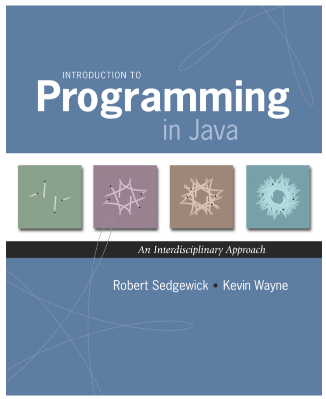
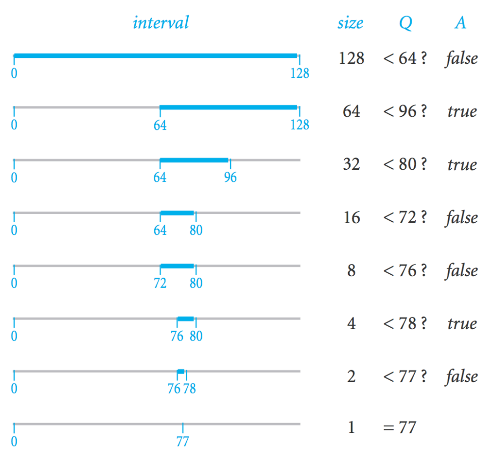
4.2 Sorting and Searching



Binary Search

Twenty Questions

Intuition. Find a hidden integer.



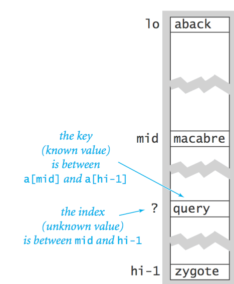
Searching a Sorted Array

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 in an array (one step)

Binary Search: Java Implementation

Invariant. Algorithm maintains `a[lo] ≤ key ≤ a[hi-1]`.

```
public static int search(String key, String[] a) {
   return search(key, a, 0, a.length);
}

public static int search(String key, String[] a, int lo, int hi) {
   if (hi <= lo) return -1;
   int mid = lo + (hi - lo) / 2;
   int cmp = a[mid].compareTo(key);
   if      (cmp > 0) return search(key, a, lo, mid);
   else if (cmp < 0) return search(key, a, mid+1, hi);
   else              return mid;
}
```

Java library implementation: `Arrays.binarySearch()`

Exception Filter

Exception filter. Read a sorted list of strings from a whitelist file, then print out all strings from standard input not in the whitelist.

```
public static void main(String[] args) {
   In in = new In(args[0]);
   String s = in.readAll();
   String[] words = s.split("\\s+");
   while (!StdIn.isEmpty()) {
      String key = StdIn.readString();
      if (search(key, words) == -1)
         StdOut.println(key);
   }
}
```

```
more test.txt
bob@office
carl@beach
marvin@spam
bob@office
bob@office
mallory@spam
dave@boat
eve@airport
alice@home
```

```
% more whitelist.txt
alice@home
bob@office
carl@beach
dave@boat

% java BinarySearch whitelist.txt < test.txt
marvin@spam
mallory@spam
eve@airport
```


Binary Search: Mathematical Analysis

Analysis. To binary search in an array of size N: do one comparison, then binary search in an array of size $N / 2$.

$$N \to N/2 \to N/4 \to N/8 \to \ldots \to 1$$

Q. How many times can you divide a number by 2 until you reach 1?
A. $\log_2 N$.

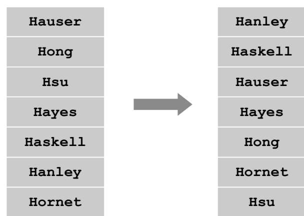
$$1$$
$$2 \to 1$$
$$4 \to 2 \to 1$$
$$8 \to 4 \to 2 \to 1$$
$$16 \to 8 \to 4 \to 2 \to 1$$
$$32 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$64 \to 32 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$128 \to 64 \to 32 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$256 \to 128 \to 64 \to 32 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$512 \to 256 \to 128 \to 64 \to 32 \to 16 \to 8 \to 4 \to 2 \to 1$$
$$1024 \to 512 \to 256 \to 128 \to 64 \to 32 \to 16 \to 8 \to 4 \to 2 \to 1$$

Sorting

Case Study: Sorting

Sorting problem. Rearrange N items in ascending order.

