13. Sorting and Searching

A typical client: Whitelist filter

- **WhiteList filter**
  - Read a list of strings from a **whitelist** file.
  - Read strings from StdIn and write to StdOut only those in the whitelist.

**Example. Email spam filter** (message contents omitted)

<table>
<thead>
<tr>
<th><strong>whiteList</strong></th>
<th><strong>stdin</strong></th>
<th><strong>stdout</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>alice@home</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>bob@office</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>carl@beach</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>dave@boat</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Examples: Overdrawn account

- Spamers

Examples: Account in good standing

- Friends and relatives

Search client: Whitelist filter

```java
public class WhiteFilter {
    public static int search(String key, String[] a)
    {
        // Search method (stay tuned).
        public static void main(String[] args)
        {
            In in = new In(args[0]);
            String[] words = in.readAllStrings();
            while (!Stdin.isEmpty())
            {
                String key = Stdin.readString();
                if (search(key, words) != -1)
                    Stdout.println(key);
            }
        }
    }
}
```

% more whitext.txt
alice@home
bob@office
carl@beach
dave@boat
% more test.txt
bob@office
carl@beach
marvin@spam
bob@office
mallory@spam
dave@boat
eve@airport
alice@home
% java WhiteFilter whitext.txt < test.txt
alice@home
Hey, Alice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.

Hey, rAlice. I think I’m going to start an internet company.

Me too. I’m thinking about having 1 thousand customers next month and 1 million next year.

We’re hoping to grow even faster than that.

Good luck!

BTW, you’re going to need a whitelist filter.

Yes, I know. I’m going to a hackathon to knock it out.

I’m going to take a few CS courses first.
Random representative inputs for searching and sorting

Generate N random strings of length L from a given alphabet

```java
class Generator {
    public static String randomString(int length, String alphabet) {
        char[] a = new char[length];
        for (int i = 0; i < length; ++i) {
            int t = StdRandom.uniform(alphabet.length());
            a[i] = alphabet.charAt(t);
        }
        return new String(a);
    }
}
```

Random strings of length 10:
- abcd
- efgh
- ijkl
- mnop

Random strings of length 6:
- abcdef
- ghijkl
- mnoqrs
- tuvxyz

Empirical tests of sequential search

Whitelist filter scenario
- Whitelist of size N.
- 10N transactions.

Hypothesis. Order of growth is \( N^2 \). Does NOT scale.

Binary search

- Keep the array in sorted order (stay tuned).
- Examine the middle key.
- If it matches, return its index.
- If it is larger, search the half with lower indices.
- If it is smaller, search the half with upper indices.

Match found. Return 10.
Binary search arithmetic

Notation. \( a[lo,hi) \) means \( a[lo], a[lo+1] \ldots a[hi-1] \) (does not include \( a[hi] \)).

<table>
<thead>
<tr>
<th>Search in ( a[lo,hi) )</th>
<th>( mid = lo + (hi-lo)/2 )</th>
<th>Lower half: ( a[lo,mid) )</th>
<th>Upper half: ( a[mid,hi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( lo )</td>
<td>( lo )</td>
<td>( lo )</td>
<td>( lo )</td>
</tr>
<tr>
<td>( hi )</td>
<td>( hi )</td>
<td>( hi )</td>
<td>( hi )</td>
</tr>
</tbody>
</table>

\( mid \rightarrow \)

\( mid \rightarrow \)

\( mid \rightarrow \)

\( mid \rightarrow \)

\( mid \rightarrow \)

Tricky! Needs study...

Binary search: Java implementation

```java
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) {
    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;
}
```

Recursion trace for binary search

```java
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;
}
```

Mathematical analysis of binary search

**Proposition.** Binary search uses \(-\lg N\) compares for a search miss.

**Proof.** An (easy) exercise in discrete math.

**Proposition.** Binary search uses \(-\lg N\) compares for a random search hit.

**Proof.** A slightly more difficult exercise in discrete math.
Empirical tests of binary search

<table>
<thead>
<tr>
<th>N</th>
<th>Tsw (seconds)</th>
<th>Tsw/Tlo</th>
<th>Transactions per second</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,000</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200,000</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>400,000</td>
<td>6</td>
<td>2</td>
<td>67,000</td>
</tr>
<tr>
<td>800,000</td>
<td>14</td>
<td>2.35</td>
<td>57,000</td>
</tr>
<tr>
<td>1,600,000</td>
<td>33</td>
<td>2.33</td>
<td>48,000</td>
</tr>
<tr>
<td>10.28 million</td>
<td>264</td>
<td>2</td>
<td>48,000</td>
</tr>
</tbody>
</table>

Whitelist filter scenario
• Whitelist of size N.
• 10N transactions.

