4.2 Sorting and Searching

Sequential Search: Java Implementation

Scan through array, looking for key.

- search hit: return array index
- search miss: return -1

```java
public static int search(String key, String[] a)
{
   for (int i = 0; i < a.length; i++)
      if ( a[i].compareTo(key) == 0 ) return i;
   return -1;
}
```


Search Client: Exception Filter

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

```java
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
```


TEQ on Searching 1

A credit card company needs to whitelist 10 million customer accounts, processing 1000 transactions per second.
Using sequential search, what kind of computer is needed?

A. Toaster
B. Cellphone
C. Your laptop
D. Supercomputer
E. Google server farm

TEQ on Searching 1

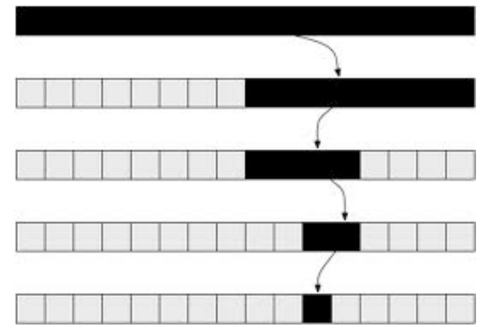
A credit card company needs to whitelist 10 million customer accounts, processing 1000 transactions per second.
Using sequential search, what kind of computer is needed?

A. Toaster
B. Cellphone
C. Your laptop
D. Supercomputer
E. Google server farm

D. or E.

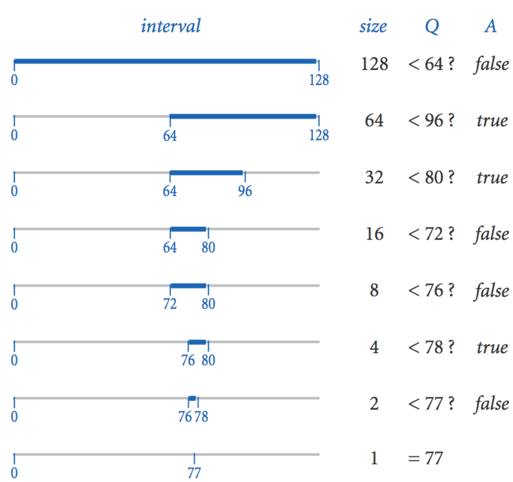
- BOE rule of thumb for any computer: (need enough memory for 10M accounts)
  N bytes in memory, ~N memory accesses per second.
- sequential search touches about half the memory
- 2 transactions per second, 500 seconds for 1000 transactions
- fix 1: Increase memory (and speed) by factor of 1000 (supercomputer)
- fix 2: Increase number of processors by factor of 1000 (server farm)
- fix 3: Use a better algorithm (stay tuned)

Binary Search



Twenty Questions

Intuition. Find a hidden integer.



Binary Search

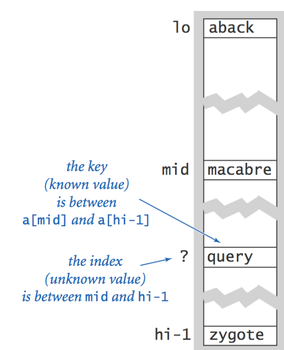
Idea:

- Sort the array (stay tuned)
- Play "20 questions" to 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 a sorted 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()`

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 $N / 2$.

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

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

A. $\log_2 N$.

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

TEQ on Searching 2

A credit card company needs to whitelist 10 million customer accounts, processing 1 thousand transactions per second.
Using binary search, what kind of computer is needed?

A. Toaster
B. Cellphone
C. Your laptop
D. Supercomputer
E. Google server farm

Sorting

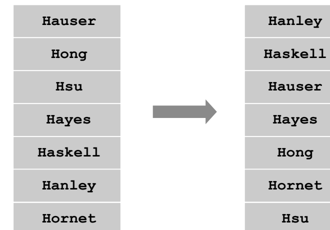
TEQ on Sorting 0

Q. What's the fastest way to sort 1 million 32-bit integers?

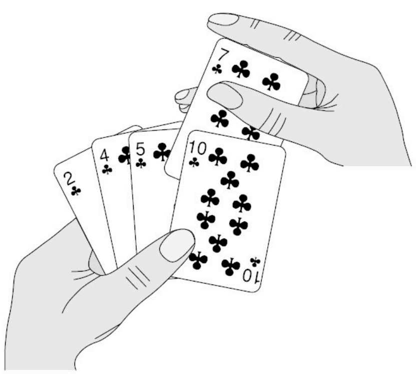
Sorting

Sorting problem. Rearrange N items in ascending order.