Applications. Statistics, databases, data compression, bioinformatics, 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.

i	j	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
6	6	and	had	him	his	was	you	the	but
6	5	and	had	him	his	was	the	you	but
6	4	and	had	him	his	the	was	you	but
		and	had	him	his	the	was	you	but

Inserting `a[6]` into position by exchanging with larger entries to its left

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.

i	j	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
		was	had	him	and	you	his	the	but
1	0	had	was	him	and	you	his	the	but
2	1	had	him	was	and	you	his	the	but
3	0	and	had	him	was	you	his	the	but
4	4	and	had	him	was	you	his	the	but
5	3	and	had	him	his	was	you	the	but
6	4	and	had	him	his	the	was	you	but
7	1	and	but	had	him	his	the	was	you
		and	but	had	him	his	the	was	you

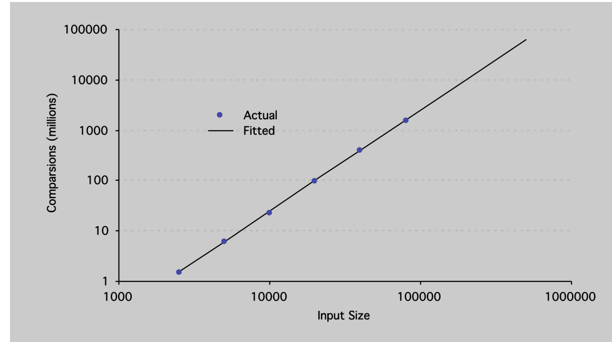
Inserting `a[1]` through `a[N-1]` into position (insertion sort)

Insertion Sort: Java Implementation

```
public class Insertion {

   public static void sort(String[] a) {
      int N = a.length;
      for (int i = 1; i < N; i++)
         for (int j = i; j > 0; j--)
            if (a[j-1].compareTo(a[j]) > 0)
               exch(a, j-1, j);
            else break;
   }

   private static void exch(String[] a, int i, int j) {
      String swap = a[i];
      a[i] = a[j];
      a[j] = swap;
   }
}
```


Insertion Sort: Empirical Analysis

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



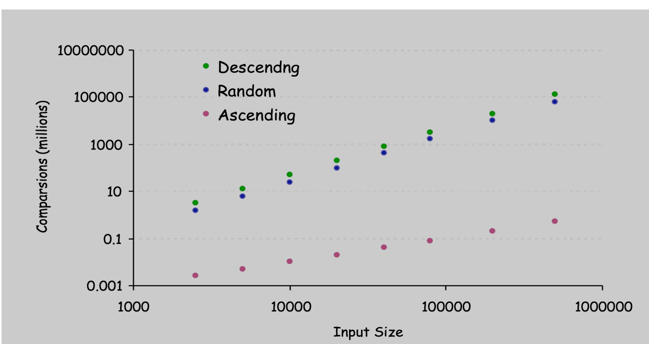
slope

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

Insertion Sort: Empirical Analysis

Observation. Number of comparisons depends on input family.

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



Insertion Sort: Mathematical Analysis

Worst case. [descending]

- Iteration i requires i comparisons.
- Total = $(0 + 1 + 2 + ... + N-1) \sim N^2 / 2$ compares.

E	F	G	H	I	J	D	C	B	A
						i			

Average case. [random]

- Iteration i requires $i / 2$ comparisons on average.
- Total = $(0 + 1 + 2 + ... + N-1) / 2 \sim N^2 / 4$ compares

A	C	D	F	H	J	E	B	I	G
						i			

Insertion Sort: Lesson

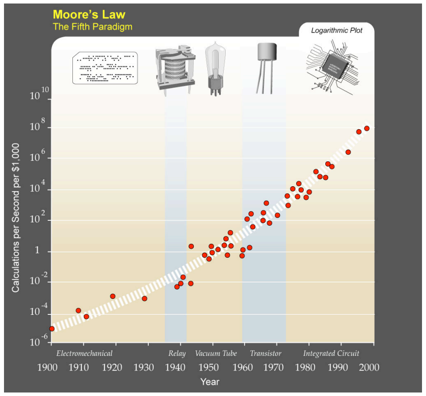
Lesson. Supercomputer can't rescue a bad algorithm.