Confirms hypothesis that order of growth is $\log N$.

Will scale.

Sorting: Rearrange N items to put them in ascending order

Applications
• Binary search
• Statistics
• Databases
• Data compression
• Bioinformatics
• Computer graphics
• Scientific computing
• ...
• [Too numerous to list]
Insertion sort trace

| 0 | wendy |
| 1 | alice |
| 2 | dave |
| 3 | walter |
| 4 | carlos |
| 5 | carol |
| 6 | erin |
| 7 | oscar |
| 8 | peggy |
| 9 | trudy |
| 10 | eve |
| 11 | trent |
| 12 | bob |
| 13 | craig |
| 14 | frank |
| 15 | victor |

Insertion sort algorithm

**Insertion sort**
- Move down through the array.
- Each item **bubbles up** above the larger ones above it.
- Everything above the current item is in order.
- Everything below the current item is untouched.

Like bubble sort, but not bubble sort.
**We don’t teach bubble sort any more because this is simpler and faster.**

**Insertion sort: Java implementation**

```java
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);
                    break;
                }

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

    public static void main(String[] args) {
        String[] a = StdIn.readAllStrings();
        sort(a);
        for (int i = 0; i < a.length; i++)
            StdOut.println(a[i]);
    }
}
```
Empirical tests of insertion sort

<table>
<thead>
<tr>
<th>$N$</th>
<th>$T_N$ (seconds)</th>
<th>$T_N/T_{N/2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20,000</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>40,000</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>80,000</td>
<td>35</td>
<td>9</td>
</tr>
<tr>
<td>160,000</td>
<td>225</td>
<td>6.4</td>
</tr>
<tr>
<td>320,000</td>
<td>1019</td>
<td>4.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1.28 million</td>
<td>14400</td>
<td>4</td>
</tr>
</tbody>
</table>

Confirms hypothesis that order of growth is $N^2$.

will NOT scale

A rule of thumb

Moore’s law. The number of transistors in an integrated circuit doubles about every 2 years.

Implications

- Memory size doubles every two years.
- Processor speed doubles every two years.

Sedgewick’s rule of thumb.

It takes seconds to access every word in a computer.

Scalability

An algorithm scales if its running time doubles when the problem size doubles.

2x faster computer with 2x memory using an alg that scales?

• Can solve problems we’re solving now in half the time.
• Can solve a 2x-sized problem in the same time it took to solve an x-sized problem.
• Progress.

2x faster computer with 2x memory using quadratic alg?

• Can solve problems we’re solving now in half the time.
• Takes twice as long solve a 2x-sized problem as it took to solve an x-sized problem.
• Frustration.

Bottom line. Need algorithms that scale to keep pace with Moore’s law.
Merge sort
- Divide array into two halves.
- Recursively sort each half.
- Merge two halves to make sorted whole.

Abstract inplace merge
- Merge a[lo, mid] with a[mid, hi).
- Use auxiliary array for result.
- Copy back when sort complete.

private static String[] aux;
public static void merge(String[] a, int lo, int mid, int hi) {
    // Merge a[lo, mid] with a[mid, hi) into aux[0, hi-lo).
    int i = lo, j = mid, N = hi - lo;
    for (int k = lo; k < N; k++)
        if (lo == N) aux[k] = a[i++];
        else if (j == N) aux[k] = a[j++];
        else if (a[i].compareTo(a[j]) < 0) aux[k] = a[i++];
        else aux[k] = a[j++];
    // Copy back into a[lo, hi)
    for (int k = lo; k < N; k++)
        a[lo + k] = aux[k];
}

public class Merge {
    private static String[] aux;
    public static void merge(String[] a, int lo, int mid, int hi) {
        // See previous slide.
        public static void sort(String[] a) {
            // Sort a[lo, hi).
            sort(a, 0, a.length);   } `sort(a, 0, a.length);`
            if (N <= 1) return;
            int mid = lo + N/2;
            sort(a, lo, mid);
            sort(a, mid, hi);
            merge(a, lo, mid, hi);
        }
    }
}

Merge: Java implementation

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

% more names16.txt
wendy
alice
dave
walter
carlos
erin
oscar
frank
trent
peggy
craig
victor
% java Merge < names16.txt
alice
dave
carlos
erin
walter
frank
trent
craig
victor
wendy

Merge sort: Java implementation

% more names16.txt
wendy
alice
dave
walter
carlos
erin
oscar
frank
trent
peggy
craig
victor
% java Merge < names16.txt
alice
dave
carlos
erin
walter
frank
trent
craig
victor
wendy
Mergesort analysis

Cost model. Count data moves.

