribute	Better Machine	Better Algorithm
Cost	$$$ or more.	$ or less.
Applicability	Makes "everything" run faster.	Does not apply to some problems.
Improvement	Quantitative improvements.	Dramatic qualitative improvements possible.