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

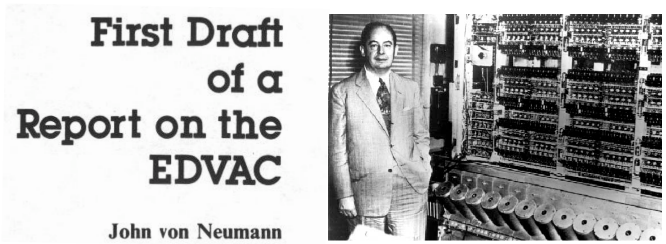
Quadratic algorithms do not scale with technology.

- 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!

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

Lesson. Need linear (or linearithmic) algorithm to keep pace with Moore's law.

Mergesort



Mergesort

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

input
was had him and you his the but

sort left
and had him was you his the but

sort right
and had him was but his the you

merge
and but had him his the was you

Mergesort: Example

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

Merging

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

How to merge efficiently? Use an auxiliary array.

i	j	k	aux[k]	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
				and	had	him	was	but	his	the	you
0	4	0	and	and	had	him	was	but	his	the	you
1	4	1	but	and	had	him	was	but	his	the	you
1	5	2	had	and	had	him	but	but	his	the	you
2	5	3	him	and	had	him	but	but	his	the	you
3	5	4	his	and	had	him	was	but	his	the	you
3	6	5	the	and	had	him	was	but	his	the	you
3	6	6	was	and	had	him	was	but	his	the	you
4	7	7	you	and	had	him	but	but	his	the	you

Merging

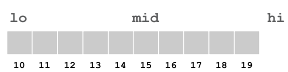
Merging. Combine two pre-sorted lists into a sorted whole.

How to merge efficiently? Use an auxiliary array.

```
String[] aux = new String[N];
// merge into auxiliary array
int i = lo, j = mid;
for (int k = 0; k < N; k++) {
   if      (i == mid)  aux[k] = a[j++];
   else if (j == hi)   aux[k] = a[i++];
   else if (a[j].compareTo(a[i]) < 0)  aux[k] = a[j++];
   else                                aux[k] = a[i++];
}

// copy back
for (int k = 0; k < N; k++)
   a[lo + k] = aux[k];
}
```

Mergesort: Java Implementation

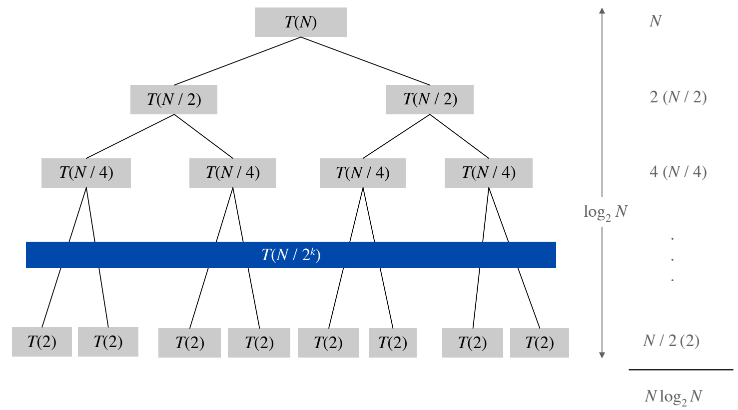
```java
public class Merge {

   public static void sort(String[] a) {
      sort(a, 0, a.length);
   }

   // Sort a[lo, hi).
   public static void sort(String[] a, int lo, int hi) {
      int N = hi - lo;
      if (N <= 1) return;

      // recursively sort left and right halves
      int mid = lo + N/2;
      sort(a, lo, mid);
      sort(a, mid, hi);

      // merge (see previous slide)
   }

}
```



Mergesort: Mathematical Analysis

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

we assume N is a power of 2



Mergesort: Mathematical Analysis

Mathematical analysis.

analysis	comparisons
worst	$N \log_2 N$
average	$N \log_2 N$
best	$1/2\ N \log_2 N$

Validation. Theory 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

Longest Repeated Substring

Redundancy Detector

Longest repeated substring. Given a string, find the longest substring that appears at least twice.

a	a	c	a	a	g	t	t	t	a	c	a	a	g	c

Brute force.