Applications. Binary search, statistics, databases, data compression, bioinformatics, computer graphics, scientific computing, (too numerous to list) ...



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)*

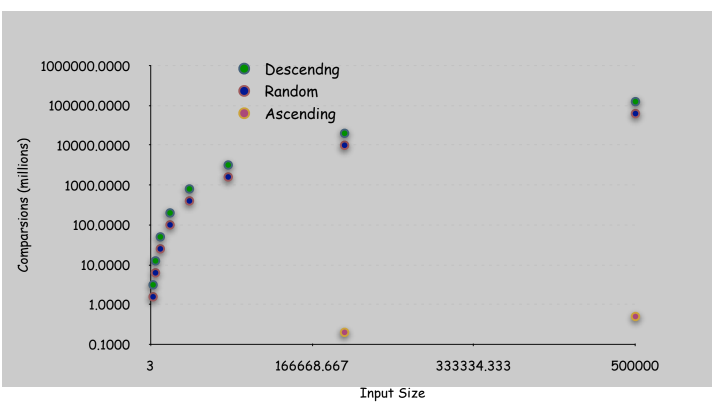
Insertion Sort: Java Implementation

```
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

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 = $(0 + 1 + 2 + ... + N\text{-}1) \sim N^2/2$ compares.

E	F	G	H	I	J	D	C	B	A
						i			

Average case. [random]

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

A	C	D	F	H	J	E	B	I	G
						i			

Insertion Sort: Scientific Analysis

Hypothesis: Running time is ~ a N^b seconds

Initial experiments:

N	Comparisons	Time	Ratio
5,000	6.2 million	0.13 seconds	
10,000	25 million	0.43 seconds	3.3
20,000	99 million	1.5 seconds	3.5
40,000	400 million	5.6 seconds	3.7
80,000	1600 million	23 seconds	4.1

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

Doubling hypothesis:

- b = lg 4 = 2, so running time is ~ a N^2
- checks with math analysis
- a ≈ 23 / 80000^2 = 3.5×10^{-9}

Refined hypothesis: Running time is ≈ $3.5 \times 10^{-9} N^2$ seconds

Insertion Sort: Scientific Analysis (continued)

Refined hypothesis: Running time is ≈ $3.5 \times 10^{-9} N^2$ seconds

Prediction: Running time for N = 200,000
should be $3.5 \times 10^{-9} \times 4 \times 10^{10} \approx 140$ seconds

Observation:

N	Time
200,000	145 seconds

Observation matches prediction and validates refined hypothesis.

TEQ on Sorting 1

A credit card company uses insertion sort to sort 10 million customer account numbers, for use in whitelisting with binary search. What kind of computer is needed?

A. Toaster
B. Cellphone
C. Your laptop
D. Supercomputer
E. Google server farm

Insertion Sort: Lesson

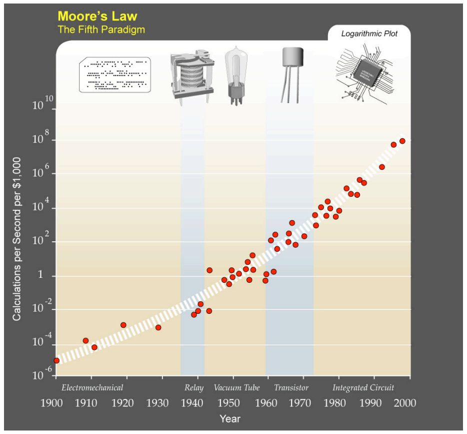
Lesson. Supercomputer can't rescue a bad algorithm.

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

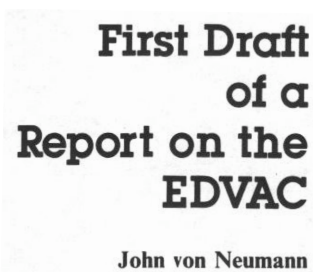
Moore's Law and Algorithms

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!

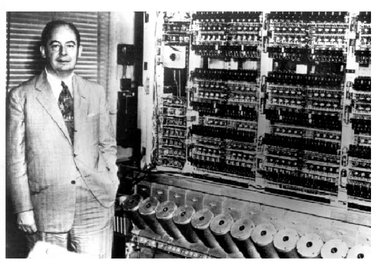
"Software inefficiency can always outpace Moore's Law. Moore's Law isn't a match for our bad coding." – Jaron Lanier

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

Mergesort


First Draft of a Report on the EDVAC

John von Neumann



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	E	T	A	X	M	P	E	L
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	1	2	3	4	5	6	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	was	but	his	the	you
2	5	3	him	and	had	him	was	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	was	but	his	the	you

Trace of the merge of the sorted left half with the sorted right half

Merging

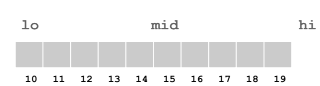
Merging. Combine two pre-sorted lists into a sorted whole.

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

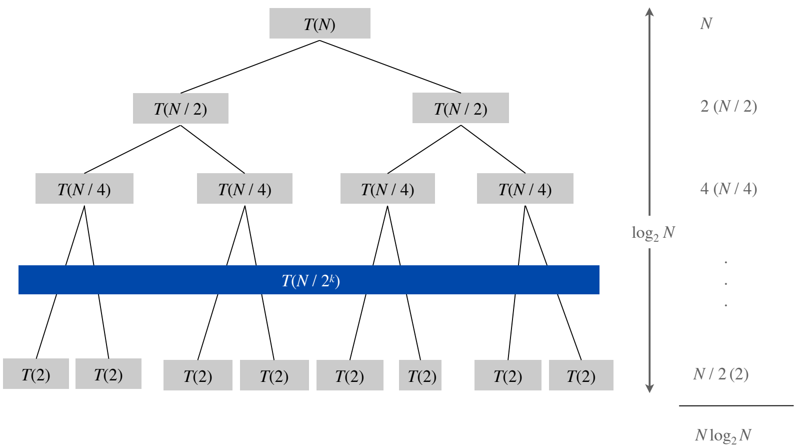
```
public class Merge
{
   public static void sort(String[] a)
   {  sort(a, 0, a.length);  }