Exact analysis for \( N = 2^n \).

- Note that \( n = \log_2 N \).
- \( 1 \) subarray of size \( 2^0 \).
- \( 2 \) subarrays of size \( 2^1 \).
- \( 4 \) subarrays of size \( 2^2 \).
- ... 
- \( 2^n \) subarrays of size \( 1 \).
- Total # data moves: \( 2N \log N \).

Empirical tests of mergesort

Sort random strings
- Array of length \( N \).
- 10-character strings.

\[
\begin{array}{|c|c|c|}
\hline
N & T_N & T_{rd2} \\
\hline
1 \text{ million} & 1 & \text{seconds} \\
2 \text{ million} & 2 & \text{seconds} \\
4 \text{ million} & 5 & 2.5 \text{ seconds} \\
8 \text{ million} & 10 & 2 \text{ seconds} \\
16 \text{ million} & 20 & 2.5 \text{ seconds} \\
\cdots & \cdots & \cdots \\
1.02 \text{ billion} & 1280 & 2 \text{ minutes} \\
\hline
\end{array}
\]

Confirms hypothesis that order of growth is \( N \log N \). WILL scale.

Detecting repeats in a string

Longest repeated substring
- Given: A string \( s \).
- Task: Find the longest substring in \( s \) that appears at least twice.

Example 1.
\[
\text{a c a a g t t t a c a a g c}
\]

Example 2.
\[
\text{a c a a g t t t a c a a g t t t a c a a g c t a g c}
\]

Example 3 (first 100 digits of \( \pi \)).
\[
\begin{align*}
3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117069
\end{align*}
\]
LRS example: repetitive structure in music

Mary had a little lamb

Fur Elise

source: http://www.bbewitched.com/match/
Two alternatives for implementing substrings

1. Refer to original string.
   - No need to copy characters.
   - Constant time.

2. Copy the characters to make a new string.
   - Allows potential to free up memory when the original string is no longer needed.
   - Linear time (in the length of the substring).

### LRS: An efficient solution that uses sorting

**Form suffix strings**
- Genome: aacaagttttacaagc

**Sort suffix strings**
- Sorted suffixes: 1. aacaagttttacaagc, 2. aagttttacaagc, 3. gttttacaagc, 4. tttacaagc, 5. tacaagc, 6. caagc, 7. agc, 8. g, 9. a

**Find longest LCP among adjacent entries.**

### LRS implementation

```java
public class LRS {
    public static String lcp(String s) {
        // See previous slide.
        int N = s.length();
        String[] suffixes = new String[N];
        for (int i = 0; i < N; i++) {
            suffixes[i] = s.substring(i, N);
        }
        String[] sortedSuffixes = sort(suffixes);
        int[] longestLCPs = findLongestLCPs(sortedSuffixes);
        return longestLCPs[0];
    }
}
```

### LRS: Empirical analysis

**Model**
- Alphabet: actg.
- N-character random strings.

<table>
<thead>
<tr>
<th>Doubling</th>
<th>N</th>
<th>Tn</th>
<th>Tn/Tn/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2,000,000</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4,000,000</td>
<td>7</td>
<td>2.3</td>
<td></td>
</tr>
<tr>
<td>8,000,000</td>
<td>16</td>
<td>2.3</td>
<td></td>
</tr>
<tr>
<td>16,000,000</td>
<td>39</td>
<td>2.4</td>
<td></td>
</tr>
</tbody>
</table>

Confirms hypothesis that the order of growth is $N \log N$ (for the sort).

**Bottom line.** Scales with the size of the input and enables new research and development.
Important notes on LRS implementation

Long repeats
- More precise analysis reveals that running time is quadratic in the length of the longest repeat.
- Model has no long repeats.
- Real data may have long repeats.
- Linear time algorithm (guarantee) is known.

String representation
- Efficiency depends on constant-time substring operation.
- 1995–2012: Java substring is constant-time.
- 2013: Java 7 changes to linear-time substring operation!
  (breaks this and many classic algorithms).
- Need to implement our own constant-time substring.

Example: Chromosome 11 has a repeat of length 12,567.

Summary

Binary search. Efficient algorithm to search a sorted array.
Merge sort. Efficient algorithm to sort an array.
Applications. Many, many, many things are enabled by fast sort and search.

Hey, Bob. Our IPO is next week!
I think I’ll take a few CS courses.