- Try all indices i and j for start of possible match.
- Compute longest common prefix for each pair (quadratic).

a	a	c	a	a	g	t	t	t	a	c	a	a	g	c
	i									j				

Applications. Bioinformatics, Burrows-Wheeler transform, …

Longest Repeated Substring: A Sorting Solution

input string

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
a a c a a g t t t a c a a g c
```

form suffixes

suffixes

```
 0  a a c a a g t t t a c a a g c
 1  a c a a g t t t a c a a g c
 2  c a a g t t t a c a a g c
 3  a a g t t t a c a a g c
 4  a g t t t a c a a g c
 5  g t t t a c a a g c
 6  t t t a c a a g c
 7  t t a c a a g c
 8  t a c a a g c
 9  a c a a g c
10  c a a g c
11  a a g c
12  a g c
13  g c
14  c
```

sort suffixes to bring repeated substrings together

sorted suffixes

```
 0  a a c a a g t t t a c a a g c
11  a a g c
 3  a a g t t t a c a a g c
 9  a c a a g c
 1  a c a a g t t t a c a a g c
12  a g c
 4  a g t t t a c a a g c
14  c
10  c a a g c
 2  c a a g t t t a c a a g c
13  g c
 5  g t t t a c a a g c
 8  t a c a a g c
 7  t t a c a a g c
 6  t t t a c a a g c
```

compute longest prefix between adjacent suffixes

longest repeated substring

```
  1               9
a a c a a g t t t a c a a g c
```


Longest Repeated Substring: Java Implementation

Suffix sorting implementation.

```
int N = s.length();
String[] suffixes = new String[N];
for (int i = 0; i < N; i++)
   suffixes[i] = s.substring(i, N);
Arrays.sort(suffixes);
```

Longest repeated substring. Search only adjacent suffixes.

```
String lrs = "";
for (int i = 0; i < N-1; i++) {
   String x = lcp(suffixes[i], suffixes[i+1]);
   if (x.length() > lrs.length()) lrs = x;
}
```

LCP. Find the longest string that is a prefix of both s and t.

Ex. `lcp("acaagtttac", "acaagc") = "acaag"`.

Longest Repeated Substring: Empirical Analysis

Input File	Characters	Brute	Suffix Sort	Length
LRS.java	2,162	0.6 sec	0.14 sec	73
Amendments	18,369	37 sec	0.25 sec	216
Aesop's Fables	191,945	3958 sec	1.0 sec	58
Moby Dick	1.2 million	43 hours †	7.6 sec	79
Bible	4.0 million	20 days †	34 sec	11
Chromosome 11	7.1 million	2 months †	61 sec	12,567
Pi	10 million	4 months †	84 sec	14

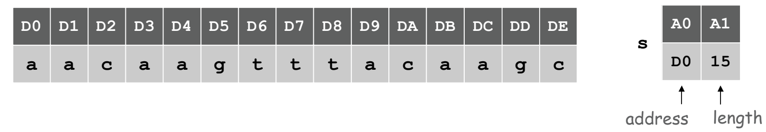
† estimated

Lesson. Sorting to the rescue; enables new research.

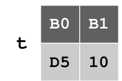
OOP Context for Strings

Possible memory representation of a string.

- `s = "aacaagtttacaagc";`



- `t = s.substring(5, 15);`



- No characters are copied when you invoke `substring()` method!

Consequences.

- Calling `substring()` takes constant time (instead of linear).
- Creating suffixes takes linear space (instead of quadratic).

Summary

Binary search. Efficient algorithm to search a sorted array.

Mergesort. Efficient algorithm to sort an array.

Applications. Many many applications involve sorting and searching.