   // 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 sorted halves (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: Scientific Analysis

Hypothesis. Running time is a N lg N seconds

Initial experiments:

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

- $a \approx 3.2 / (4 \times 10^6 \times 32) = 2.5 \times 10^{-8}$

Refined hypothesis. Running time is 2.5×10^{-7} N lg N seconds.

Prediction: Running time for N = 20,000,000 should be about $2.5 \times 10^{-8} \times 2 \times 10^7 \times 35 \approx 17.5$ seconds

Observation:

N	Time
20 million	17.5 sec

Observation matches prediction and validates refined hypothesis.

TEQ on Sorting 2

A credit card company uses mergesort to sort 10 million customer account numbers, for use in whitelisting with binary search. What kind of computer is needed?

A. Toaster
B. Cellphone
C. Your laptop
D. Supercomputer
E. Google server farm

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, cryptography, ...

LRS application: patterns in music

Music is characterized by its repetitive structure

Mary Had a Little Lamb

Fur Elise

source: http://www.bewitched.com/match/

LRS applications: patterns in sequences

Repeated sequences in real-world data are causal.

Ex 1. Digits of pi
- Q. are they "random"?
- A. No, but we can't tell the difference
- Ex. Length of LRS in first 10 million digits is 14

Ex 2. Cryptography
- Find LRS
- Check for "known" message header identifying place, date, person, etc.
- Break code

Ex 3. DNA
- Find LRS
- Look somewhere else for causal mechanisms
- Ex. Chromosome 11 has 7.1 million nucleotides

Brute-force solution

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 (LCP) for each pair

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

Analysis.
- all pairs: $1 + 2 + ... + N \sim N^2/2$ calls on LCP
- too slow for long strings

Longest Repeated Substring: A Sorting Solution

input string

```
0 1 2 3 4 5 6 7 8 91011121314
a a c a a g t t t a c a a g c
```

1. 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
```

2. 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
```

3. 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 common prefix: `lcp(s, t)`.
- longest string that is a prefix of both s and t
- Ex: `lcp("acaagtttac", "acaagc") = "acaag"`.
- easy to implement (you could write this one).

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;
}
```

Java substring operation

Memory representation of strings.

```
s = "aacaagtttacaagc";
```

D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE
a	a	c	a	a	g	t	t	t	a	c	a	a	g	c

s

A0	A1
D0	15
↑ address	↑ length

- A `String` is an address and a length.
- Characters can be shared among strings.
- `substring()` computes address, length (instead of copying chars).

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

t

B0	B1
D5	10

Consequences.

- `substring()` is a constant-time operation (instead of linear).
- Creating suffixes takes linear space (instead of quadratic).
- Running time of LRS is dominated by the string sort.

TEQ on Sorting 3

Q. Four researchers A, B, C and D are looking for long repeated subsequences in a genome with over 1 billion characters.
A. has a grad student do it.
B. uses brute force (check all pairs) solution.
C. uses sorting solution with insertion sort.
D. uses sorting solution with mergesort.

Which one is more likely to find a cancer cure?

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.

Many, many, many other things enabled by fast sort and search!

Summary

Binary search. Efficient algorithm to search a sorted array.

Mergesort. Efficient algorithm to sort an array.

Applications. Many, many, many things are enabled by fast sort and search.